ON-LINE NOVELTY DETECTION THROUGH SELF-ORGANISATION, WITH APPLICATION TO INSPECTION ROBOTICS

A THESIS SUBMITTED TO THE UNIVERSITY OF MANCHESTER
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
IN THE FACULTY OF SCIENCE AND ENGINEERING

December 2001

By
Stephen Marsland
Department of Computer Science

Contents

List of Tables

List of Figures

Abstract

Novelty detection, the recognition that a perception differs in some way from the features that have been seen previously, is a useful capability for both natural and artificial organisms. For many animals the ability to detect novel stimuli is an important survival trait, since the new perception could be evidence of a predator, while for learning machines novelty detection can enable useful behaviours such as focusing attention on novel features, selecting what to learn and – the main focus of this thesis – inspection tasks.

There are many places where an autonomous mobile inspection robot would be useful – examples include sewers, pipelines and even outer space. The robot could explore its environment and highlight potential problems for further investigation. The challenge is to have the robot recognise the evidence of problems. For inspection applications it is better to err on the side of caution, detecting potential faults that are, in fact not problems, rather than missing any faults that do exist. However, by training the robot to recognise each individual fault, other problems will be missed. This is where novelty detection is useful. Instead of training the robot to recognise the faults, the robot learns a model of the 'normal' environment that does not have any problems and the novelty filter detects deviations from this model.

In training the robot it may well be found that the initial training set was deficient in some way, for example some feature that should be found normal was missing and is therefore always detected as novel. To deal with this situation the novelty filter should be capable of continuous on-line learning, so that the filter can learn to recognise the missing feature without having to relearn every other perception.

This thesis introduces a novelty filter that is suitable for the inspection task. The novelty filter uses a model of the biological phenomenon of habituation, a decrement in behavioural response to a stimulus that is seen repeatedly without ill effects, together with an unsupervised neural network that learns the model of normality. A variety of neural networks are investigated for suitability as the basis of the novelty filter on a

number of robot experiments where a robot equipped with sonar sensors explores a set of corridor environments.

The particular needs of the novelty filter require a self-organising network that is capable of continuous learning and that can increase the number of nodes in the network as new perceptions are seen during training. A suitable network, termed the 'Grow When Required' network, is devised. The network is applied to a variety of problems, initially non-novelty detection classification tasks, at which its performance compares favourably to other algorithms in terms of accuracy and speed of learning, and then a series of inspection problems – both robotic and not – again with promising results. In addition to the sonar sensors that were used for the earlier robotic inspection tasks, the output of a CCD camera is also used as input. Finally, an extension to the novelty detection algorithm is presented that enables the filter to store multiple models of a variety of environments and to autonomously select the best one. This means that the filter can be used in a set of environments that demonstrate different characteristics and can automatically select a suitably trained filter.

Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institution of learning.

Copyright

Dedication

For Samina, my angel,
and my parents.

Acknowledgements

There are many people who have helped me immensely over the past three years. Firstly, the greatest thanks are due to my supervisors, Ulrich Nehmzow and Jon Shapiro, whose completely different supervisory styles have complemented each other perfectly.

Another great debt is to my fellow Ph.D. students (and others) in the Artificial Intelligence group, Tom Duckett, Ten-min Lee, Jason Fleischer, Howsun Jow, Andy Johnson, Nickie John Player, Susie Lydon and Crispin Miller. Thanks for talks, laughs, proof-reading and friendship. I am particularly grateful to Tom and Jason who have read early drafts of this thesis. Similar thanks to my friends from the Santa Fe Institute and Bremen University.

The lecturers in the AI group, David Brée, Magnus Rattray, Mary McGee-Wood and Ian Pratt-Hartmann have provided valuable feedback during seminars and talks, and the use of the robots, especially the rapidly decaying *FortyTwo* would have been impossible without Andy Pickering. This PhD was supported by an EPSRC studentship.

Finally, I thank Samina Munir, whose help and encouragement have been unfailing.

Chapter 1

Introduction

This first chapter discusses what novelty detection is and why it is worthy of study. A set of aims and objectives for the research are derived, which will enable the work to be evaluated in the final chapter.

1.1 A Description of the Problem

This first section provides a brief description of why novelty detection is useful and the types of application to which a novelty filter could usefully be applied. Following this, there is a discussion of the definition of novelty detection and the practical problems that are associated with detecting novelty.

1.1.1 Motivation

Novelty detection, recognising that a perception is different from those that you have seen before, is a very important survival instinct for many animals. The unexpected perception could be a potential predator or a possible victim. By detecting novel features, the animal's attention is directed first to the most potentially dangerous features of its current environment (as it has experienced the other, less novel, perceptions and lived to remember them, presumably they are less than fatal). Novelty detection, then, is a way of reducing the large amounts of extraneous information that the animal is receiving, so that the animal can concentrate on the most important features.

Examples like these from biology suggest some of the ways in which novelty detection can be of interest to artificial learning systems. For instance, novelty detection

can be used to suggest where an agent should focus its attention. Although novel perceptions are rarely of fatal importance to robots, they may be more interesting than other perceptions for many tasks. By directing the attention of the robot to particular features, and so ignoring others, novelty detection significantly reduces the amount of information that has to be processed at the current time. Such uses of a novelty filter can be classified as preprocessing – the novelty filter is used to select which of a set of inputs should be given further attention, and possibly responded to, by the agent – be it animal or robot. Another use of the novelty filter in these circumstances is to decide what should be learned – there is no need to learn further examples of inputs that became familiar long ago, better to conserve resources for when something new comes along.

Other important cases where novelty detection is especially useful do not have clear parallels in biology. One case in particular has received a lot of attention in the neural network literature – novelty detection provides a way of dealing with the problem that one important class is under-represented in the data. Typical examples of this type of data are medical diagnosis and machine fault detection. In both cases there are plenty of examples of smooth running – healthy test results and normal machine operation, but relatively few instances of a particular disease, the signature of a component under stress, or whatever, and therefore a network cannot be trained to recognise the feature reliably.

Circumstances such as these have other limiting factors. There is never a guarantee that every example of the interesting class – the disease or breakdown – has been seen. Having categorised 50 examples, perhaps two diseases will display symptoms simultaneously, or a new variant may occur, and therefore the problem will be missed because it looks different. In examples such as these, false positives (categorising healthy examples as unhealthy) are preferable to false negatives, as further attention will discover the error, while not recognising a problem is potentially fatal.

These types of task can be thought of as inspection tasks. The novelty filter is given a model of normality (the healthy data) and then has to detect deviations from this model that could signal problems. A similar type of application is inspection robotics, which is discussed in more detail in the section 1.2. For all of these reasons novelty detection is an interesting and useful technique for both animals and artificial learning systems. This section has given some detail about a few of the areas where novelty detection is a useful tool as a motivation for the study. These observations are expanded upon in section 1.2.

1.1.2 What is Novelty Detection?

As the previous section has suggested, identifying a definition of novelty is not an easy task. Intuitively it is easy to say whether something is novel – the strange fruit from a foreign country is novel, the alien that landed on the road on the way back from the pub was certainly novel and that paperback sure was a novel. Whether or not something is novel depends on whether it, or something similar, has been seen before. This is borne out by consultation of the dictionary; Chambers (1990) English Dictionary defines novelty as:

> *newness: unusual appearance: anything new, strange, or different from what was known or usual before*

and the adjective *novel* is defined as:

> *new: new and strange: of a new kind: felt to be new.*

However, the most telling definition comes from Merriam-Webster (1999):

> *new and not resembling something formerly known or used.*

So novelty is defined in terms of what has been seen before, novelty is a hole in memory, a failure in memory look-up. This makes it completely individual, since it is based on previous experience. The link with memory is important. There is evidence in psychology that when asked to memorise a stream of symbols the ones that are remembered best are those that differ in some way from the rest, that is, the novel ones. This is known as the von Restorff effect (von Restorff, 1933) and suggests that novel items are stored in memory in a different way to less novel stimuli.

Ignoring familiar stimuli so that the novel inputs stand out is something that animals do using habituation (Thompson and Spencer, 1966). Habituation is a reversible decrement in an animal's response to a stimulus that is seen repeatedly without ill effects. Further details of habituation, which will be a very important part of the novelty filter introduced in this research, are given in section 2.3.

Types of Novelty

There are different types of novelty. The types that are of interest in this thesis are described here:

Object novelty the object that has not been seen before and that does not resemble any previously seen object is novel

Environment novelty the place that has not been visited before is novel

Situation novelty the conjunction of non-novel objects can also be novel. As O'Keefe and Nadel (1978) put it (page 241):

> ... the novelty of the wife in the best friend's bed lies neither in the wife, nor the friend, nor the bed, but in the unfamiliar conjunction of the three.

The link between these categories is that a novel object is unexpected, either because it has not been experienced before or because seeing it *there* is inconsistent with previous experience. In many ways, object and environment novelty are similar – they are concerned with the novelty of the current perception, either the surroundings or some actual features, without regard to the rest of the situation. The difference is that object novelty requires the segmentation of the perceptions in order to recognise the individual objects. Situation novelty is rather different, because it requires the separation of the object from the environment and the recognition of both, together with the recognition that there is no known link between the two. However, in all cases, novelty detection is the task of recognising that a particular perception or set of perceptions has not been seen before, nor has anything similar.

1.1.3 Practical Problems

How Different Should a Stimulus Be?

How similar do two things have to be before seeing one of them means that seeing the second one is not novel, or only slightly novel? After seeing a few Granny Smiths, a Cox apple is only a little bit novel – it is different rather than unusual. However, an orange would still be novel after seeing hundreds of apples.

This serves to show that it is possible to generalise between different stimuli, providing that there are some common features that link them together. This is presumably a function of the processes in memory that record perceptions, and recognise similarities between stimuli. It will not be considered further here, except to note that a good novelty filter must be able to generalise between similar perceptions.

How Often Must a Stimulus Be Seen Before It Stops Being Novel?

When does a particular perception stop being novel? Must an observer see something once, ten times, one hundred times? The answer seems to be that it depends on the perception, the viewer and whether something at all similar has been seen before. Hence, no hard and fast answer can be given. Certainly, a feature that is seen a second time will usually be recognised and as such is not entirely novel, but equally, seeing an object for a second or third time, the observer will still be unsure of it and therefore afford it special regard, still treating it as novel. The amount of exposure is also important – a new environment, such as a new school or new office, stops being novel quickly because of the repeated exposure, the dentist's surgery visited only for the yearly check-up takes much longer.

The Desired Behaviour of a Novelty Filter

The preceding discussion has highlighted three important questions that require answers to define the behaviour of a novelty filter:

- What types of novelty should be detected?
- How well does the novelty filter generalise between inputs?
- How fast does the novelty filter stop finding a stimulus to be novel?

A novelty filter evaluates the current input with respect to some previously acquired model of normality. The most obvious way to detect situation novelty requires a number of other systems to be used first – perceptions of objects need to be separated from the environment, and each has to be individually identified (and recognised as not being novel) before the lack of a link between them can be discovered. This is different to the other two types of novelty, which only need to check whether a particular perception (be it an object or the whole environment) is novel, and situation novelty will not be dealt with explicitly in this research. A simplistic approach to recognising environment novelty automatically detects situation novelty in that if an environment that has been seen before is changed through the addition of some object, then the overall perception is different and will be found to be novel providing that the change is not too small.

It is obvious that a novelty filter should be able to generalise between inputs – if every single perception has to be seen, then a look-up table would be sufficient, and for

any sufficiently complex environment almost every perception would be novel. However, the amount of generalisation that is desirable will vary from application to application and as such should be a tunable parameter of the novelty filter. This requirement means that a machine learning solution is required, such as a neural network, to act as the basis for the novelty filter.

Another important question is how many times a perception should be seen before it stops being novel. For some applications one-shot learning is desirable, but often a stimulus should be seen several times before it is learnt fully. Ideally, the amount of novelty found in a perception should decrease from a maximum on first exposure to zero after some number of perceptions. It seems reasonable to expect a novelty filter worthy of the name to be able to tell quickly whether the current input is in line with previous inputs and whether it, or something similar, has never been seen, seen only occasionally, or is a familiar object. These considerations are used to form a set of objectives for the research in section 1.3.

1.2 Target Applications

1.2.1 Novelty Detection for Robot Inspection Tasks

One important application to which a novelty filter can be readily applied is that of inspection, as was suggested previously. However, as well as machine fault diagnosis, there is the related potential application of robotic inspection. Training a robot to inspect an environment and highlight problems is a very good use of a robot, but ensuring that it could recognise all potential problems would be at best extremely difficult. This is an example of the second class of problems for which novelty detection is useful – when there are too few examples of the target class and finding false positives is less of a problem than missing potential problems. It cannot be guaranteed that every example of a potential problem could be shown to a robot during training and that the robot could learn a model of all of them, and so novelty detection is useful because it does not require this.

There are a number of places where such an application would be very useful, for example anywhere that is dangerous, unpleasant or inaccessible for humans. Examples include outer space, accident sites, sewers and pipelines. A number of other researchers have looked at some aspects of this problem, particularly with respect to sewer inspection, see for example Hertzberg et al. (1998) and Paletta et al. (1999),

although they do not look at novelty detection, or even the inspection problem, but rather at how suitable robots should travel, perceive their environment and navigate in sewers.

A framework for such an application would take the form of a series of training runs in a number of environments known to be normal, that is to exhibit no problems; these could be examined by human inspectors, or designed especially for the task. Once the robot had learnt a perfect model of these environments, so that no perceptions were found to be novel, the robot could be allowed to explore the whole area under investigation, highlighting the features that were not seen in the training environments and that were therefore novel.

It is very likely that a number of 'normal' (i.e., non-fault) perceptions would be missed during the training, so that they were found to be novel during exploration of the wider environment. Provided that the novelty filter used was capable of continuous learning, these perceptions could be added to the model at a later date without any difficulties. Any novel features detected by the robot would be marked for further attention by human operators, changing the operator's task from following many hours of repetitive, boring data to examining a few 'snapshots' of problem areas. This application will act as the basis for this thesis; most of the experiments will consider on-line inspection tasks for mobile robots.

1.2.2 Other Applications to Mobile Robotics

In a very influential paper, Brooks (1986) suggested that, in order to appear intelligent, a robot would need to display a number of different behaviours that operate in parallel, with behaviours subsuming each other as is required by the application and environment. A diagram showing the behaviours he proposed in increasing order of complexity is given in figure 1.1. The bottom two of these behavioural layers (numbered 1 and 2) are trivial to implement, and the third poses few problems. However, at this point the complexity of the task increases considerably – over the past thirty years there has been an immense amount of research directed at level four, that of navigation. Even then, only in the last few years have systems that allow a robot to navigate competently in unstructured environments (Kurz, 1996; Burgard et al., 1998) and relocalise successfully when lost (Oore et al., 1997; Duckett and Nehmzow, 1998) been produced.

It is argued here that level five, 'monitor changes' is about novelty detection, or at

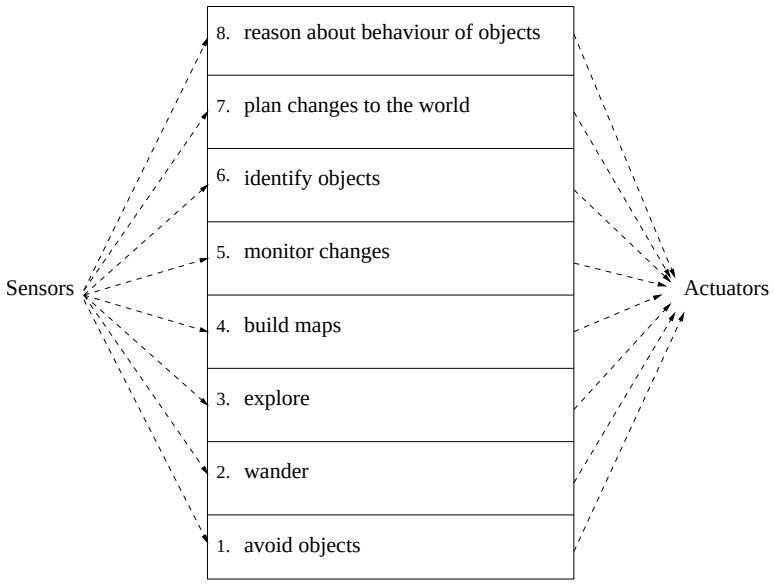


Figure 1.1: Task achieving behaviours for a mobile robot, adapted from Brooks (1986).

least that novelty detection is an important part of change monitoring. Brooks' intention was that the robot would be able to notice changes in the environment and respond to them if required. A robot that can detect changes in its surroundings will be able to deal with dynamic environments, currently a significant problem for navigation systems, which usually rely on detecting landmarks that are fixed within the environment and that can become obscured if the environment changes. However, this is only one of a number of advantages that the ability to monitor changes – of which novelty detection is a crucial part – can impart to a mobile robot. Two particular applications are discussed below.

One advantage that was mentioned in section 1.1.1 was concerned with attentional focus and cutting down the amount of information that the robot has to process. Directing attention has been investigated from a non-novelty detection viewpoint by a number of other researchers, notably Marom and Hayes (1999, 2000), who use attention to make a learner robot focus on the behaviour of a teacher, Balkenius (2000), who investigates simple models of how animals direct their attention towards novel stimuli and Scheier and Egner (1997), who investigate how visual attention can be useful for a mobile robot. The novelty filter that is the aim of this work should be a useful contribution to this area because it would enable the robot to concentrate on the novel features of its environment.

Another area where novelty detection could be useful in mobile robotics is in landmark selection. Typical robot navigation systems use perceptual landmarks to locate

the robot within the environment. Such systems often take landmarks at regular intervals, such as every 1 metre of travel (Yamauchi and Beer, 1996; Duckett and Nehmzow, 1998). This approach suffers from perceptual aliasing, where multiple locations are indistinguishable. By selecting as landmarks those perceptions that are novel, it is conjectured that consistent and comparatively rare landmarks can be selected. An example of such a system is given by Marsland et al. (2001).

1.2.3 Further Uses of Novelty Detection

The most popular applications of a novelty filter in the machine learning field were described in section 1.1.1. They are all based on applications where the examples of one important class in the training set are relatively rare, such as medical diagnosis and machine fault monitoring.

Chapter 2 describes a number of approaches to the problem that have been proposed in the literature. In the main, these approaches are based on batch training, where all the training is done before the novelty filter is ready for use. Then, if further data becomes available later, the whole training process has to be restarted. In addition, there is no indication of how novel an input is, either it fits into the acquired model or it does not.

Another area where novelty detection can be useful is in the related field of statistical outlier detection, deciding whether the current input is an extremely noisy example from the current input distribution or an unexpected datum. A novelty filter can be used to preprocess a dataset and remove any outlying datapoints before other machine learning techniques are used. Outlier detection is discussed further in section 2.8.

Having described some areas where novelty detection will be useful, the next section gives the aims and objectives of the research, as well as giving some assessment criteria by which the work will be evaluated in the final chapter.

1.3 Aims and Objectives

This section first describes the overall aims of the research. The aims are then broken down into a set of objectives that the research should meet. These objectives are stated as falsifiable statements, so that a proper scientific investigation of each of the objectives can be performed (Popper, 1959). The objectives will be returned to in section 9.2 of the concluding chapter and assessment criteria introduced, so that an analysis of the

success of the research can be performed.

1.3.1 Aims

The overall aims of this research can be given in the following sentences:

> To develop, and investigate the behaviour of, an adaptive novelty filter that is suitable for on-line use, that quantifies the amount of novelty in a perception with respect to an acquired model, and then adds the knowledge of the perception to the model. The novelty filter will be applied to the perceptions of a mobile robot as it performs simple inspection tasks, first exploring a number of environments, learning a model of these environments and then highlighting deviations from the model.

That is, this exploratory research aims to design a novelty filter that can operate on-line and is therefore suitable for use on a robot. The filter should be able to learn about perceptions and will therefore be based on a neural network. Experiments will be used to investigate the abilities of the filters that are proposed and to formulate how they should be improved for further development.

In themselves, these aims are not sufficiently descriptive, and therefore a set of objectives are given next. These objectives will be used to guide the development of the functionality of the novelty filter.

1.3.2 Objectives

This section describes the objectives of the research. These objectives were derived using the discussion in sections 1.1 and 1.2 above. The objectives are broken down into three areas – desired functionality of the novelty filter, experimental results that should be demonstrated, and desired functionality of the neural network that forms the basis of the novelty filter. The necessity of some of the objectives will become clearer as the thesis progresses.

- The novelty filter should:
  1. successfully detect and learn about novelty in the inputs
  2. generalise between similar inputs
  3. operate on-line

4. quantify novelty
5. always find perceptions novel if they are seen very infrequently
6. be able to forget about perceptions that are not seen for a long time
7. scale with the size of the input set
8. deal with a wide variety of inputs
9. handle multiple models of different environments

- The experiments should demonstrate that the novelty filter can:

10. enable a robot to focus its attention on novel features
11. enable a robot to highlight novel features of an environment
12. highlight novelty in other datasets

- The neural network basis for the novelty filter should:

13. be capable of continuous learning
14. learn parsimonious and efficient representations of the input space
15. preserve the topology of the input space

1.4 Summary

This section has motivated and described the proposed research, in addition to suggesting a number of areas where the novelty filter created during this work would be useful.

Three different types of novelty were identified:

Object novelty not seen *that* before

Environment novelty not been *there* before

Situation novelty not seen *that there* before

and the important features of a novelty filter were discussed. This discussion was used to make a set of objectives for the research, together with assessment criteria that will be used to evaluate the research in the final chapter. These objectives stress the factors that are important for the novelty filter and the experiments that will be used to investigate the properties of the filter.

Novelty detection has been shown to be useful to artificial learning systems. A number of applications to mobile robotics and machine learning have been given and the importance of enabling the novelty filter to operate on-line was stressed.

1.5 A Guide to the Thesis

Chapter 2 provides an overview of the literature on novelty detection, first in animals and then in artificial neural networks.

The aim of the first section is to demonstrate that animals can and do detect novelty for survival reasons, and to describe some of the ways in which this has been tested in the laboratory. Habituation, one of the ways in which animals filter out stimuli that are seen frequently, and which forms the basis of the proposed novelty filter, is described in detail.

The different approaches proposed by other researchers in the machine learning field are described and evaluated with respect to the objectives given in section 1.3 in order to demonstrate that, to date, no other novelty detection algorithm suitable for use on-line for a wide variety of different input modalities has been proposed, nor one that usefully quantifies the amount of novelty in a perception.

Chapter 3 introduces the novelty filter on which this research is based, first as an abstract idea and then a variety of different possible implementations are described and discussed.

Chapter 4 details experiments performed with the Habituating Self-Organising Map (HSOM), the implementation of choice from the previous section. Two different types of experiments are performed, one that introduces a behaviour of travelling towards new stimuli to the robot, and one that shows how the novelty filter can be used to create an on-line inspection agent, where the novelty filter learns a model of an environment perceived by the sonar sensors of the robot as it explores, and then evaluates perceptions with respect to this model.

It is shown that the HSOM has problems with continuous learning in that the network can saturate, so that all perceptions are considered to be normal, no matter what they are, if the number of different perceptions approaches the order of the number of nodes in the network.

Chapter 5 investigates possible ways round this problem in the literature on growing neural networks, but finds that none of the networks described are suitable. A new network, termed the 'Grow When Required' (GWR) network, is therefore introduced that adds new nodes into the map space whenever necessary. A number of non-novelty detection datasets are used to demonstrate the behaviour of this network and an analysis of the topology preservation and learning abilities is given.

Chapter 6 tests the new network on the robot inspection tasks that were performed in chapter 4, in order to demonstrate that the principles still work well. The ability of the network to scale with the size of the environment is demonstrated and compared to results from networks proposed in the literature.

In addition, this chapter tests the novelty filter based on the new network on novelty detection tasks from other domains; medical diagnosis, machine fault detection and test data taken from oil pipelines.

Chapter 7 extends the range of input modalities that the novelty filter can deal with by testing different ways of generating an input vector from complex camera images. The resulting systems are tested in the environments used for the experiments with the robot's sonar sensors that were described in chapters 4 and 6.

Chapter 8 completes the experimental sections by showing that a simple extension to the algorithm allows one of a battery of different pre-trained novelty filters to be selected autonomously, so that a novelty filter suitable for the particular environment can be selected.

Chapter 9 concludes the thesis with a summary, evaluation of the research, open questions and a discussion.

Chapter 2

Background Literature on Novelty Detection

The literature on novelty detection is reviewed in this chapter. Novelty detection in biological systems is discussed first, and then neural network novelty filters that have been proposed are described.

"*A rip-roaring roller coaster of a read, bursting out of its leather pants with sex and drugs and rock and roll.*" Robert Rankin

2.1 Introduction

In this chapter the literature on novelty detection is discussed. The first sections provide a brief overview of novelty detection capabilities found in animals, and detail theories that have been described in the biological and psychological literature of how and why novelty is detected. Later sections describe the research that has been performed in the machine learning field, evaluating each method as to how well it fulfils the objectives for a novelty filter suitable for the robot inspection tasks that were described in section 1.3.

Animals can and do detect novel stimuli in their surroundings. Any stimulus that the animal does not expect to see provokes an immediate response as the animal decides how to respond to it – possible responses are known as the four Fs (flee, fight, feed and reproduce). This has a clear survival benefit.

One method by which animals learn to ignore stimuli that are seen often, so that only comparatively rare features are given attention by the animal, is habituation,

thought to be one of the simplest forms of learning. As an animal perceives a stimulus repeatedly without ill effects, it responds to it progressively less, until eventually it is ignored. This is one possible behaviour of an on-line novelty filter – perceptions that are seen for the first time (novel perceptions) receive a strong response, but as they are seen again and again the response is reduced.

Most of this chapter describes the methods of novelty detection that have been proposed in the machine learning literature. Many of these techniques are designed for batch learning, with a neural network being trained off-line on a set of inputs known to contain no features of any class that should be detected (the 'novel' class), and then highlighting test inputs that do not belong to trained classes. This approach is brittle, in that if a few examples of the novel class are included in the training set, elements of that class will not be found to be novel.

One particularly interesting filter is the earliest neural network-based novelty filter, that of Kohonen and Oja (section 2.4), which builds an adaptive model of an input set and then calculates the bit-wise difference between the current input and the closest match amongst the training set. Other approaches to novelty detection that are described include kernel density estimation (section 2.6.2) where the probability density function for different parts of the input space is used to suggest how confidently the predictions of a trained network can be treated and methods based on self-organising networks (section 2.7).

The related topic of statistical outlier detection is introduced in section 2.8, together with some of the techniques that are used in this area. Finally, a number of less well-known methods are described, and a summary highlights which of the criteria derived in section 1.3 have not been met by the filters described.

The novelty filters that are described are evaluated according to some of the criteria that were derived in section 1.3. Of particular interest are the following abilities of the novelty filter:

- to generalise between similar inputs
- to operate on-line
- to quantify novelty
- to also find perceptions novel if they are seen infrequently
- to be able to forget about perceptions that are not seen for a long time

- to scale with the size of the input set

Many of these features are not necessary for a novelty filter that is only required to run on a simple dataset that is known in advance, and that operates in circumstances where the training set can be monitored very carefully to ensure that it contains no examples of novel features. However, this is not always the case. In particular, for the robot inspection task that is the primary application of this thesis, this is not true – the environments may not be completely controllable and the robot may need several training runs in an environment to see every feature.

2.2 Biological Novelty Detection

Researchers in biology and psychology have been studying the ability of animals to detect novelty for a long time. This section presents an overview of some of the more salient work so that some of the effects of novelty detection in animals can be seen. Many animals respond to unexpected stimuli with an orienting response that demonstrates that novelty causes fear (O'Keefe and Nadel, 1978). This is followed by an exploration phase, where the animal carefully begins to explore its environment again.

As was discussed in chapter 1, novelty is a property of items or places that have not been experienced before, that is, stimulation that has never before been encountered. It can also include the unfamiliar conjunction of familiar objects (the situation novelty of chapter 1). To O'Keefe and Nadel this implies that memories of items include their context. They describe several sets of experiments by a number of different researchers that have investigated the responses of a variety of animals (fish, rats, gerbils, etc.) to novel stimuli. In one particularly interesting case rats were subjected to three types of novel occurrence:

- introducing a novel item in a familiar environment
  - with the animal engaged in directed activity (competitive)
  - with the animal resting (non-competitive)
- introducing the animal into a novel environment
- spontaneous alteration of the environment

The results of the experiments show an interesting dichotomy between the response of hippocampal animals (that is, animals with hippocampal lesions) and control animals. In general, normal animals appear to be distracted by unexpected features and will explore them, finishing any task that they were already performing later, after they had dealt with the interruption, while hippocampal animals will not explore the novel stimuli. There is evidence that they do recognise that the stimulus is novel, but just ignore it. This suggests that the hippocampus is involved in dealing with novelty and that it controls the extent to which the exploration impulse overcomes fear. So curiosity may have killed the cat, but only if its hippocampus was intact.

The hippocampus has also been shown to be a critical part of the brain network that detects novelty in humans. This was demonstrated by Knight (1996) using electrophysiological recording of scalp event-related potentials (ERPs). A series of target and novel stimuli (tones and finger taps) were embedded in a stream of repetitive background stimuli, and the ERPs recorded throughout the experiment. Experiments on patients with damage to the hippocampus showed that these patients responded less to novel events than control patients. These results are not surprising when the fact that the hippocampus is widely thought to be involved in the processes of memory is considered – any type of novelty detection requires the recall of previously seen stimuli.

One method of testing for novelty detection in humans that is used in the laboratory is to test the *von Restorff* effect. This is the finding that, after presentation of a number of stimuli, recall is better for isolates (i.e., a stimulus that differs from the others along some dimension, such as being a different colour or size) than for non-isolates (von Restorff, 1933; Parker et al., 1998). Evidence from tests like these suggest that novelty detection is important for coding information in memory (Brown, 1996). This makes sense because it may reduce the demands made on long-term memory, if perceptions have been seen before then there is no need to remember them. Further evidence of this is provided by the CHARM model of memory storage and retrieval (Metcalfe, 1993), where the storage of a stimulus in memory depends on an assessment of how similar the current stimulus is to memories already laid down.

Further experiments are described by Pribram (1961, 1992), who tested the attraction of rhesus monkeys with a lesioned frontal cortex (frontal monkeys) to novel stimuli. In the experiments, frontal monkeys and controls were presented with a board that had twelve holes drilled in it. A peanut was placed under one of the holes and that hole was covered with an object. All of the monkeys quickly learnt to uncover the

hole and find the peanut. However, when a second (novel) object was introduced to cover another hole, and the peanut placed under the new object, normal monkeys took several trials to learn to lift the new object, rather than the old one where previous experience had shown the object to be. This experiment was repeated using a third object as well as the other two, with the same results. Only after about six objects had been added to the board did normal monkeys associate the reward with the novel cue rather than the one that had previously received the reward. In contrast, frontal monkeys were attracted to the novel stimulus immediately, always went for the new object and were therefore rewarded. There is no report of the experiment in reverse, to see if frontal monkeys would learn to look under an object that was not novel if this was reinforced. It would have been interesting to see if it took frontal monkeys as long to learn to pick the object that was reinforced in the previous trial rather than the novel object.

Another area of the brain that is thought to be important for novelty detection is the perirhinal cortex. Brown and Xiang (1998) demonstrate that this region is instrumental in the judgement of prior occurrence. The authors describe the existence of three types of neuron useful for this task, which they claim are found throughout the anterior inferior temporal cortex (which includes perirhinal cortex): *recency neurons* that fire strongly for perceptions that have been seen recently, whether or not they are familiar, *familiarity neurons* that give information about the relative familiarity of a stimulus, and *novelty neurons* that respond strongly to presentations of novel stimuli.

Animals need to know how to focus on the novel stimuli amongst the huge number of features that are perceived. One way that this is done is by responding less to features that are seen repeatedly without ill effect. In the psychological literature this ability is known as habituation, and is the subject of the next section.

2.3 Habituation

2.3.1 What Habituation Is

Habituation is a reversible decrement in behavioural response to a stimulus that is seen repeatedly without any ill effects. It is thought to be one of the simplest examples of plasticity in the brain, and as such has attracted a lot of interest. Habituation can be thought of as a way of defocusing attention from features that are seen often, allowing the animal to concentrate fully on other, potentially more dangerous, experiences. Evidence of habituation can be seen clearly in an animal as simple as the sea snail *Aplysia*.

This mollusc has a gill that is withdrawn when its siphon is touched. However, repeated gentle stimulation of the siphon results in an habituated response meaning that the gill is withdrawn less and less, and finally not at all. Repeated series of training show that, while the defensive withdrawal returns over time (dishabituation), further stimulation habituates faster. As well as its simplicity there is another reason why habituation has generated so much interest – it occurs in almost all animals, and affects the behaviour of the animal throughout an experiment. Once an animal has perceived a stimulus several times, it will respond to it less because of habituation. This can clearly have a large impact on the experiment. As Zeaman (1976) puts it,

> "Habituation is like rats and cosmic rays. If you are a psychologist, it is hard to keep them out of your laboratory."

Habituation has been investigated by a large number of researchers, and in a wide variety of different animals, from *Aplysia* (Bailey and Chen, 1983; Byrne and Gingrich, 1989; Castellucci et al., 1978; Greenberg et al., 1987) through to cats (Thompson, 1986) and toads (Ewert and Kehl, 1978; Wang and Arbib, 1991b). There are also books on aspects of habituation, which give a wide overview of the field from the psychological angle (Peeke and Herz, 1973a; Tighe and Leaton, 1976) and the neuronal (Peeke and Herz, 1973b) angle.

A complete definition of habituation was provided by Thompson and Spencer (1966), who defined nine criteria that describe habituation, based on their studies of the phenomenon. Their main aim was to differentiate habituation from other forms of decrement in behaviour, such as fatigue. The criteria are given below:

1. Given that a particular stimulus gives a response, repeated application of the stimulus leads to a decreased response; usually this is a negative exponential function of the number of presentations

2. If the stimulus is withheld, the response recovers over time

3. If repeated series of training and recovery are used, habituation should become more rapid. This is known as potentiation

4. The more rapid the frequency of stimulus presentation, the more rapid the habituation

5. The weaker the stimulus, the more rapid the habituation

6. The effects of habituation training may go beyond zero or the asymptotic response level

7. A stimulus can be generalised to other stimuli

8. Presentation of a non-generalised stimulus leads to dishabituation

9. Repeated application of a dishabituation stimulus leads to a habituation of the amount of dishabituation (that is, if a stimulus is habituated and then dishabituated repeatedly (so that the animal stops responding to it and then responds again), eventually the amount of dishabituation reduces, which means that the animal stays habituated to the stimulus)

2.3.2 Models of Habituation

A number of authors have described models of the quantitative effects of habituation, as measured with a particular behavioural response. These models can be considered to represent the cellular processes of habituation as they occur in simple organisms.

The first model proposed was that of Stanley (1976). This model is based on the work of Groves and Thompson (1970) and follows the dual-process theory of habituation. The dual-process theory states that the response to a stimulus is controlled by the output of two independent channels, a sensitising process and an habituation process, with the overall behavioural outcome being a summation of the two channels. One of the effects of the sensitisation channel is to enable dishabituation. The two process channels can be seen in figure 2.1, which shows Stanley's proposed circuit. The output of cells H, X and O are given by equations 2.1, where I is the external stimulus, labels n, h and s on synapses represent non-plastic, habituating and sensitising synapses respectively, and w_0, w_1, etc. represent the strengths of the synapses.

$$\begin{aligned} H &= w_0 I \\ X &= w_1 H \\ O &= w_2 H + w_3 X . \end{aligned} \tag{2.1}$$

The equation that controls the value of the habituation synapses (labelled h in figure 2.1) is given in equation 2.2:

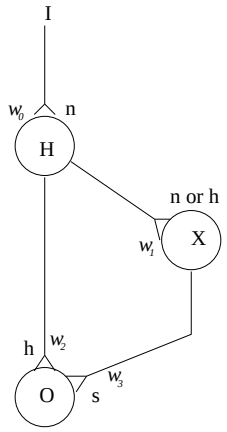


Figure 2.1: The two process circuit model proposed by Stanley (1976). Cell H receives external input from I, and propagates it to cells X and O via a combination of habituating synapses (h), sensitising synapses (s) and non-plastic synapses (n) that have strengths w_i for $i = 0,..,3$.

$$\tau \frac{dy(t)}{dt} = \alpha [y_0 - y(t)] - S(t), \tag{2.2}$$

where y_0 is the initial value of the weight y, $S(t)$ is the external stimulation and τ and α are time constants governing the rate of habituation and recovery rate respectively. A graph showing the effects of equation 2.2 is given in figure 2.2.

Equation 2.2 can be solved without difficulty:

$$y(t)e^{\frac{\alpha t}{\tau}} + c = \int \left(\frac{-S(t)}{\tau} + \alpha \frac{y_0}{\tau} \right) e^{\frac{\alpha t}{\tau}} dt. \tag{2.3}$$

This allows for two different behaviours, depending on whether or not a stimulus S is being applied. Assuming that a constant non-zero stimulus S is applied,

$$y = y_0 - \frac{S}{\alpha} \left(1 - e^{\frac{-\alpha t}{\tau}} \right), \tag{2.4}$$

and when the stimulus is withdrawn, so that $S = 0$, the solution is

$$y = y_0 - (y_0 - y_1) e^{\frac{-\alpha t}{\tau}}, \tag{2.5}$$

where the stimulus was withdrawn when the value of y was $y = y_1$. The two behaviours are shown in figure 2.2. The first behaviour, detailed in equation 2.4, describes a drop in the efficacy of the synapse, as is seen initially in the figure, while the second behaviour (equation 2.5) gives the part of the graph where the efficacy recovers to y_0, its original value (between 60 and 100 presentations in figure 2.2).

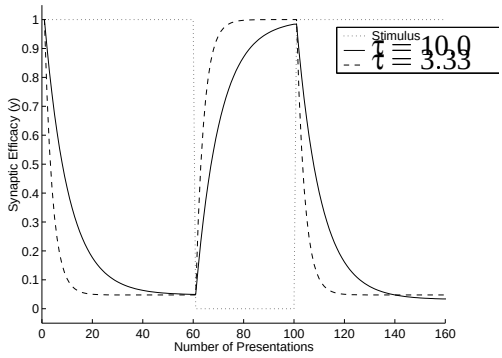


Figure 2.2: An example of how the synaptic efficacy drops when habituation occurs using Stanley's model (equation 2.2). In both curves a constant stimulus $S(t) = 1$ is presented, causing the efficacy to fall. The stimulus is reduced to $S(t) = 0$ at time $t = 60$ where the graphs rise again and becomes $S(t) = 1$ again at $t = 100$, causing another drop. The two curves show the effects of varying τ in equation 2.2. It can be seen that a smaller value of τ causes both the learning and forgetting to occur faster. The other variables were the same for both curves, $\alpha = 1.05$ and $y_0 = 1.0$.

Similar models were proposed by Lara and Arbib (1985) and Staddon (1992). Stanley's model (equation 2.2) only models the short-term effects of habituation, in that once a stimulus has dishabituated, it will not habituate any faster a second time. This is not what is found in biological investigations, where a stimulus that has habituated once habituates faster a second time (potentiation, point 3 in the list of Thompson and Spencer (1966)). The first model incorporating long-term habituation was produced by Wang and Hsu (1990), who used these equations:

$$\tau \frac{dy(t)}{dt} = \alpha z(t)\,[y_0 - y(t)] - \beta y(t) S(t), \tag{2.6}$$

$$\frac{dz(t)}{dt} = \gamma z(t)\,[z(t) - 1] S(t). \tag{2.7}$$

This pair of equations gives an S-shaped curve that displays both short-term and long-term effects of habituation. β controls the speed of habituation and the input $S(t)$ is gated by being multiplied by $y(t)$ instead of the direct input of equation 2.2. The new variable, $z(t)$, changes slowly compared with $y(t)$ and is used to control the rate of recovery. It has a single point of inflection, above which recovery is rapid, corresponding to short-term memory; below the point of inflection long-term effects dominate.

2.3.3 The Habituation Hierarchy

Ewert and Kehl (1978) demonstrated the existence of an habituation hierarchy, where some stimuli dishabituate others, but the same is not true the other way round. A pair of stimuli are at the same level in the hierarchy if their effects on each other are symmetrical, that is showing stimulus A followed by stimulus B is equivalent to showing stimulus B followed by stimulus A.

It was suggested by Wang and Ewert (1992) that the cause of the hierarchy is that dishabituation is a return to normal behaviour (the sensitisation channel of the dual-process theory), and cross-talk between the habituation and dishabituation process causes the hierarchy. The paper of Wang and Ewert (1992) is one of a series that attempt to model the neural processes underlying the orienting reflex and prey-catching behaviour in toads (genus *bufo bufo*), and its habituation. The other papers model the tectal relay and anterior thalamus (Wang and Arbib, 1991a,b), the areas that process the image taken from the retina, and the medial pallium (Wang and Arbib, 1992), the analogue in reptiles of the mammalian hippocampus, and the region in which habituation is thought to take place.

The experiments reported in these papers all took the same approach. A toad sat in a cylindrical glass jar and an electrically driven prey dummy was moved at $20°s^{-1}$ in a horizontal plane, 7 cm in front of the toad. When the toad recognised the dummy as prey, the toad followed it by making turning movements to orient itself towards the dummy. The number of orienting responses per minute were measured for the duration of the presentation of the prey dummy in order to measure the amount of habituation. By using a variety of prey dummies with different appearances it was noted that in some cases, after habituating to one dummy, the toad did not respond to another, although this was not true the other way round – if the toad habituated to the second dummy, it still responded to the first. This was taken to demonstrate the existence of the hierarchy.

Habituation has been used in an artificial neural network, too. Stiles and Ghosh (1995, 1997) consider the problem of how dynamic neural networks can be used to recognise temporal patterns in data. Their solution is to include an habituation term on the weights that connect the inputs to the neural network, a multi-layer perceptron or radial basis function network. These weights then have short-term temporal information in them, which affects the dynamics of the network.

2.4 Kohonen and Oja's Novelty Filter

2.4.1 The Novelty Filter

The first known adaptive novelty filter is that of Kohonen and Oja (1976), who proposed an orthogonalising filter that extracts the parts of an input vector that are 'new', with respect to previously learnt patterns. This is the desired functionality of a novelty filter. Another description of the filter is given in (Kohonen, 1993). The Kohonen and Oja novelty filter is a pattern matching algorithm with the following properties:

- patterns that are seen in training are stored
- inputs are compared to each of these stored patterns
- the best-matching stored pattern is selected
- the difference between the best-matching replica and the input is displayed

Mathematically, the effects of the novelty filter can be described as follows. Let $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_m} \in \mathbb{R}^n$ be distinct Euclidean vectors spanning a subspace $L \subset \mathbb{R}^n$. Then any vector $\mathbf{x} \in L$ can be uniquely written as

$$\mathbf{x} = \hat{\mathbf{x}} + \tilde{\mathbf{x}}, \tag{2.8}$$

where $\hat{\mathbf{x}} \in L$ is the orthogonal projection of $\mathbf{x}$ on L, and $\tilde{\mathbf{x}} \in L^{\perp}$ is the projection of $\mathbf{x}$ on $L^{\perp}$, the complement space that is orthogonal to L.

It can be shown that the decomposition in equation 2.8 has the property that $\|\tilde{\mathbf{x}}\|$ is the distance of $\mathbf{x}$ from L, i.e.,

$$\text{norm}\|\tilde{\mathbf{x}}\| = \min_{\|\hat{\mathbf{x}}\|} \|\tilde{\mathbf{x}}\|. \tag{2.9}$$

Let $A \in \mathbb{R}^{n \times m}$ be a matrix with $\mathbf{x_i}$ as the ith column. Then

$$\hat{\mathbf{x}} = AA^{+}\mathbf{x}, \tag{2.10}$$

where A^{+} is the pseudoinverse (Penrose, 1955) of A, and so

$$\tilde{\mathbf{x}} = (I - AA^{+})\mathbf{x}, \tag{2.11}$$

with I being the $2n \times 2n$ identity matrix.

The Gram-Schmidt process can be used to compute the orthogonal projections of vectors. A new vector basis is defined by the following recursion for the subspace L spanned by training vectors $\{\mathbf{x_i}\}, i = 1, \ldots, m$:

$$\tilde{\mathbf{x}}_\mathbf{1} = \mathbf{x_1}, \tag{2.12}$$

$$\tilde{\mathbf{x}}_\mathbf{k} = \mathbf{x_k} - \sum_{i=1}^{k-1} \frac{\mathbf{x_k}\tilde{\mathbf{x}}_\mathbf{i}^T}{\|\tilde{\mathbf{x}}_\mathbf{i}\|^2}\tilde{\mathbf{x}}_\mathbf{i}, \tag{2.13}$$

where the sum is over the $\|\tilde{\mathbf{x}}_\mathbf{i}\| \neq 0$. Then

$$\tilde{\mathbf{x}} = \tilde{\mathbf{x}}_{\mathbf{m+1}} \tag{2.14}$$

and

$$\hat{\mathbf{x}} = \mathbf{x} - \tilde{\mathbf{x}}_{\mathbf{m+1}}, \tag{2.15}$$

so that $\tilde{\mathbf{x}}$ is the residual $\notin L$, i.e., the part of $\mathbf{x}$ that is independent of the vectors $\{\mathbf{x_i}\}$. This is the amount of $\mathbf{x}$ that is 'maximally new', the 'novelty' in $\mathbf{x}$.

So a neural network that has equation 2.11 as the transfer function will extract the novelty in the input. Kohonen and Oja (1976) propose a network with neurons that implement

$$\tilde{x}_i = x_i + \sum_j m_{ij}\tilde{x}_j \tag{2.16}$$

for weights $m_{ij} = m_{E,ij} - m_{I,ij}$ where E and I represent excitatory and inhibitory synapses respectively, and real valued inputs x. This use of different synapses for excitatory and inhibitory connections is biologically more plausible, but does not affect the calculations otherwise.

The following constraints on the network connections are deduced:

- $m_{I,ij}$ increases if $\tilde{\mathbf{x}}_\mathbf{j}$ is high and $\tilde{\mathbf{x}}_\mathbf{i}$ is high
- $m_{I,ij}$ decreases if $\tilde{\mathbf{x}}_\mathbf{j}$ is high and $\tilde{\mathbf{x}}_\mathbf{i}$ is low
- $m_{E,ij}$ increases if $\tilde{\mathbf{x}}_\mathbf{j}$ is low and $\tilde{\mathbf{x}}_\mathbf{i}$ is high
- $m_{E,ij}$ decreases if $\tilde{\mathbf{x}}_\mathbf{j}$ is low and $\tilde{\mathbf{x}}_\mathbf{i}$ is low
- in other cases, $m_{I,ij}$ and $m_{E,ij}$ will be stationary,

and used to derive the following linearised model of the network:

$$\frac{dm_{ij}}{dt} = -\alpha \tilde{x}_i \tilde{x}_j, \tag{2.17}$$

where $\tilde{x}_i = x_i + \sum_j m_{ij}\tilde{x}_j$ and α is a positive constant. So, for inputs $\mathbf{x}$ the network produces outputs $\tilde{\mathbf{x}}$, $(\mathbf{x}, \tilde{\mathbf{x}} \in \mathbb{R}^n)$.

Equation 2.17 can be written in matrix notation as:

$$\frac{dM}{dt} = -\alpha \tilde{\mathbf{x}} \tilde{\mathbf{x}}^T, \tag{2.18}$$

where the network output is

$$\tilde{\mathbf{x}} = \mathbf{x} + M\tilde{\mathbf{x}} \tag{2.19}$$

for $M \in \mathbb{R}^{n\times n}$, $M|_{ij} = m_{ij}$, and network transfer function $\Phi \in \mathbb{R}^{n\times n}$ is defined as

$$\Phi\mathbf{x} = (I - M)^{-1}\mathbf{x} = \tilde{\mathbf{x}} \tag{2.20}$$

$$\Rightarrow \frac{d\Phi^{-1}}{dt} = -\Phi^{-1}\frac{d\Phi}{dt}\Phi^{-1} = -\frac{dM}{dt}, \tag{2.21}$$

$$\frac{d\Phi}{dt} = -\alpha\Phi^2 \mathbf{x}\mathbf{x}^T \Phi^2, \tag{2.22}$$

for Φ initially symmetric (and therefore symmetric for all t). Further detail of the derivations of equations 2.21 and 2.22 are given in appendix A. Equation 2.22 forms a matrix Bernoulli equation. Kohonen and Oja (1976) show that stable solutions exist for $\alpha \geq 0$, a result that is extended by reducing the constraints on Φ by Oja (1978).

Kohonen noted the similarity of habituation and novelty filtering, commenting on the functionality of his novelty filter in (Kohonen, 1993) (page 101):

> If this phenomenon [producing a non-zero output only for novel stimuli] were discussed within the context of experimental psychology, it would be termed habituation.

While the analogy is not exact, the essence of the idea is sound – storing inputs and then computing the difference between the current input and the best-matching stored pattern does mean that non-zero output is only seen for novel stimuli, although the mechanism by which the effect is accomplished is certainly not biologically realistic.

This novelty filter detects novelty reliably and has some ability to generalise between similar perceptions, since a perception that is very like another will have very

few bits different. This is a very primitive quantification of the amount of novelty, although it assumes that all the bits are equally important, which may not be valid. However, the network has to be trained in batch, the size of the network is predefined and additional effort is required to enable the novelty filter to forget (see section 2.4.3). Thus, although it is a useful novelty filter, it does not satisfy many of the constraints of the applications that are considered in this thesis.

2.4.2 Implementations of the Novelty Filter

In the original description of the novelty filter (Kohonen and Oja, 1976), the network is a fully-connected feedback system, meaning that the computational costs are huge. Ko et al. (1992) shows that the novelty filter can be implemented as an auto-associative network (i.e., a network where the input vector is reproduced at the outputs, as is shown in figure 2.3) trained using back-propagation of error, see for example Bishop (1995). They derive a non-linear transfer function for the hidden layer and show that the resulting network is equivalent to the filter of Kohonen and Oja.

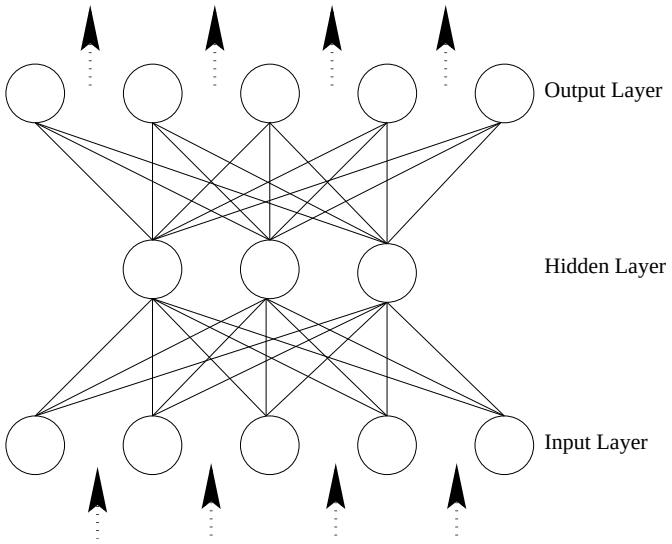


Figure 2.3: An auto-associative neural network.

An alternative approach is given by Matsuoka and Kawamoto (1993), who analyse a linear, single-layer network with Hebbian and anti-Hebbian learning rules, and show that, for different learning parameters, the network can implement Principal Component Analysis (PCA), orthogonal projection, or novelty filtering of the Kohonen and Oja style.

A different version of the novelty filter implemented as a single-layer network is described in Ko and Jacyna (2000). They propose a continuous, auto-associative perceptron. The update rule for the network weights is converted to a first-order random differential equation, and the paper shows that the probability density function of the weights satisfies the Fokker-Planck equation.

2.4.3 Variations on the Novelty Filter

Other authors have provided further understanding of the properties of the novelty filter. Aeyels (1990) investigated the convergence properties of the network for more general initial conditions than were used by Kohonen and Oja (1976). He also proposes a modification of the network to allow for the network to forget inputs over time. The network model is then

$$\frac{dM}{dt} = -\alpha\tilde{\mathbf{x}}\tilde{\mathbf{x}}^T - \beta M, \tag{2.23}$$

with $\beta > 0$ (compare with equation 2.18). The equivalent transfer function (c.f. equation 2.22) is then

$$\frac{d\Phi}{dt} = -\alpha\Phi^2\mathbf{x}\mathbf{x}^T\Phi^2 + \beta(\Phi - \Phi^2). \tag{2.24}$$

The equilibrium solutions of this equation can be analysed as follows. Consider the output of the novelty filter. For a trained input $\mathbf{x}$, the network produces $y(\mathbf{x}) = \mathbf{x}$ at the output. Let a new input be $\mathbf{x}^* = \mathbf{x} + \Delta\mathbf{x}$ for small $\Delta\mathbf{x}$. Then the output of the network is $y(\mathbf{x}^*) = y(\mathbf{x} + \Delta\mathbf{x}) = \mathbf{x}$. The output of the novelty filter is found by subtracting the output of the network from the input, producing

$$\begin{aligned} n(\mathbf{x}^*) &= \mathbf{x}^* - y(\mathbf{x}^*) \\ &= \mathbf{x} + \Delta\mathbf{x} - y(\mathbf{x} + \Delta\mathbf{x}) \\ &= \Delta\mathbf{x}, \end{aligned} \tag{2.25}$$

which is true if $y(\mathbf{x} + \Delta\mathbf{x}) = \mathbf{x}$.

Elsimary (1996) proposes a new training algorithm for the novelty filter (implemented as an auto-associative network) to ensure that it is insensitive to perturbations in the input patterns. A genetic algorithm is used to search the weight space of the neural network in order to minimise the network error. The resulting training algorithm is

compared to back-propagation for a problem of motor fault detection, and is shown to improve the performance on this task. As was discussed in chapter 1, novelty detection is often used on fault detection tasks, in both the medical and engineering fields. However, as each researcher investigates their own specific problem, comparisons between the different novelty filters are very difficult. Some applications of the Kohonen and Oja novelty filter are described in the next section.

2.4.4 Applications

The novelty filter proposed by Kohonen and Oja (1976) has been applied to a number of problems. Kohonen demonstrates the actions of the filter in (Kohonen, 1993) on a toy problem of characters made from 7×5 blocks of pixels. He also gives a more impressive application, showing the output of the filter when it is presented with radiographic images of the brain. The network is presented with 30 images from normal patients (i.e., patients where no medical problems were diagnosed) for training, and then a number of test images, some normal and some abnormal were presented. The filter highlighted those parts of the images that were not seen in the training set, that is the features that were novel. Since these features were not in the training set of normal images they were presumed to be abnormal, and therefore potential brain tumours. Detailed results are not given, so it is not possible to assess how useful the technique would be in practice.

The novelty filter has also been used for radically different applications. For example, Ardizzone et al. (1990) used it to detect motion in a series of images. The network identifies the trajectory of an object by identifying the novelty in the current image with respect to patterns related to previous positions.

Kohonen and Oja's novelty filter has also been used to detect machine breakdowns and for related engineering tasks. For example, Worden (1997) and Worden et al. (2000) compared the novelty filter to kernel density estimation (described in section 2.6.2) and to measuring the Mahalanobis distance (see section 2.8) on structural fault detection. Similar results were found for all three methods, although the Mahalanobis distance method was the fastest, requiring little training and very little calculation for the evaluation. The novelty filter is also used in Streifel et al. (1996) to detect shorted turns in turbine-generator rotors.

In a variation on the theme of breakdowns, Chen et al. (1998) use the novelty filter as one of a number of methods of novelty detection (the others are PCA, described in section 2.8.5, and density estimation (section 2.6.2)) to detect motorway incidents. All

three techniques were found to be successful, although density estimation had a faster response time, but a higher false alarm rate.

2.4.5 Related Approaches

A similar approach to that of the Kohonen and Oja novelty filter is proposed by Japkowicz et al. (1995). In their scheme, which is part of a model of the hippocampus, an auto-associative network is also used. However, rather than highlighting the novel parts of the input, instead the number of bits that are different between the input and output are counted. If this exceeds some threshold then the current input is considered to be novel. Pomerleau (1992) uses a related approach known as Input Reconstruction Reliability Estimation, which consists of computing the following statistic between an input and its reconstruction using an auto-associative network:

$$\rho(I,R) = \frac{\overline{IR} - \overline{I} \cdot \overline{R}}{\sigma_I \sigma_R}, \tag{2.26}$$

where $\overline{I}$ is the mean activation value of the input, $\overline{R}$ is the mean activation value of the reconstruction (the output), σ_R is the standard deviation in the activation of the reconstruction, and $\overline{IR}$ is the mean of the set formed by the unit-wise product of the input and output images. Pomerleau (1992) uses the value of ρ to evaluate how confident the output of a back-propagation network trained on the input is. Qualitatively these ideas are similar to the novelty filter.

An interesting claim was made by Daunicht (1991), who stated that the mechanoreceptors found in muscles and tendons (such as those used to rotate the eyeball) provide a basis for auto-association and novelty detection. The interactions between elastic actuators (i.e., muscles) that act on a rigid body cause this effect, which is shown through the minimisation of the potential energy to find the optimum actuator commands.

The characteristic that links the applications described above is that the class that is wanted is under-represented in the data, so that it is not possible to train a classifier to recognise this class explicitly. That is one reason why novelty detection is useful. Kohonen and Oja's novelty filter exhibits many of the hallmarks of novelty detection. The network is trained off-line on positive examples, and then the deviation between the input and the best-matching prototype vector is used to evaluate the novelty of the input. These features will be seen in many of the other systems described in the next sections. This type of novelty filter cannot be used on-line.

2.5 The Gated Dipole

2.5.1 A Description of the Gated Dipole

An animal gets a steady electric shock until it presses a lever. It therefore learns to press the lever. How can a motor response associated with the absence of negative reinforcement become positively reinforcing? This problem was addressed by Grossberg (1972a,b), who proposed a solution known as the gated dipole, a picture of which is given in figure 2.4.

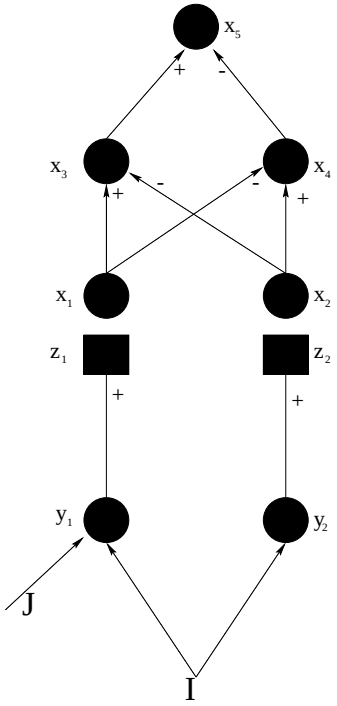


Figure 2.4: A gated dipole. The value of input J affects which side of the dipole has the higher activity and hence whether x_5 receives excitatory (+) or inhibitory (-) input. Squares represent reservoirs of neurotransmitter. Adapted from Levine and Prueitt (1989).

The gated dipole compares current values of a stimulus with recent past values. It has two channels, which compete. Both receive an arousal stimulus, I, but only one (the left one in figure 2.4) receives the sensory stimulus J. In the figure, the squares z_1 and z_2 are synapses, with a transmitter that depletes with activity. While the sensory stimulus J is present, the transmitter at z_1 will be depleted more than that at z_2, but the left-hand column will also receive more stimulation and so x_3 will dominate the output. However, when sensory stimulus J is removed, z_2 will have more of the transmitter remaining, and hence x_4 will dominate at the output. Whichever of x_3 and x_4 dominates

controls the output of x_5 via the connecting synapse, which is excitatory for x_3 and inhibitory for x_4.

This can be seen more clearly in the equations of the dipole, which are given below in equations 2.27-2.35 (g, a_1, a_2 and b are positive constants).

$$\frac{dy_1}{dt} = -gy_1 + I + J \tag{2.27}$$

$$\frac{dy_2}{dt} = -gy_2 + I \tag{2.28}$$

$$\frac{dz_1}{dt} = a_1\left(\frac{1}{2} - z_1\right) - a_2 y_1 z_1 \tag{2.29}$$

$$\frac{dz_2}{dt} = a_1\left(\frac{1}{2} - z_2\right) - a_2 y_2 z_2 \tag{2.30}$$

$$\frac{dx_1}{dt} = -gx_1 + by_1 z_1 \tag{2.31}$$

$$\frac{dx_2}{dt} = -gx_2 + by_2 z_2 \tag{2.32}$$

$$\frac{dx_3}{dt} = -gx_3 + b\left[x_1 - x_2\right]^+, \tag{2.33}$$

where x^+ denotes $\max(x, 0)$.

$$\frac{dx_4}{dt} = -gx_4 + b\left[x_2 - x_1\right]^+ \tag{2.34}$$

$$\frac{dx_5}{dt} = -gx_5 + (1 - x_5)\, x_3 - x_5 x_4. \tag{2.35}$$

On its own, one dipole is not very useful. However, by combining a number of them it is possible to compare stimuli. The combination of gated dipoles is known as a dipole field. For example, Levine and Prueitt (1989, 1992) use a dipole field to model an animal's attention to novelty. They use two dipoles coupled with a reward locus (shown in figure 2.5). The reward locus provides feedback to each of the competing cues. The first dipole receives some familiar input, while the second dipole receives the test input. The output nodes of the two dipoles ($x_{i,5}$ in figure 2.5) compete, with the plastic synapses from u favouring cues that have previously won. The only change

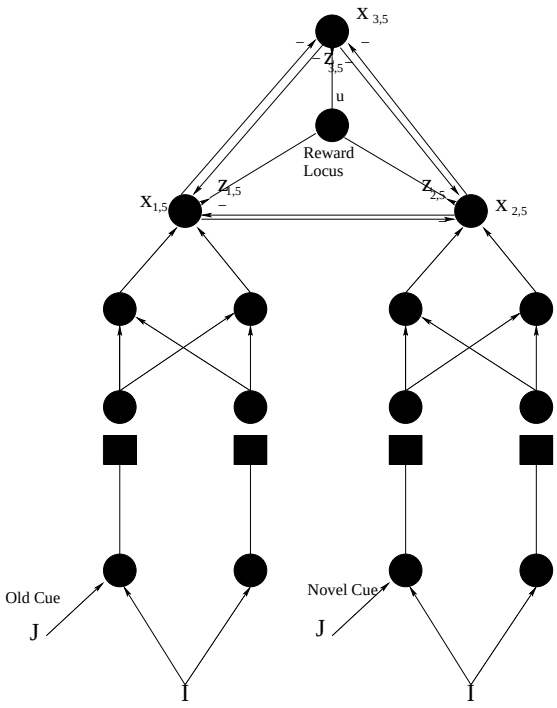


Figure 2.5: A dipole field. The two dipoles receive different stimuli and compete to see which stimuli to respond to. The reward locus, u, biases the output towards previous winners. Adapted from Levine and Prueitt (1989).

required to the equations is that equation 2.35 becomes

$$\frac{dx_{i,5}}{dt} = -gx_{i,5} + \left(1 - x_{i,5}\right)\left(\alpha u z_{i,5} + x_{i,5}\right) - cx_{i,5}\left(x_{i,4} + \sum_{j \neq i} x_{j,5}\right) \tag{2.36}$$

(α, c are positive constants), where the new synapses connecting the reward node and the outputs of the dipoles, ($x_{i,5}$, $z_{i,5}$) are controlled by:

$$\frac{dz_{i,5}}{dt} = f_1 z_{i,5} + f_2 u x_{i,5}. \tag{2.37}$$

In these equations, u is the output of the reward node, which has activity

$$\frac{du}{dt} = gu + r, \tag{2.38}$$

for some constant r. Note also that each of the dipoles has an inhibitory effect on the others.

2.5.2 Applications

Levine and Prueitt (1989) used the dipole field to model some data reported by Pribram (1961) that was described in section 2.2. These experiments demonstrated that monkeys with lesions of the frontal cortex had a preference for novelty. It is these results that Levine et al. wanted to explain through their hypothetical model of the frontal lobes as a dipole field. They found that it is the value of α in equation 2.36 that controls the gain of signals from the reward locus that is critical. They claim that monkeys with frontal damage have lower values of α and hence the output for novel cues is larger, so that they are favoured. Levine et al. claim that their method can be generalised so that further dipoles can be added to represent more cues, which seems reasonable, although they do not provide any evidence of this occurring.

The gated dipole has also been used in a novelty detection system. Öğmen et al. (1992) and Öğmen and Prakash (1997) use the gated dipole in a system that aims to calculate two different types of novelty; spatial novelty and object novelty. Object novelty is used to provide good-bad categorisation using two ART networks. ART is described in section 2.7.2. The first ART network categorises inputs into object types, while the second categorises the output of the first network into good or bad. Novelty detection in this system is performed by the first ART network. If none of the nodes in the ART network match the input then that input is declared to be novel.

It is in the testing for spatial novelty that the gated dipole is used. The area that the system can see is divided into discrete spatial locations and an array of gated dipoles, one for each spatial location, is used, with the outputs of the dipoles feeding into a winner-takes-all network called the attention spanning module. The system is implemented on a robot arm by Prakash and Öğmen (1998), and the system is used to explore an environment and modify its behaviour according to feedback from the environment.

Systems based on the gated dipole cannot generalise to stimuli that do not have a dipole to represent them. Nor can they scale with the size of the dataset. The amount of transmitter in a dipole does not say how novel a particular stimulus is, but rather how often any novel stimulus has been seen, compared to the non-novel stimulus that the dipole is tuned to recognise.

2.6 Validating the Output of a Neural Network

2.6.1 Introduction

One of the principal uses of artificial neural networks is classification, clustering data into two or more classes. Neural networks can be trained in two ways, *supervised* learning, where each training input vector is paired with a target vector or desired output, and *unsupervised* learning, where the network self-organises to extract patterns from the data without any target information.

This section concentrates on supervised neural networks. Typical examples are the perceptron (Rosenblatt, 1962), and related multi-layer perceptron (MLP) (McClelland et al., 1986) and the radial basis function (RBF) network (Moody and Darken, 1989). Overviews of these and other networks can be found in any standard text, such as Bishop (1995) or Haykin (1999). These networks adapt the connection weights between layers of neurons in order to approximate a mapping function that models the training data. In the trained network every input produces an output. For classification tasks this is usually an identifier for the best-matching class. However, there is no guarantee that this best-matching class is a good match, only that it is better than the other classes for the set of training data that was used. This is what makes a novelty filter useful for these types of problems, recognising inputs that were not covered by the training data and that therefore the classifier cannot categorise reliably. Another approach that can be used is to add error bars to the output, so that the confidence in each prediction can be seen (Weigend and Nix, 1994; Penny and Roberts, 1997).

2.6.2 Kernel Density Estimation

Assuming that the network has been trained well, the main reason why the predictions of the network could be poor is that the dataset that was used to train the network is not representative of the whole set of potential inputs. There are two possible reasons for this – there are only a few examples of an important class, or the classification set is incomplete.

One interpretation of this is that there is a strong relationship between the reliability of the output of the network and the degree of novelty in the input data. This approach has been taken by Bishop (1994), who evaluated the sum-of-squares error function of the network:

$$\begin{aligned} E &= \sum_{j=1}^{m} \int \left[y_j(\mathbf{x},\mathbf{w}) - t_j \right]^2 p(\mathbf{x},t_j) d\mathbf{x} dt_j \\ &= \sum_{j=1}^{m} \int \left[y_j(\mathbf{x},\mathbf{w}) - \langle t_j|\mathbf{x}\rangle \right]^2 p(\mathbf{x}) d\mathbf{x} + \\ &\quad \sum_{j=1}^{m} \int \left\{ \langle t_j^2|\mathbf{x}\rangle - \langle t_j|\mathbf{x}\rangle^2 \right\} p(\mathbf{x}) d\mathbf{x}, \end{aligned} \tag{2.39}$$

where $p(\mathbf{x},t_j)$ is the joint probability density function for the data, $j = 1,\ldots,m$ are the output units, $\mathbf{w}$ the weights, $\mathbf{x}$ is the input to the network, t_j the associated target for unit j, and y_j the actual output of unit j. The conditional averages of the target data in equation 2.39 are given by:

$$\langle t_j|\mathbf{x}\rangle \equiv \int t_j p(t_j|\mathbf{x}) dt_j, \tag{2.40}$$

$$\langle t_j^2|\mathbf{x}\rangle \equiv \int t_j^2 p(t_j|\mathbf{x}) dt_j. \tag{2.41}$$

Only the first of the two parts of equation 2.39 is a function of the weights $\mathbf{w}$, so if the network is sufficiently flexible (i.e., the network has enough hidden units), the minimum error E is gained when

$$y_i(\mathbf{x},\mathbf{w}) = \langle t_j|\mathbf{x}\rangle, \tag{2.42}$$

which is the regression of the target vector conditioned on the input. The first term of equation 2.39 is weighted by the density $p(\mathbf{x})$ and so the approximation is most accurate where $p(\mathbf{x})$ is large (i.e., the data is dense).

In general we do not know very much about the density $p(\mathbf{x})$. However, we can generate an estimate $\hat{p}(\mathbf{x})$ from the training data and use this estimate to get a quantitative measure of the degree of novelty for each new input vector. This could be used to put error bars on the outputs of the network, or to reject data where the estimate $\hat{p}(\mathbf{x}) < \rho$ for some threshold ρ, effectively generating a new class of 'novel' data. The distribution of the novel data is generally completely unknown. It can be estimated most simply as being constant over a large region of the input space and zero outside this region to make it possible to normalise the density function, as is shown in figure 2.6.

This approach was used by Bishop (1994) for data collected from oil pipelines.

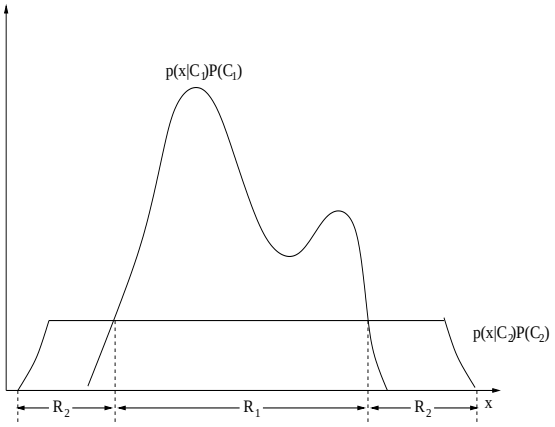


Figure 2.6: Novelty detection in the Bayesian formalism. The training data is used to estimate $p(\mathbf{x}|C_1)P(C_1)$ using $\hat{p}(\mathbf{x})$, with novel data (class C_2) having a distribution that is assumed constant over some large region. Vectors that are in the regions labelled R_2 are considered to be novel. Adapted from Bishop (1994).

The training set is first examined by hand to ensure that it has no examples of the novel class, the class that should be detected. A Parzen window estimator (Silverman, 1986) with one Gaussian kernel function for each input is then used to model the training set, so that

$$\hat{p}(\mathbf{x}) = \frac{1}{n(2\pi)^{d/2}\sigma^d} \sum_{q=1}^{n} \exp\left\{ -\frac{|\mathbf{x}-\mathbf{x}^q|^2}{2\sigma^2} \right\} p(q), \tag{2.43}$$

where $\mathbf{x}^q$ is a data point in the training set, σ is the smoothing function and d is the dimensionality of the input space. Any point where the likelihood $\hat{p}(\mathbf{x})$ is below some threshold is considered to be novel.

Tarassenko et al. (1995) followed the same approach, but used a local representation of the data by using the k-means algorithm (Duda and Hart, 1973) to partition the data and then used a different value of the smoothing parameter σ in equation 2.43 for each data cluster, so that local properties of the data could be taken into account. This approach was tested with reasonable success on various datasets, mostly of rather small size, including mammograms (Tarassenko et al., 1995) and jet engine data (Nairac et al., 1997, 1999).

A similar approach is taken by Roberts and Tarassenko (1994). A Gaussian mixture model, a method of performing semi-parametric estimation of the probability density function (Trávén, 1991) is used to learn a model of the 'normal' data from the training set. The number of mixtures is not defined in advance, with new mixtures being added

if the mixture that best represents the data is further from the input than some threshold. In testing, any input that would require a new mixture to be generated is considered to be novel. This is the first technique that has been capable of continuous learning, as the size of the model grows as more data is provided, but the filter does not quantify the novelty.

This work is applied to a large number of datasets taken from medical problems, such as EEG data for sleep disturbances (Roberts and Tarassenko, 1994), epilepsy (Roberts, 1998) and MRI images of brain tumours (Roberts, 2000). However, these approaches are not useful for the types of applications that are considered in this thesis. Most of the networks cannot be used on-line, nor can they forget about perceptions that are seen only infrequently.

In Parra et al. (1996), the problem of density estimation is considered through minimum mutual information, which is used to factorise a joint probability distribution. A Gaussian upper bound is put on the distribution, and this is used to estimate the density of the probability functions.

Instead of estimating the output density, Tax and Duin (1998) measure the instability of a set of simple classifiers. A number of classifiers are trained on bootstrap samples the same size as the original training set. For new data, the output of all the classifiers is considered. If the data is novel, then the variation in responses from the different classifiers will be large. This approach is applied to three different types of network – a Parzen window estimator, a mixture of Gaussians and a nearest neighbour method. A similar method is employed in Roberts et al. (1996), where a committee of networks (Perrone and Cooper, 1993) initialised at random is used. These approaches have the benefit that if the dataset changes, or some aspects of the 'normal' set are missed in training, then further networks can be trained and the network can therefore still be used. However, there is still no quantification of the amount of novelty in a perception, nor the ability to forget about inputs that are seen only infrequently.

2.6.3 Extending the Training Set

In Roberts et al. (1994) a method of extending the training set is considered, so that the neural network can be trained to recognise data from regions that were not included in the original set. Suppose that the training set for some problem spans the region $\mathcal{R} \subset \mathbb{R}^n$. Then

$$p(\text{class}_1|\mathbf{x}) = \frac{p(\mathbf{x}|\text{class}_1)p(\text{class}_1)}{p(\mathbf{x}|\mathcal{R})}, \tag{2.44}$$

where

$$p(\mathbf{x}|\mathcal{R}) = p(\mathbf{x}|\text{class}_1)p(\text{class}_1) + p(\mathbf{x}|\text{class}_2)p(\text{class}_2). \tag{2.45}$$

Note that this implies that $\text{class}_1 \cup \text{class}_2 = \mathcal{R}$. Using Bayes' theorem,

$$\begin{aligned} p(\mathcal{R}|\mathbf{x}) &= \frac{p(\mathbf{x}|\mathcal{R})p(\mathcal{R})}{p(\mathbf{x}|\mathcal{R})p(\mathcal{R}) + p(\mathbf{x}|\mathcal{R}')p(\mathcal{R}')}, \\ &= 1 - p(\mathcal{R}'|\mathbf{x}) \end{aligned} \tag{2.46}$$

where $\mathcal{R}'$ is the missing class, which is separate from $\mathcal{R}$. The aim is then to generate data in $\mathcal{R}'$. They do this by generating data at random and removing any members of it that are in $\mathcal{R}$.

2.7 Novelty Detection Based on Self-Organising Networks

The previous section discussed validation techniques for supervised learning algorithms. This section describes methods that have been used to detect novelty using unsupervised learning algorithms. For novelty detection these algorithms are partially supervised in that, although explicit target vectors are not given, the training set is tailored to ensure that there are no examples of inputs that the network should find to be novel.

2.7.1 Novelty Detection with the Self-Organising Map

A number of authors have used the Self-Organising Map (SOM) of Kohonen (1982, 1993) for novelty detection tasks. Further details of the SOM are given in section 3.3.

The approach taken by Taylor and MacIntyre (1998) is the simplest. They use a set of data that is known to contain no novelties (i.e., no examples of undesirable features) to train the SOM to recognise a model of normality. Once the SOM is trained, new data is presented to the network and the distance between the node that best matches the new data and any of the nodes that fired when data known to be normal were

presented is calculated. If this distance exceeds some threshold then the data used as input is declared to be novel. The aim of the work is to produce machine-monitoring equipment. The system cannot operate on-line, since the nodes that represent normal data need to be found in a training phase, nor is there any indication of how novel a perception is.

A similar kind of idea is used by Höglund et al. (2000) to detect attacks on computers. Their approach, which they term, 'anomaly detection', is based on the belief that if the behaviour of a process is consistent it will be concentrated on only a few regions of the feature space. In their case, the processes that they model are the users of a UNIX network, and they are attempting to discover intruders. They compute an anomaly P value by computing the distance to the best-matching unit (BMU) for the current input and then counting the number of BMU distances for the training inputs that are bigger than this distance, and dividing this number by the number of training inputs. This approach is not suited to on-line usage, nor are the other objectives that were described at the beginning of the chapter satisfied.

The method proposed by Ypma and Duin (1997) is very different. They investigate how to tell whether two datasets come from the same distribution, as a way of measuring if the second dataset is novel with respect to the first. A number of suitable techniques have been presented in the literature for measuring the quality of the mapping from input space to feature space. These measures are reviewed in more detail in section 5.5.2. Ypma and Duin (1997) use a different measure, a normalised measure of the distance between map units,

$$
\begin{aligned}
d(x) \quad = \quad & \frac{\|x - m_{p1(x)}\|}{\frac{1}{Q}\sum_{q \in q(x)} \|m_{p1(x)} - m_q\|} + \\
& \left\{ \sum_{n=2}^{k} \alpha_n \left| \min_i \sum_{j=0}^{K_{p_k(x),i}-1} \|m_{I_i(j)} - m_{I_i(j+1)}\| - \right. \right. \\
& \left. \left. \min_l \sum_{j=0}^{K_{q_k(x),l}-1} \|m_{J_l(j)} - m_{J_l(j+1)}\| \right\}, \qquad (2.47)
\end{aligned}
$$

where m_s is the reference vector of unit s, $I_i(k)$ is the index of the kth unit on the shortest path on the map grid from $I_i(0) = p_1(x)$ (the best match) to $I_{i(K_{p2(x)_1})} = p_2(x)$, the next best match. Similarly, $J_l(j)$ denotes the index of the jth unit on the shortest path l along the map grid from unit $J_l(0) = p_1(x) = q(x)$ to a k-nearest map unit. Q

is the cardinality of set q. They also measure the mean-squared error of the mapping. Two problems are investigated using this approach. The first is a variation on the common problem of mechanical fault detection, while the second is the problem of detecting leaks in pipelines. The approach is designed to compare two datasets that are known before the process begins, so it is not suitable for on-line use, although it does compute a potentially useful comparison for the amount of novelty in the second dataset compared to the first one.

A way of using the SOM to perform multivariate outlier detection is considered by Muñoz and Muruzábal (1998). They consider two separate techniques:

- graphical methods, plotting the distance between nodes in the map
- measuring the quantisation errors

The first technique is useful for detecting outlying neurons, while the second detects outlying datapoints that project onto neurons that fit into the map well, a feature that may be quite common for high dimensional datasets. The quantisation error is measured by

$$e_k = d\left(\mathbf{x_k}, \mathbf{w}(i^*, j^*)\right) \tag{2.48}$$

where (i^*, j^*) are the coordinates of the winning node and $d(\cdot)$ is the Euclidean distance. They use box plots to decide when an outlier is detected. These approaches are designed to process the results of the mapping for human analysis, so they are not suitable for use on the robot.

Two methods are used in order to visualise the data for the first technique. One is the Sammon mapping, E, an error calculation that measures the distance between pairs of points, initially in their original space (d_{ij} is the Euclidean distance in that space) and then between the points that represent them in the map space (d'_{ij} is the Euclidean distance in this space) (Sammon, 1969):

$$E = \frac{\sum_j \sum_{i<j} (d_{ij} - d'_{ij})^2}{d_{ij} \sum_j \sum_{i<j} d_{ij}}. \tag{2.49}$$

This is usually solved by a gradient-descent minimisation technique. The other method is to calculate a matrix of the median-interneuron-distances (MID) over the network.

The problems of using the SOM for on-line learning, together with scalability issues when the size of the input dataset is not known in advance will be discussed

extensively in section 3.5. In general, the techniques described in this section are not suitable for on-line learning, do not quantify the amount of novelty in a perception in any useful way and do not allow for forgetting.

2.7.2 The Adaptive Resonance Theory Network

The Adaptive Resonance Theory (ART) networks (Carpenter and Grossberg, 1988) attempt to create stable memories from inputs using match-based learning. The basic ART networks perform unsupervised learning, ART1 on binary inputs and ART2 on continuous inputs. When an input is given to the network, the network searches through the categories currently stored for a match. If a match is found then this category is used to represent the input, if no category is found (so that the strength of response from each of the categories is low) then a new category is added. In itself, this ability to add new nodes whenever none of the current categories represents the data is a form of novelty detection.

This approach has been used by a number of researchers, for example, Öğmen et al. (1992) use an ART network to detect novel objects, as was described in section 2.5.2. Caudell and Newman (1993) use an ART network to process a time series, monitoring the creation and usage of the ART categories to see when the time series is stable and when changes occur. A different approach is taken by Lozo (1996), who proposes a match/mismatch detection circuit in a selective attention ART model. The ART networks are capable of on-line learning, something that many of the novelty filters have not been able to do. However, the systems that have been based on it have not been capable of forgetting, nor of quantifying the amount of novelty in a perception.

2.7.3 The Reduced Coulomb Energy Network

The Reduced Coulomb Energy (RCE) network (Reilly et al., 1982) was originally designed for supervised learning of pattern categories. A simplified RCE network for unsupervised learning was introduced by Kurz (1996), who used it to categorise robot sonar readings for navigation. The appearance of the network is shown in figure 2.7. Input vectors are compared to each of the prototype vectors in the map space, usually using the inner product. The pattern that is closest in the Euclidean sense to the input is selected as the best match, unless the distance is greater than some threshold (the sphere of influence), in which case the input vector is added into the network as a new prototype. Once added to the network, the prototype vectors do not change.

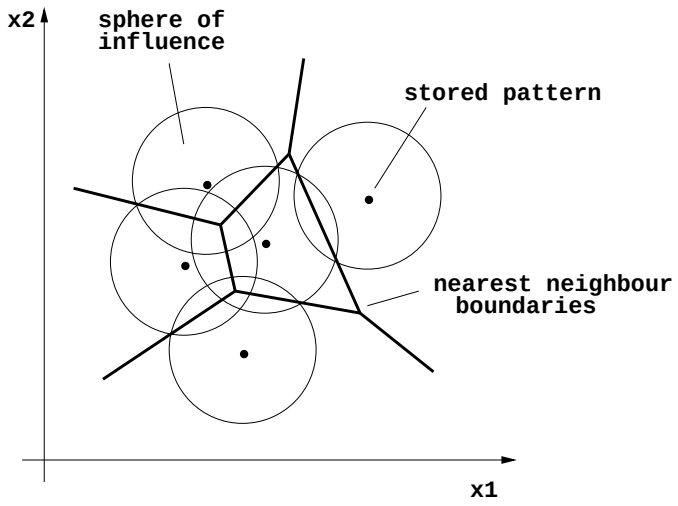


Figure 2.7: The RCE classifier in two dimensions. A new prototype is created if the input pattern is outside the sphere of influence of all the current stored patterns. Adapted from Kurz (1996).

While the RCE network does operate on-line, the extent to which it generalises is determined only by the sphere of influence, since there is no attempt at self-organisation of neighbourhoods.

2.8 Outlier Detection

2.8.1 Introduction

The problem of statistical outlier detection is closely related to that of novelty detection. No precise definition of an outlier seems to have been produced, but most authors agree that outliers are observations that are inconsistent with, or lie a long way from, the remainder of the data. Outlier detection aims to handle these rogue observations in a set of data, which can have a large effect on analysis of the data (datapoints that have a large effect are known as influential observations). The problem is that it is not possible to find an outlier in multivariate data by examining the variables one at a time. Statistical techniques that do this are certainly not capable of on-line usage, as they require the entire input set to be analysed. This makes them unsuitable for use on the problems considered in this thesis, especially as they do not quantify the amount of novelty either. However, they are important techniques for off-line novelty detection.

How important outlier detection is to statistical methods can be seen in figure 2.8 – an outlying datapoint can completely change the least-squares regression line of the

data. Generally, statistical methods are concerned with ignoring unrepresentative data, rather than explicitly recognising those points. Techniques that avoid many of the problems of outliers are known as *robust statistics* (Huber, 1981). There are also sets of tests for deciding whether predictions from particular distributions have been affected by outliers, see for example Barnett and Lewis (1994). The appearance of some outliers in two dimensions are shown in figure 2.9. The next four subsections describe statistical techniques that are used to detect and deal with outlying datapoints.

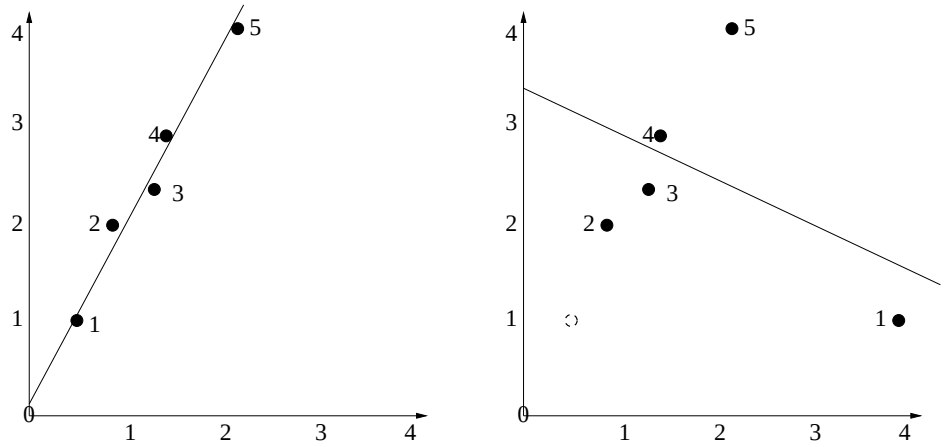


Figure 2.8: A demonstration of why outlier detection is important in statistics. The five points comprise the data and the line is the least-squares regression line. In the graph on the right, point 1 has been misread. It can be seen that this completely changes the least-squares regression line.

2.8.2 Outlier Diagnostics

The residual of a point is $r_i = y_i - \hat{y}_i$, that is, the difference between the actual point (y) and the prediction of a point ($\hat{y}$). The linear model of statistical regression for data $\mathbf{X}$ is:

$$\mathbf{y} = \mathbf{X}\theta + \mathbf{e}, \tag{2.50}$$

where θ is the vector of (unknown) parameters and $\mathbf{e}$ is the vector of errors. The hat matrix, $\mathbf{H}$ (so called because $\mathbf{Hy} = \hat{\mathbf{y}}$) is defined as:

$$\mathbf{H} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T. \tag{2.51}$$

Then

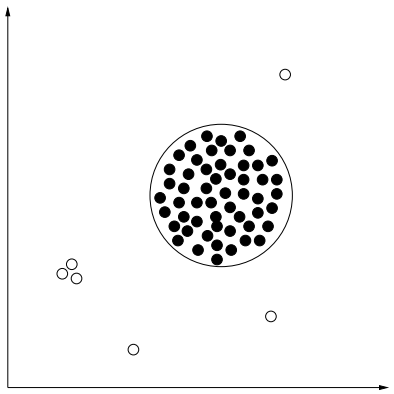


Figure 2.9: The principle of outlier detection. Empty circles are outliers to the dataset comprised of the black circles. The circle surrounding the datapoints demonstrates a potential threshold, beyond which points are outliers.

$$\text{cov}(\hat{\mathbf{y}}) = \sigma^2\mathbf{H}, \tag{2.52}$$

$$\text{cov}(\hat{\mathbf{r}}) = \sigma^2(\mathbf{I}-\mathbf{H}), \tag{2.53}$$

where $\mathbf{r}$ is the vector of residuals and σ^2 the variance. So, each element h_{ij} of $\mathbf{H}$ can be interpreted as the effect exerted by the jth observation on $\hat{y}_i$, and $h_{ii} = \frac{\partial \hat{y}_i}{\partial y_i}$, the effect that an observation has on its own prediction. The average of this is p/n, where $p = \sum_{i=1}^{n} h_i$, and in general points are considered to be outliers if $h_{ii} > 2p/n$ (Rousseeuw and Leroy, 1987).

It is interesting to note that $\mathbf{H}$ is the pseudoinverse if $(\mathbf{X}^T\mathbf{X})^{-1}$ exists. Therefore, the hat matrix method is related to the approach of Kohonen and Oja (section 2.4), which can be considered as an implementation of the hat matrix.

The values along the diagonal of the hat matrix can be used to scale the residuals. Three methods are shown below:

standardised $\frac{r_i}{s}$, where $s^2 = \frac{1}{n-p}\sum_{j=1}^{n} r_j^2$

studentised $\frac{r_i}{s\sqrt{1-h_{ii}}}$

jackknifed $\frac{r_i}{s_i\sqrt{1-h_{ii}}}$, ($s_i$ = s without the ith case).

Another method that is used is the Mahalanobis distance of each point:

$$D^2 = (\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu), \tag{2.54}$$

where Σ is the covariance matrix and μ the mean. The Mahalanobis distance is a useful measure of the similarity between two sets of values.

2.8.3 Recognising that the Generating Distribution has Changed

One question that outlier detection aims to answer can be phrased:

Given n independent random variables from a common, but unknown, distribution μ, does a new input X belong to the support of μ?

The support, or kernel, of a set is a binary valued function that is positive in those areas of the input space where there is data, and negative elsewhere. The standard approach to the problem of outlier detection (Hájek and Šidák, 1967) is to take further independent measurements of the new distribution, which is assumed to have a common probability measure ν, and to test if $\mu \neq \nu$ i.e., to see if the support of $\nu \in S$, where S is the support of μ. The problem then is how to estimate the support S from the independent samples $X_1, \ldots X_n$.

The obvious approach (Devroye and Wise, 1980) is to estimate S_n as

$$S_n = \bigcup_{i=1}^{n} A(X_i, \rho_n), \tag{2.55}$$

where $A(x,a)$ is the closed sphere centred on x with radius a and ρ_n is a number depending only upon n.

Then the probability of making an error on datapoint X, given the data so far, is

$$\begin{aligned} L_n &= P(X \in S | X_1, \ldots X_n) \\ &= \nu(S). \end{aligned} \tag{2.56}$$

The detection procedure is said to be consistent if $L_n \to 0$ in probability, and strongly consistent if $L_n \to 0$ with probability one.

2.8.4 Extreme Value Theory

In Roberts (1998), extreme value theory (EVT) (Gumbel, 1958) is used to approach the problem of detecting outliers in data. The approach investigates the distributions of data that have abnormally high or low values in the tails of the distribution that generates the data. Although this approach has not been used for on-line outlier detection,

it could be by using on-line estimates of the mean and standard deviation of the data.

Let $\mathcal{Z}_m = \{z_1, z_2, \ldots, z_M\}$ be a set of m independent and identically distributed random variables $z_i \in \mathbb{R}$ drawn from some arbitrary distribution $\mathcal{D}$, and let $x_m = \max(\mathcal{Z}_m)$. Then, when observing other samples, the probability of observing an extremum $x \geq x_m$ may be given by the cumulative distribution function

$$p(x_m|\mu_m, \sigma_m, \gamma) = \exp\left\{-\left[1 + \frac{\gamma(x_m - \mu_m)}{\sigma_m}\right]^{1/\gamma}\right\}, \tag{2.57}$$

where $\gamma \in \mathbb{R}$ is the shape parameter. In the limit as $\gamma \to 0$, this leads to the Gumbel distribution

$$P(x_m \leq x|\mu_m, \sigma_m) = \exp\{-\exp(-y_m)\}, \tag{2.58}$$

where μ_m and σ_m depend on the number of observations m, and y_m is the reduced variate

$$y_m = \frac{(x_m - \mu_m)}{\sigma_m}. \tag{2.59}$$

2.8.5 Principal Components Analysis

Principal Components Analysis (PCA) is a standard statistical technique for extracting structure from a dataset by performing an orthogonal basis transformation to the coordinate system in which the data is described. This can reduce the number of features needed for effective data representation.

PCA can be used for detecting outliers that are in some sense orthogonal to the general distribution of the data (Jolliffe, 1986). By looking at the first few principal components, any datapoints that inflate the variances and covariances to a large extent can be found. However, by looking at the last few principal components, features that are not apparent with respect to the original variables (i.e., outliers) can be seen. There are a number of test statistics that have been described to find these points. Two examples are a measure of the sum of squares of the values of the last few principal components and a version that is weighted by the variance in each principal component. Further details can be found in Jolliffe (1986).

2.9 Other Models of Novelty Detection

2.9.1 Hidden Markov Models

A Hidden Markov Model (HMM) is comprised of a number of states, each with an associated probability distribution, together with the probability of moving between pairs of states (transition probabilities) at each time. The actual state at any time is not visible to the observer (hence the *hidden* part of the name), instead an outcome or observation generated according to the probability distribution of the current state is observed. Further details can be found in Rabiner and Juang (1986). A picture of an HMM is shown in figure 2.10. HMMs have been found to be very useful in a number of different applications, in particular speech processing (Rabiner, 1989).

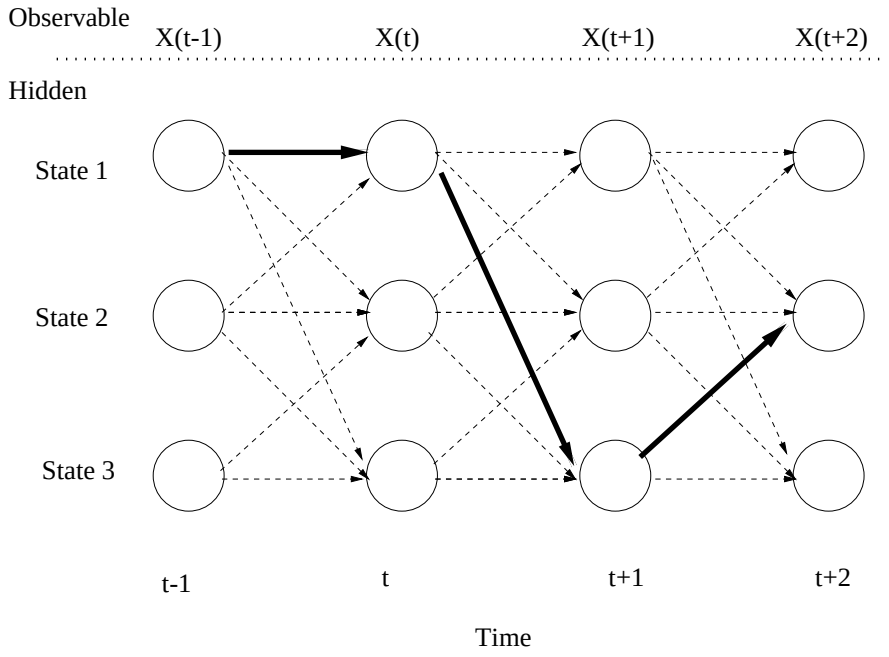


Figure 2.10: An example of a Hidden Markov Model. Adapted from Smyth (1994b).

As the standard HMM has a predetermined number of states, it is not a useful technique for novelty detection. This has been addressed by Smyth (1994a) to investigate the problem of fault detection in dynamic systems using HMMs. The faults that can occur need to be identified in advance, and states generated for each model. The model also assumes that faults do not happen simultaneously, as this would cause problems with faults not being recognised. The technique is related to the density estimation method that was described in section 2.6.2, but with the inputs being sequences.

The modification proposed by Smyth (1994b) is to allow extra states to be added

while the HMM is being used. Let $w_{\{1,\ldots,m\}}$ be the event that the true system is in one of the states $w_1,\ldots,w_m$, and $p(w_{\{1,\ldots,m\}}|y)$ be the posterior probability that the data is from a known state, given the observation y. Then

$$p(w_i|y) = p_d(w_i|y, w_{\{1,\ldots,m\}})\, p(w_{\{1,\ldots,m\}}|y),\ 1 \leq i \leq m, \tag{2.60}$$

where $p_d(\cdot)$ is the posterior probability of being in state i, generated from some discriminative model, and the second part of the product can be calculated using Bayes' rule and the fact that

$$p(w_{m+1}|y) = 1 - p(w_{\{1,\ldots,m\}}|y). \tag{2.61}$$

The probability of getting a novel state, i.e., a machine fault in the example used by Smyth, can be estimated from the mean time between failures. As further states can be added to the HMM on-line, this approach could be used on the robot. However, it is not possible to implement forgetting in the system, nor to quantify the amount of novelty usefully.

2.9.2 Support Vector Machines

The Support Vector Machine (SVM) is a statistical machine learning technique that performs linear learning by mapping the data into a high dimensional feature space (Vapnik, 1995). The SVM operates by selecting the optimal hyperplane that maximises the minimum distance to the training points that are closest to the hyperplane. This is done in some high dimensional feature space into which the input vectors are mapped using a non-linear mapping called the kernel. For a more detailed description of SVMs, see Burges (1998) or Cristianini and Shawe-Taylor (2000).

SVMs can also be used to describe a dataset (Tax et al., 1999). The aim is to model the 'support' of a data distribution, i.e., a binary valued function that is positive in those parts of the input space where the data lies, and negative otherwise. This means that the SVM can then detect inputs that were not in the training set – novel inputs.

This generates a decision function

$$\text{sign}\,(\mathbf{w}\cdot\Phi(\mathbf{z}) + b) = \text{sign}\left(\sum_j \alpha_j K(\mathbf{x_j}, \mathbf{z}) + b\right), \tag{2.62}$$

where K is the kernel function (see equation 2.68), $\Phi(\cdot)$ the mapping into feature space, b the bias, $\mathbf{z}$ the test point, $\mathbf{x_i}$ an element of the training set and $\mathbf{w}$ is the vector

$$\mathbf{w} = \sum_j \alpha_j \Phi(\mathbf{x_j}). \tag{2.63}$$

A hyperspherical boundary with minimal volume is put around the dataset. This is done by minimising an error function containing the volume of the sphere using Lagrangian multipliers L (R is the radius of the hypersphere):

$$L(R, \mathbf{a}, \lambda) = R^2 - \sum_i \lambda \left[R^2 - (\mathbf{x_i} - \mathbf{a})^2\right], \ \lambda \geq 0, \tag{2.64}$$

which gives a value of

$$L = \sum_i \alpha_i (\mathbf{x_i} \cdot \mathbf{x_i}) - \sum_{i,j} \alpha_i \alpha_j (\mathbf{x_i} \cdot \mathbf{x_j}), \tag{2.65}$$

with $\alpha_i \geq 0, \sum_i \alpha_i = 1$.

An object $\mathbf{z}$ is considered to be normal if it lies within the boundary of the sphere, i.e.,

$$\begin{aligned} (\mathbf{z} - \mathbf{a})(\mathbf{z} - \mathbf{a})^T &= \left(\mathbf{z} - \sum_i \alpha_i \mathbf{x_i}\right)\left(\mathbf{z} - \sum_i \alpha_i \mathbf{x_i}\right) \\ &= (\mathbf{z} \cdot \mathbf{z}) - 2\sum_i \alpha_i (\mathbf{z} \cdot \mathbf{x_i}) + \sum_{i,j} \alpha_i \alpha_j (\mathbf{x_i} \cdot \mathbf{x_j}) \\ &\leq R^2. \end{aligned} \tag{2.66}$$

Further flexibility in the boundary can be gained by replacing the inner products in the above equations by kernel functions $K(x, y)$. Campbell and Bennett (2000) also point out that using slack variables allows certain datapoints to be excluded from the hypersphere, so that the task becomes to minimise the volume of the hypersphere and the number of datapoints outside, i.e.,

$$\min \left[R^2 + \lambda \sum_i \xi_i\right] \text{ such that } (\mathbf{x_i} - \mathbf{a}) \cdot (\mathbf{x_i} - \mathbf{a}) \leq R^2 + \xi_i, \ \xi_i \geq 0. \tag{2.67}$$

In general, this requires quadratic programming to generate a solution. However, Campbell and Bennett (2000) follow Schölkopf et al. (1999) in an alternative approach that significantly reduces the amount of computation required. If the kernel

mapping is restricted to be the RBF kernel

$$K(\mathbf{x_i}, \mathbf{x_j}) = e^{-\|\mathbf{x_i}-\mathbf{x_j}\|^2/2\sigma^2} \tag{2.68}$$

then the data lies on the surface of a hypersphere in the feature space (since $K(\mathbf{x}, \mathbf{x}) = 1$). The hyperplane that separates the surface where the data lies from the region containing no data is generated by constructing the hyperplane that separates all of the datapoints from the origin, but is maximally distant from the origin. Hence, the dual problem is to find

$$\min W(\alpha) = \frac{1}{2}\sum_{i,j=1}^{m} \alpha_i \alpha_j K(\mathbf{x_i}, \mathbf{x_j}) \text{ such that } 0 \leq \alpha_i \leq C, \ \sum_{i=1}^{m} \alpha_i = 1. \tag{2.69}$$

This can be implemented using linear programming techniques. In Campbell and Bennett (2000), the performance of their algorithm on a number of datasets that are available in the public domain is given. These datasets are described in section 6.4 and the results compared to the novelty filter created in this work. SVM methods are only suitable for supervised learning in batch, and therefore are not applicable to the on-line inspection tasks.

2.9.3 The Hopfield Network

The Hopfield network (Hopfield, 1982) is a set of non-linear neurons with binary states, with synaptic connections between every pair of neurons, so that a feedback system is formed. A Hebbian learning rule (Hebb, 1949) is used, so that synapses are strengthened when two neurons fire simultaneously. The network acts as a context addressable memory, storing trained patterns in stable states so that any input that is presented to the network settles into one of the stable states.

The Hopfield network was used by Jagota (1991) to store dictionary words, with the network being used for error correction of text, i.e., as a spelling checker. Jagota (1991) shows experimentally that, although the Hopfield network may occasionally miss an error in the input, even with large dictionaries the network never misclassifies a stored word as a novelty.

One of the key points is that the Hopfield net is not used to retrieve the words, at which task the network does not do well, but merely to state whether or not a particular input is a word or not. This is done by testing whether the state of the network after a

word is presented is stable or not. This work has been extended by Bogacz et al. (1999, 2000) in their FamE (Familiarity based on energy) model. They measure the energy of the network,

$$E(\overline{x}) = -\frac{1}{2}\sum_{i=1}^{N} x_i \sum_{j=1}^{N} x_j w_{ij}, \tag{2.70}$$

where x_i is the value of neuron i, w_{ij} is the strength of the weight vector between neurons i and j, and the sums are over all the N neurons in the network. They suggest that the value of this energy function is lower for stored patterns (of which there are P), being of the order of

$$-N + O(0, \sqrt{2P}) \tag{2.71}$$

for stored patterns and

$$O(0, \sqrt{2P}) \tag{2.72}$$

for novel patterns that are not correlated with any previous inputs.

They therefore define a threshold halfway between these two values ($-N/2$), and assign any input where the energy of the network is above this threshold to be novel. It is shown that the Hopfield network stores significantly more patterns when the network is not required to reproduce the patterns, merely to classify them. This difference is put at $0.023N^2$ for retrieval, as opposed to $0.145N$ for storage in Bogacz et al. (1999). It is claimed in Bogacz et al. (2000) that the actions of this network are similar to the activities of the perirhinal cortex for familiarity discrimination in animals. This is based on the discovery of so-called novelty neurons (Brown and Xiang, 1998) that respond maximally to stimuli that have not been seen before, as was described in section 2.2.

The FamE model is suitable for on-line operation, and has been applied to a robot inspection task (Crook and Hayes, 2001) in work that parallels many of the experiments reported here. A mobile robot travelled parallel to a wall, viewing an 'image gallery' of orange rectangles mounted on the wall using a CCD camera. The image was reduced to a 48×48 binary image, with a 1 at any point signifying the presence of orange, and a 0 the absence of it. This input pattern was presented to a $48 \times 48 (= 2,304)$ node Hopfield network and the energy of the network computed, as described above. If the energy was above the threshold then the current input was set to be novel, otherwise it was considered normal. Experimental observations showed that for a pattern

to be found to be normal it had to differ from previously seen patterns by at least 15%. This method is not capable of quantifying novelty as it is currently used, although it is possible that the energy could be used in this way. However, it is not possible to forget about perceptions that are seen only infrequently.

2.9.4 Time to Convergence

Ho and Rouat (1997, 1998) propose a novelty detection method that is based on an integrate-and-fire neuronal model. The usual approach is taken, in that a training set of patterns that are known not to be novel are used to train the network and then test patterns are evaluated with respect to this training set. However, for this technique, it is the time that the network takes to converge when an input is presented that suggests whether or not an input pattern is novel or not.

The network architecture is based on a very simple model of layer IV of the cortex. It consists of a two-dimensional sheet of excitatory and inhibitory neurons with recurrent connections, positioned according to a pseudo-random distribution. Neurons have local connections in a square neighbourhood, with training through Hebbian learning. The state of each neuron is given by

$$S_i(t) = \begin{cases} 0 & \text{if } (t - t_{\text{spike}}) < \rho; \\ \mathcal{H}(U_i(t) - \theta) & \text{otherwise,} \end{cases} \tag{2.73}$$

where $\mathcal{H}(\cdot)$ is the Heaviside function, $\mathcal{H}(x) = 1$ for $x > 0$ and $\mathcal{H}(x) = 0$ otherwise, and $U_i(t)$ is the control potential,

$$U_i(t) = \sum_i C_{ij} S_j(t+1) + U_i(t-1) + s_i + f_i \tag{2.74}$$

for connection strength C_{ij}, input s_i and variable firing frequency function f_i.

The network is applied to 7×5 images of numerical digits, together with noisy versions of the digits, as was done for Kohonen and Oja's novelty filter, and is shown to be superior on this task to a back-propagation network. Unlike Kohonen and Oja's filter, the network is not limited to off-line use, since novel inputs can be shown repeatedly and so learnt. Furthermore, a quantification of the amount of novelty is possible by looking at the time to convergence. However, the filter is not capable of forgetting perceptions that are seen only infrequently.

2.9.5 Change Detection

Several authors have investigated change detection, i.e., recognising abrupt changes in signals. This is one part of novelty detection. Such a change may be found when machinery breaks, for example. One approach that has been used in the literature is the Generalised Likelihood Ratio (GLR) test. This is a Neyman-Pearson test (MacDonald, 1997) that decides whether the null hypothesis, that no change occurred in the time between two measurements with unknown probability density functions, is true. The GLR has been implemented as a time-delay neural network and applied to a time series problem by Fancourt and Principe (2000). It is not suitable for on-line use.

Another approach to change detection is described by Linaker and Niklasson (2000). The purpose of the method is to segment a time series made up of robot sensor data into a number of different phases so that the robot can recognise different parts of an environment, for example, walls, corridors and corners. A similar task was performed by Tani and Nolfi (1999) using a hierarchical mixture of recurrent networks that attempted to predict the next perception based on previous inputs. By way of contrast, Linaker and Niklasson (2000) propose an adaptive resource allocating vector quantisation (ARAVQ) network. This is based on the idea of finite moving averages of sensor data, encoding the current situation as a function of a bounded interval of past inputs. A change in input is detected when there is a significantly large mismatch between the moving average $\overline{\mathbf{x}}$ of the input and the model vectors $M(t)$, and the situation is stable, i.e., the difference between the values of $\overline{\mathbf{x}}$ for the last n perceptions and the actual last n values is less than some threshold. The approach is shown to be able to differentiate between corridors, corners and rooms using the Khepera simulator, although the processing is done off-line after all the data has been collected.

2.9.6 Unusual Approaches to Novelty Detection

A number of other methods have been proposed in the literature for novelty detection. As they do not fall into any of the previous categories, they are described in this section.

Self-Nonself Discrimination

A technique inspired by the action of the immune system is presented in Dasgupta and Forrest (1996). The immune system performs self-nonself discrimination to recognise foreign cells, which are potential infections. This technique has been used for computer virus detection in previous work (D'Haeseleer et al., 1996).

A set of strings are generated that describe the state of the system. From this, a set of detectors are generated that fail to match any of the strings in the set. The match that is measured need only be partial, so that the strings only need to match in r contiguous places, for some value of r. At each timestep a new set of strings that describe the state of the system are generated, and compared to the set of detectors. If a match is found at any time then the state of the system is decreed to be novel. The approach is applied to some data on tool breakdowns and is shown to recognise up to 90% of problems on that dataset.

This technique could be adapted to operate on-line after the initial training stage, with detectors being removed from the test set if they matched inputs that were added at a later date. Similarly, forgetting could be implemented by timestamping the creation of strings and monitoring them. However, overall it is not ideal for use on the robot, especially since there has to be an initial training stage before the selection of a set of strings to represent the data.

The Independent Component Classifier

Linares et al. (1997) describe a feed-forward neural network based on an Independent Component Classifier, with one neuron for each class, and another for the novel class. Each class has an associated prototype. The novelty cell is activated if the current input is significantly outside the space spanned by the current prototypes, i.e., if for input $\mathbf{x}$,

$$\frac{\|\mathbf{x} - \mathbf{x}_{\mathrm{Ct}}\|}{\|\mathbf{x}\|} > \rho \tag{2.75}$$

where $\mathbf{x}_{\mathrm{Ct}}$ is the component of $\mathbf{x}$ inside the prototype space. This approach can be used on-line, since the creation of a new prototype can signify novelty, but it cannot be used for forgetting.

A Competitive Learning Tree

A very different approach is described by Martinez (1998), who proposes a neural competitive learning tree that adapts on-line to track time-varying distributions. The current estimated model from the tree is compared to an *a priori* estimate (for example, made from the tree at some previous stage of learning), with a mismatch between the two models signifying novelty. This is usable on-line, but the quantification of novelty is not possible, nor can the method forget perceptions that are seen only rarely.

The Generalised Radial Basis Function Network

A Generalised Radial Basis Function (GRBF) network is used for novelty detection by Albrecht et al. (2000). Reverse connection from the output layer back to the central layer are used to make the GRBF self-organise a Bayes classifier (an off-line method). This network is then used as a novelty detector by evaluating the activity of the central layer,

$$A(\mathbf{x}|\theta_c) = \sum_{r=1}^{M} a_r^l(\mathbf{x}|\hat{P}_r, \theta_r) \tag{2.76}$$

for parameters of each centre $\hat{P}_r$ and θ_r, with a_r^l being defined by

$$a_r^l(\mathbf{x}|\hat{P}_r, \theta_r) = \hat{P}_r \hat{p}(\mathbf{x}|r, \theta_r), \tag{2.77}$$

where $\hat{p}(\mathbf{x}|r, \theta_r)$ is the multivariate normal distribution. Then, if the activity of the central layer, $A(\mathbf{x}|\theta_c)$, is below some threshold, the current input is considered to be novel.

The Eigenface Algorithm

Hickinbotham and Austin (2000) use a method that is particularly tailored to the problem of detecting faults in aeroplanes. During each flight a frequency of occurrence matrix (FOOM) is generated, with counts of particular stress events. A novelty detection technique is applied to analyse these FOOMs, a technique that it is only possible to perform off-line. This method is related to the Eigenface algorithm (Turk and Pentland, 1991) used to recognise images of faces.

The mean average FOOM, Φ, of the training set $\{\Gamma_n\}$ is calculated, together with the deviation from this mean for each FOOM ($\Psi_n = \Gamma_n - \Phi$) in the training set. Then the eigenvalues λ_k and eigenvectors $\mathbf{v}_k$ of the matrix $\mathbf{L}$ defined by

$$\mathbf{L}_{j,k} = \Psi_j^T \Psi_k \tag{2.78}$$

are computed. The M eigenvectors with the largest eigenvalues are used to compute the so-called eigenFOOMs,

$$\mathbf{u}_m = \sum_k \mathbf{v}_m k \Psi_k. \tag{2.79}$$

A new FOOM is evaluated by computing the deviation of the new FOOM from the

mean, and the distance of that to each of the principal components.

2.10 Summary

This chapter has demonstrated that novelty detection is a well-known phenomenon that has been studied in the machine learning literature for a long time. A short investigation of biological and psychological data on novelty detection demonstrated that animals and humans can and do detect novel features of their environment and that it is a useful survival trait. The von Restorff test also suggests that novelty detection has uses in deciding which perceptions should be committed to memory and neurological evidence shows that many parts of the brain that are related to memory, such as the hippocampal region, are involved in novelty detection.

Habituation, the very simple method of learning by which animals ignore stimuli that are seen frequently without ill effects, was introduced and a number of different models of it were described. The effects of habituation were described as being exactly those of a novelty filter – stimuli that have not been seen before are highlighted, while frequently seen stimuli are ignored.

The majority of the chapter has been taken up by describing methods of novelty detection already described in the machine learning literature and in the related field of statistical outlier detection. In general, all of the methods build some model of a training set that is selected to contain no examples of the important (i.e., novel) class. Deviations from this model are detected in some way, for example, by computing the bit-wise difference between the current input and the closest match in the training set, like Kohonen and Oja's novelty filter (section 2.4) or computing a local estimate of the probability density function of the data, in kernel density estimation (section 2.6.2).

The techniques that have been described from the statistical literature have followed a similar approach – modelling the support of the dataset and then detecting inputs that do not belong to that support. The choice of whether to use statistical methods or machine learning methods will always depend upon the data that is available, the application that the user has in mind and the knowledge of the implementor.

Having described so many different ways in which novel inputs (outliers) can be detected, what features of novelty detection are still interesting to researchers, and what can this thesis add?

This question will be answered from the viewpoint of the principal application of this thesis, which is the use of novelty detection for inspection tasks on a mobile robot.

The robot travels through several environments recording perceptions with its various sensors, and has to build a model of the environment through these perceptions. For this reason the filter must be able to operate on-line; the novelty of each input should be evaluated and then, during training, novel inputs should be added to the model. The adaptation time should be fast, too, so that perceptions do not have to be seen hundreds of times before they are learnt. None of the novelty filters described covers all of these points.

Another point that has not been considered fully in the work described in this chapter is the quantification of novelty. It can be useful to know whether a stimulus has been seen frequently, only occasionally or not at all. It can also be useful to know whether a similar perception has been seen. One reason for this would be so that the most novel of a set of novel stimuli can be dealt with first, or because completely new features require special attention. Quantifying the amount of novelty in a perception also makes the filter robust to small errors in the training set, as the feature is still novel the next time that it is seen, albeit less so. Therefore, a few novel stimuli being present in the training set is not an insurmountable problem.

Finally, very few of the novelty filters described in this chapter have allowed stimuli to be forgotten about, so that if they are not seen for a long time they are found to be novel again (although it could be added to several of them without too much difficulty). This is a useful ability for both on-line learning and the quantification of novelty, and has a significant benefit during training – if a stimulus is seen once or twice in the training set, but not again then it should still be found to be novel.

A novelty filter that demonstrates all of these properties is the focus of this thesis. The filter is described in the next chapter.

Chapter 3

The Novelty Filter Based on Habituation

An on-line novelty detector is introduced, initially in abstract, and then a number of neural networks suitable to act as the basis of the filter are described.

3.1 Introduction

The previous chapter described a number of approaches to novelty detection that have been used in the literature. Two principal problems with these approaches were identified. Most of the novelty filters described cannot operate on-line, evaluating each perception with respect to the inputs seen so far and then assimilating the new input into the model; and the filters only provide information about whether an output is novel or not, without giving some numerical indication of how novel a particular input is, that is, whether something like it has ever been seen before.

This chapter describes a new novelty filter that deals with these problems. The novelty filter is based on a clustering network and uses synapses that habituate (decrease their efficacy) as they are used (habituation was described in detail in section 2.3). In this way the filter shows the strength of the habituation synapse for each input, which provides an evaluation of the novelty, and then the filter is trained to recognise that input better.

A number of unsupervised neural networks suitable for implementing the novelty filter are described. These different implementations are compared in chapter 4. Finally, a problem that limits the use of the novelty filter when implemented with a static

(not growing) network is described.

3.2 An Abstraction of the Novelty Filter

In section 1.3 a set of objectives for a novelty filter suitable for use on a robot inspection platform were described. Several of these objectives were used to demonstrate that the novelty filters described in chapter 2 were not capable of the task. The main shortcomings were in on-line operation, in quantifying the amount of novelty in a perception and in the ability to forget stimuli that are seen only infrequently. The novelty filter that is described in this section aims to avoid these problems. The fundamental part of the novelty filter is a collection of habituable synapses that link the nodes of a clustering network to an output node.

The node of the clustering network that best represents the current input is selected, and sends a signal along its synapse to the output neuron. This signal is modified by the fact that the synapse habituates with use. Initially, all the synapses propagate the signal without changing it, but as they are used more, the efficacy of the synapse drops and so the strength of the signal passed is attenuated.

Therefore, a strong signal arriving at the output neuron means that the input is novel, whereas no signal, or only a weak signal, arriving at the output neuron means that the input has been seen frequently before. Hence the network filters out common perceptions, only allowing inputs that have not been seen before, or not seen often (novel inputs) to pass through.

This results in a novelty filter that learns to recognise inputs as they are presented and simultaneously evaluates the novelty of the input with respect to the model learnt so far. The novelty filter, then, consists of two separate parts:

- a clustering network
- a set of habituating synapses connecting the network nodes to an output neuron.

Since the only use of the clustering network is to recognise perceptions that are similar, the novelty filter can use whatever network is suitable for the current task. In general this will be an unsupervised network. Some possible networks are described in sections 3.3 and 3.4. Any of the models of habituation described in section 2.3 can be used. The choice of which to use depends on the task that needs to be performed.

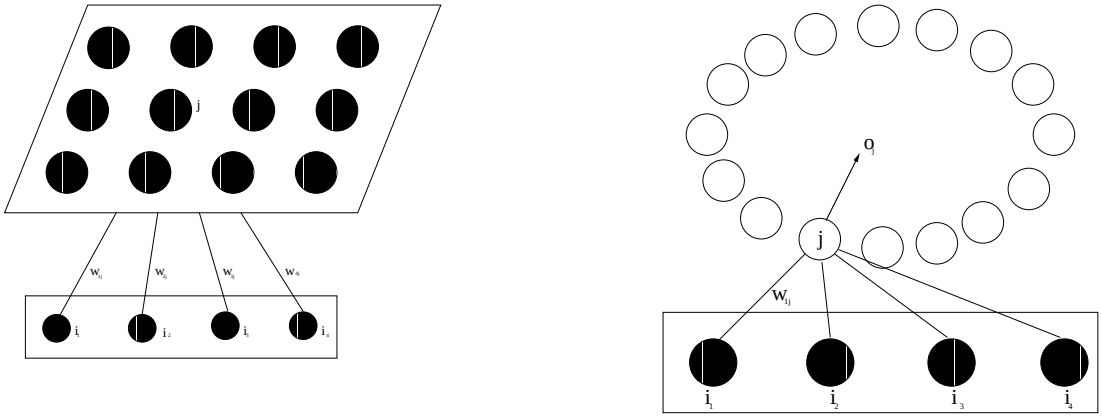


Figure 3.1: Examples of Self-Organising Maps. *Left:* A standard two-dimensional map. *Right:* A 1D map using a ring of neurons.

3.3 An Implementation of the Novelty Filter

3.3.1 The Self-Organising Map (SOM)

Kohonen's Self-Organising Map (Kohonen, 1982, 1993) is one of the best-known and most popular neural networks in the literature. Indeed, a recent survey (Kaski et al., 1998) cited 3343 papers that involved the SOM algorithm in some way, whether analysing it or using it for any of a wide variety of tasks. The SOM consists of a set of competitive units that form a lattice with a topological neighbourhood function. The algorithm itself, which was inspired by sensory mappings found in the brain, is relatively simple. The lattice of map nodes (neurons, implemented as vector quantiser units) are each connected via adaptable weight vectors to each element of the input vector. The SOM algorithm performs competitive learning, but instead of just the winning node being adapted, nodes that are close to the winner in the map space (neighbours) are also adapted, although to a lesser extent.

This means that, over time, nodes that are close together in the lattice respond to similar inputs, so that the set of local neighbourhoods self-organises to produce a global ordering. The dimensionality of the map space and the number of nodes in the network are chosen in advance, typically the map is one- or two-dimensional. The appearance of the network for these dimensionalities is shown in figure 3.1. In the one-dimensional case, shown on the right of figure 3.1, the map has been bent to form a continuous ring of neurons. This is a useful way of avoiding the problem of how to deal with the neighbourhood of a neuron at the edge of the map. A similar thing can be done for the two-dimensional map, by mapping the nodes onto the surface of a torus.

For a given input, the distance between the input and each of the nodes in the map field is calculated, usually as the Euclidean distance

$$d_j = \sum_{i=0}^{N-1} \left(v_i(t) - w_{ij}(t)\right)^2, \tag{3.1}$$

where $v_i(t)$ is the input to node i at time t and w_{ij} is the strength of the element of the weight vector between input i and neuron j.

The node with the minimum d_j is selected as the winner, and the weight for that node and its neighbours in the map field are updated using:

$$w_{ij}(t+1) = w_{ij}(t) + \eta(t)\left(v_i(t) - w_{ij}(t)\right), \tag{3.2}$$

where j is the index of a node in the neighbourhood, and η is the learning rate ($0 \leq \eta(t) \leq 1$).

In vector form these rules are written

$$d = (\mathbf{v} - \mathbf{w_i})^2 \tag{3.3}$$

$$\Delta\mathbf{w_i} = \eta h_\lambda(\mathbf{v} - \mathbf{w_i}), \tag{3.4}$$

where h_λ is the neighbourhood function.

There are a number of different ways of initialising the network and of choosing and adapting the neighbourhood size and learning rate. The simplest way of initialising the weights is to give them small random values. Then, to ensure that the network produces a topology-preserving ordering of the weights, a large neighbourhood size and learning rate are used initially, with these variables decreasing over time, for example by an exponential reduction (where η is the learning rate):

$$\eta(t) = e^{-\kappa t/\tau}, \tag{3.5}$$

for constants κ and τ, and using a similar equation for the neighbourhood size. This method assumes that the data will be presented in batch for many hundreds of iterations, usually with two different values of κ. In the first training regime κ is large. This serves to position the nodes in weight space, so that the topology ordering is preserved. Then, during the second phase a smaller value of κ is used, so that the learning rate and neighbourhood size are smaller, and the network is fine-tuned to match the input space better.

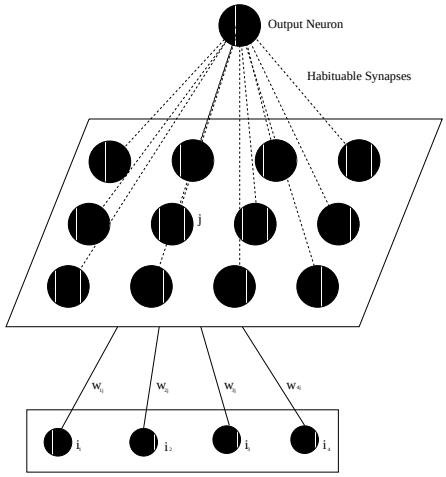


Figure 3.2: The habituating self-organising map (HSOM). The input layer connects to a clustering layer that represents the feature space. The winning neuron (i.e., the one 'closest' to the input) passes its output along an habituable synapse to the output neuron so that the output received from a neuron reduces with the number of times it fires.

Another approach, which requires that the dataset on which the network will be trained is known in advance, is to use PCA (which was described in section 2.8.5) on the dataset and initialise the nodes in the directions of the first two principal components (or as many principal components as there are dimensions in the map space). In this case the initial training phase with the large learning rate and neighbourhood size is not required.

Although the SOM learning algorithm is very simple, analysis of the network is extremely difficult. There are two important areas for analysis – under what circumstances is the convergence of the network guaranteed, and when will the self-organisation process be, in some sense, optimal. These problems have been the subject of investigation for over ten years, but only in the case where the map is one-dimensional has a complete analysis guaranteeing convergence under certain conditions been achieved. A survey of the subject is given in Cottrell et al. (1997), and self-organisation is discussed in more detail in section 5.5.

3.3.2 The Habituating Self-Organising Map (HSOM)

The Habituating Self-Organising Map (HSOM), which is shown in figure 3.2, is a novelty filter based on the SOM. The only modification is that each of the nodes in the map field is linked to an output neuron by an habituable synapse. The winning node in the SOM (the one with the smallest d_j) propagates its signal along the synapse

connecting the node to the output, while the other nodes do not fire. This synapse then habituates, as, to a lesser extent, do those of the neighbouring neurons. The habituation is controlled by equation 3.6:

$$\tau \frac{dy_j(t)}{dt} = \alpha \left[y_0 - y_j(t) \right] - S_j(t), \tag{3.6}$$

(using the notation described in section 2.3.2). When a non-zero input stimulus $S_j(t)$ is used (so that the output of that node of the SOM is propagated along the synapse) the synapse habituates, that is, its strength decreases. The values of other parameters are application specific and will be discussed in section 4.2.

By allowing the synapses of neurons that are neighbours of the winning neuron to habituate too, although to a lesser extent, perceptions that are similar to the current input will also be found to be less novel. This means that the HSOM has the ability to generalise between similar inputs by using the self-organisation property of the SOM. By varying the amount by which the synapses of neighbouring neurons habituate and the size of the neighbourhood, the amount of generalisation can be controlled.

The HSOM is the implementation that will be considered during the next chapter, where robotic experiments of the novelty detection capabilities of the novelty filter based on habituation will be investigated. This is because the SOM is one of the most commonly used unsupervised neural networks, and has the self-organisation property, which means that similar perceptions are close together in the map space, so that the novelty filter can generalise between perceptions that resemble each other. Other networks that are considered as the basis for the filter are described in section 3.4.3.

3.4 Variations on the Novelty Filter

3.4.1 Forgetting

In the previous section, no mention was made of what happens to the synapses connecting nodes that are not in the neighbourhood of the winning node to the output. There are two choices for this, either they can receive no input at all, so that the strengths of the synapses do not change, or they can receive an input of zero. In this case the strength of the synapse will increase a little (dishabituation, as discussed in section 2.3). This means that the novelty filter will find a perception that is not in the neighbourhood of the current winning node a little more novel the next time that such a perception is seen. This is called forgetting. Note that the network is not forgetting

the perception – the node of the network that recognises that particular stimulus will still recognise it, but the novelty filter will consider it to be slightly more novel next time that the perception is seen.

Under certain circumstances this is beneficial. For example, it may not be possible to exert complete control over the training set to ensure that no examples of novel data are seen. This may be because it is very costly to check every element of the set to ensure that none of them are novel, or it could be because of the way that the data is collected. An example of this is the inspection experiments that are described in section 4.3. In these experiments a robot explores an environment using its sensors to perceive the environment. These environments are not static, for example, people walk past the robot. If people walking past the robot is an occurrence seen only infrequently, it is desirable that it should be detected as novel. The forgetting ensures that perceptions that are seen only infrequently are not remembered and so are found to be novel. This gives the network the ability to deal with novel stimuli that are seen occasionally in the training set, but that should be found to be novel, which makes the filter even more useful for on-line learning. Another use of forgetting is described in the next section.

3.4.2 Encoding Recency and Familiarity

In section 2.2 the work of Brown and Xiang (1998) was described. They found evidence for the existence of neurons that respond maximally to novel stimuli (novelty neurons). These neurons were found in the perirhinal cortex of the monkey, and were one of three types, the others being recency neurons and familiarity neurons (where familiarity means seen recently, but not sufficiently often to be normal). This section describes how the same functionality can be gained using the novelty filter described in this section. No claims of biological plausibility are made, it is just shown that it is possible to demonstrate the desired functionality.

The idea is simple. Two separate habituation synapses are used to connect each node in the map field to the output neuron. One of the synapses incorporates forgetting for nodes that are not in the neighbourhood of the winning nodes, while the other does not. The forgetting synapse also has much faster habituation, so that it recognises a stimulus as being normal within two or three perceptions. Then the strengths of the two synapses can be compared for particular inputs. The different possible results, together with the interpretation are shown in the table below:

Output of		
forgetting synapse	non-forgetting synapse	Interpretation
not habituated	not habituated	novel
not habituated	habituated	familiar
habituated	not habituated	recent
habituated	habituated	normal

3.4.3 Other Networks

As has been mentioned previously, any unsupervised network is suitable for use in the novelty filter. The HSOM, where the network was based on the SOM, was described above. A number of other networks that were considered are described below. All of these networks were tried in simple simulations of robot experiments, and those that worked well were considered for robot experiments, although only some of the networks met the computational constraints of the Fischertechnik robot on which the novelty filter was used for the initial experiments. The robot and the experiments are described in section 4.2. The HSOM was used in preference to the others because the network performed best of all the networks that met these computational constraints.

In each case habituable synapses were used to link each node of the network to the output neuron, with the synapses habituating whenever that node was the winner. If the network uses the concept of the neighbourhood then the neighbouring synapses also habituated, though to a lesser extent, as in the SOM described above.

The *k*-means clustering algorithm One of the simplest way to cluster data is to use the *k*-means algorithm (Duda and Hart, 1973). This technique partitions the data into k partitions, where k is a pre-chosen constant. The partitions are chosen so that the sum-of-squares clustering function,

$$J = \sum_{j=1}^{k} \sum_{n \in S_j} \|\mathbf{x}^n - \mu_j\|^2 \tag{3.7}$$

is minimised for data points $\mathbf{x}^n$, where μ_j is the mean of the points in partition S_j.

It can be carried out as an on-line or batch procedure, with the on-line version using the update rule

$$\Delta\mu_j = \eta \left(\mathbf{x}^n - \mu_j\right). \tag{3.8}$$

The RCE network, which was described in section 2.7.3

The ART network, which was described in section 2.7.2

The Neural Gas network The Neural Gas network (Martinetz et al., 1993) is similar to Kohonen's SOM, but rather than all of the neighbours of the winning node being updated by the same amount, the reference vectors are ranked, and the weights are updated proportional to their position in the list. For on-line learning this leads to an update rule of:

$$\Delta \mathbf{w}_i = \eta h_\lambda \left(k_i(\mathbf{v}, \mathbf{w})\right) \cdot (\mathbf{v} - \mathbf{w}_i), \tag{3.9}$$

where $\mathbf{v}$ is the current input and $\mathbf{w}$ is the set of all weight vectors $(\mathbf{w}_1, \cdots, \mathbf{w}_N)$. h_λ is an exponential neighbourhood function, with λ controlling the size of the exponential and k_i being the ranking of the ith node in terms of activity.

This network has been shown to converge in fewer iterations than the SOM for a number of clustering problems and also for time-series prediction. Unfortunately, the need to sort the reference vectors at each iteration means that the time complexity of the algorithm is significantly higher.

3.4.4 Networks for Temporal Clustering

Two clustering networks designed to deal with temporal information were also tried. Neural networks are frequently used to deal with information that varies with time, for example time series prediction such as the Mackey-Glass time series (Mackey and Glass, 1977). Recognising data that varies over time is also of interest for mobile robot applications, as series of perceptions that occur as the robot travels past an object can also be clustered together.

Any network can be augmented to deal with temporal information by producing a lag vector of inputs, where the current inputs to the network are those not just of the present time, but also those of some number of previous time steps. This section describes two other methods for dealing with temporal data in an unsupervised way.

The Temporal Kohonen Map This self-organising map, proposed by Chappell and Taylor (1993), see also Taylor (1995) is based on Kohonen's SOM, but uses "leaky integrator" neurons whose activity decays over time. This is like a short-term memory, allowing previous inputs to have some effect over the processing

of the current one. It was used by Lambrinos et al. (1995) to coordinate motor controls with sensory states on a Khepera robot.

The activity of the neurons is calculated using

$$a_i(t) = d \cdot a_i(t-1) + e^{\left(-\frac{1}{2}\right)(\mathbf{w}_i(t)-\mathbf{v}(t))^2} \tag{3.10}$$

(where the constant $d < 1$ controls the time constant of the neuron) and weight update rule

$$\Delta \mathbf{w}_i = \eta \sum_{k=0}^{n} d^k (\mathbf{v}(t-k) - \mathbf{w}_i(t-k)). \tag{3.11}$$

The Temporal Activity Diffusion Network The TAD network, and its predecessor, the Self-Organising Temporal Pattern Recogniser (SOTPAR) are the work of Euliano and Principe (1996, 1999). Based on the Neural Gas and SOM respectively (both of which are described above) the TAD and SOTPAR networks exhibit:

- Competitive learning with a neighbourhood function
- Activity diffusion through the output space of the map field neurons
- Temporal decay of activations

Turing's Reaction-Diffusion equation, which describes the interaction of chemicals, is used to describe travelling waves of activity as each excited neuron passes on some of its energy to its neighbours. The neurons themselves are of the leaky integrator variety, so that their membrane potential decays over time. The network that was tested was the one based on the Neural Gas algorithm. Training is done using the update rule for the neural gas, given in equation 3.9. The connection matrix weights are trained using Hebbian learning, with weights being updated when one of the neurons fires.

3.5 Problems with the HSOM

In the description of the SOM in section 3.3 the lack of a guarantee of convergence of the network was briefly discussed. In addition, the network performs best when the inputs to the network are presented in batch many times, and when the dataset is known in advance, so that the network weights can be initialised to match the data distribution

that is presented. However, this is frequently not the case. For example, the data may not be known in advance, and the network may have to operate on-line, so that the data cannot be presented several times. In these cases the performance of the network is unlikely to be optimal; the SOM is not suitable for continuous on-line learning.

The addition of the habituation synapses exacerbates the problem. Equation 2.2 only has a fixed point at the bottom of the curve (i.e., fully habituated) when a stimulus is applied, and at the top of the curve (i.e., fully dishabituated) when a stimulus is not applied. This means that for synapses that do not dishabituate, once a particular perception has been seen sufficiently often for the output synapse attached to that node to habituate, any perception that maps to that node will be found to be normal, no matter what it is. In other words, the novelty filter habituates to the firing of a particular node, not to the perception that caused the node to fire. The number of nodes in the network is decided when the network is created, there is no option for the network to create new nodes.

Therefore, once the number of different perceptions that are seen is of the order of the number of nodes in the network, every perception – no matter what it is – will map onto one of the nodes that has already been used, and will therefore be found to be normal. That is, the network will saturate so that any input, regardless of how novel it is, will be found to be normal. If the number of perceptions is (at least approximately) known in advance, then the size of the network can be picked to ensure that the network will not saturate. In practise, though, this can be hard to do. The problem occurs less if the network is trained in two phases or initialised using the principal components of the data, because then the nodes cover the space effectively. However, as was discussed previously, this is not always possible.

The dishabituation of synapses does not affect the problem particularly. Although some of the synapses will be dishabituated, the nodes that they are attached to are already committed to a particular type of perception, and so the node is not free to take another perception.

This chapter has also described several other networks that can be used as the basis for the novelty filter. As several of them can add extra nodes into the map field, they do not suffer from the same problem. The networks that can add nodes, and therefore will not suffer from this problem, are the RCE network and ART. However, the performance of the ART network on the simple robot task described in section 4.2 was extremely disappointing. The program was too complicated to operate on the robot itself, and therefore had to be run in simulation, but even then the results were not promising. It

appears that the network suffered from noise in the readings, and did not generalise well. The RCE network is much simpler. Section 6.3 compares the performance of the RCE network against other networks on a robot inspection task and shows that the lack of the concept of a neighbourhood limits the usefulness of the network.

One possible solution to the problem would be to look at the pattern of activity over the whole network when an input is presented. However, this pattern will change as the network trains, and so will not be stable. The solution that is proposed in this research is a new neural network that adds new nodes when they are required and maintains neighbourhood connections between nodes. The new network is described in chapter 5.

3.6 Summary

This section has described the concept behind the novelty filter proposed in this research. In abstract, this novelty filter consists of an unsupervised neural network that clusters together similar inputs and a set of habituating synapses that connect each node of the network to an output neuron. The best-matching node of the network propagates its signal along the synapse to the output node, and this synapse habituates with use. For networks that use the concept of a neighbourhood, such as the SOM described above, the neighbouring synapses also habituate, although to a lesser extent.

A number of networks that are suitable for implementing the novelty filter are described, particularly the SOM. The HSOM network is used as the basis of the novelty filter in the experiments that are described in the next chapter. These experiments are designed to highlight the benefits of the novelty filter as a tool for preprocessing the perceptions of a robot to allow it to respond to changes in its operating environment. The experiments demonstrate the effectiveness of the novelty filter and the effects of allowing the synapses to dishabituate, causing forgetting.

In section 1.3 a total of nine goals were set for a novelty filter. Of these goals, the HSOM is capable of the first six, in that the filter can detect novel inputs, generalise between them by using the self-organisation property of the SOM, operate on-line and quantify the amount of novelty in a perception (with regard to the inputs that the network has seen previously) by using the strength of the habituation synapse – stimuli that have been seen frequently will have caused the synapse to have habituated fully, rare stimuli will still have strong synapses. In addition, by using the forgetting ability that comes with dishabituation, the novelty filter can always find stimuli that are seen

very infrequently to be novel.

Section 3.5 described a number of problems that can be found with the HSOM and related networks. The principal problem is that the network can saturate, since the only fixed point in the dynamics of the habituation equation is where the synapse strength is fully reduced. The network is trained on-line, so that each perception is seen only once and then discarded. Saturation means that any perception, no matter what it is, will be found to be normal. This is clearly a major problem for a novelty filter.

The next chapter describes a series of experiments with the novelty filter, first the task of directing attention towards the most novel stimuli and then on the inspection tasks that are the focus of this thesis. It is demonstrated that the novelty filter based on habituation can reliably detect novel features of an environment, although the effects of the saturation problems are demonstrated very clearly in section 4.4.

Chapter 4

Robot Experiments with the HSOM

This chapter describes the robot experiments performed with the novelty filter introduced in the previous chapter. It is demonstrated that the filter can allow a robot to focus its attention towards novel stimuli in its environment, and also to act as an inspection agent.

4.1 Introduction

This section applies the HSOM described in the previous chapter to two different robot tasks. The first is the question of attention, where a robot turns to face the most novel stimulus in its perceptual field. The ability to focus its attention is potentially useful for a robot, as it means that the robot does not have to investigate every feature of an environment, so that the amount of learning required can be reduced. Directing attention towards novel stimuli can be useful for a robot, as it is for an animal, because it means that the unexpected features of an environment can be dealt with first. It can also be used to reduce the amount of learning that the robot has to do, since common features do not need to be relearnt.

The second set of experiments is the first to deal with the principal focus of this research, on-line inspection. A robot uses its sonar sensors to investigate an environment that it travels through, building a model of that environment through the perceptions of the sonar sensors. Each time that a new perception is perceived the novelty filter produces a novelty value for that perception with respect to what has been seen so far, and then the perception is learnt, that is, it is added to the model. Once the novelty filter stops finding any perceptions of a particular environment novel, the robot is moved to a new environment, which it explores, with the novelty filter highlighting those features

that do not fit into the model of the previous environment. This means that the novelty of each perception in the second environment is evaluated with respect to the model acquired in the first, so each perception is tested to see if it would be normal in the first environment.

These inspection experiments are also used to investigate the effects of dishabituation, or forgetting. The scalability problems of the HSOM are considered, and it is demonstrated that the HSOM can saturate, so that no perceptions are considered to be novel, no matter what they are. This was discussed from a theoretical standpoint in section 3.5.

4.2 Attention

The first robot experiments with the HSOM and with the novelty filters based on networks other than the SOM (see section 3.4) are described in this section. They were performed using a Fischertechnik robot. This robot, shown in figure 4.1, has a Motorola 68HC11 microcontroller and a two wheel differential drive system. The robot can be equipped with a variety of sensors. In the experiments described in this section four light sensors mounted on a mast and facing in the cardinal directions were used. These sensors can be seen in figure 4.1. Further experiments used a linear array of photocells mounted on the front of the Fischertechnik robot.

The experiments had two purposes – to investigate how well the novelty filter would direct the attention of a robot towards newer, and hence more novel, stimuli and to test out the different clustering networks on which the novelty filter with habituation could be based. Indicating the direction of the newest stimulus by turning towards it has approximate parallels with the orienting response seen in animals, where the animal turns towards the most novel stimulus. The behaviour induced in the robot was termed *neotaxis*, meaning ‘travel towards what is new’. The principal benefits of this behaviour are that the experiments will demonstrate very clearly whether or not the novelty filter works – if the robot successfully travels towards novel stimuli then the filter is successful, otherwise it is not.

4.2.1 A Description of the System

Four separate HSOM networks (each a ring of 12 neurons) were used, one for each of the light sensors. At each cycle, the current reading on a light sensor (an integer value

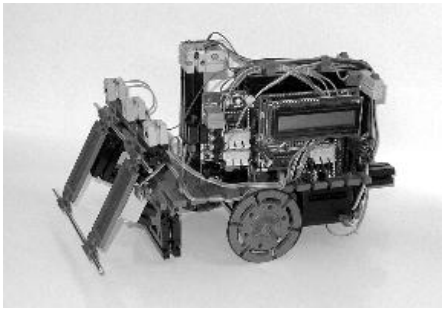


Figure 4.1: The Fischertechnik mobile robot used in the experiments where the novelty filter is used to direct attention towards the newest stimulus.

between 0 and 255) was linearly scaled to be between 0 and 1 and was concatenated with the previous n to form an $n+1$ element input vector, known as a delay line. That is, the input vector at time t consisted of the readings at times $t, t-1, t-2, \ldots, t-n$. Here, $n = 5$, so that the input vector had six elements. This input vector was classified by the HSOM network and a value for the novelty of that perception was produced. The outputs of the four novelty filters were fed into a comparator function that propagated the strongest output (i.e., the HSOM with the most novel signal), providing that it was above a predefined threshold, to the action mechanism. A schematic of the system is shown in figure 4.2. If none of the signals was strong enough to exceed the threshold, then the robot did not respond.

Each of the HSOM networks used a learning rate of $\eta = 0.25$, which remained unchanged throughout the course of the experiments so that the HSOM learned on-line. The neighbourhood comprised only the two nearest neighbours, one on each side of each node, and this also remained constant during the experiments. The dynamics of the habituation synapses were controlled by the standard habituation equation 2.2.

The values of the parameters were those used to plot the graphs in figure 2.2, $\alpha = 1.05$, $y_0 = 1$ and $\tau = 3.33$ for the winning node and $\tau = 10.0$ for the neighbours of the winning node. The 'boredom' threshold, the value of $y(t)$ below which a stimulus was no longer classified as novel, was 0.4.

As was described in section 3.4, several different neural networks were used as the basis for the novelty filter and experiments on many of them were performed. The networks that were too complex to run on the robot were run in simulation. The qualitative results for the temporal Kohonen map, and the HSOM were similar and the on-line *k*-means algorithm also performed well. The HSOM was picked for three principal reasons:

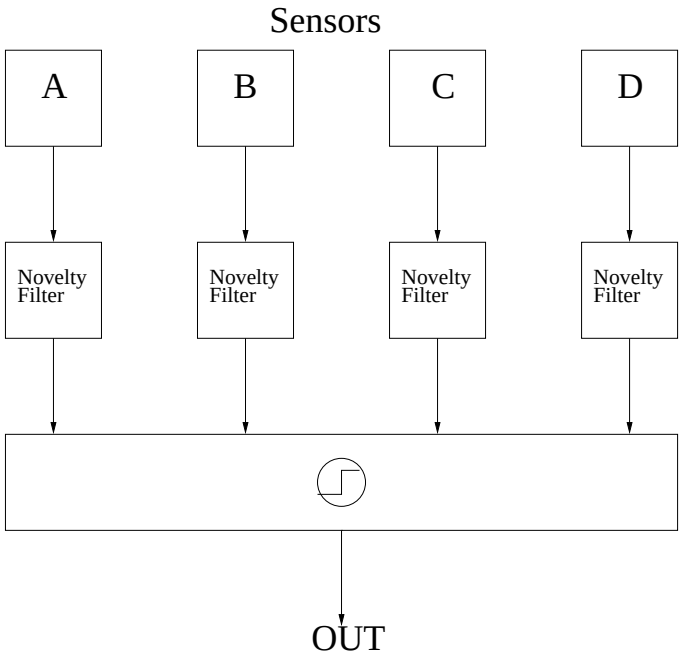


Figure 4.2: The overall system for choosing the most novel stimulus. Each sensory perception is classified by an HSOM and a value indicating the novelty of that stimulus is output. A comparator function picks the most novel of the four sensors and, providing that this is above a predefined threshold, the robot turns towards it.

- The HSOM was computationally simple enough to operate on the robot, which has very limited resources
- The HSOM appeared to be more robust to the noise inherent in the system than the other networks that operated on the robot
- The HSOM learned quickly and accurately.

4.2.2 Results

Four separate experiments were carried out. In all of them the aim was that the robot oriented itself towards the most novel stimulus at each stage of the experiment.

All four experiments had the same basic form – the robot was placed into an initially featureless environment. Then, a number of different stimuli were presented to the robot, with the stimuli being light sources that were projected onto one of the four light sensors. Once the robot had responded to this stimulus, further stimuli were introduced.

First experiment Figure 4.3 shows the sequence of actions and responses during one

of the experiments. They are also shown in table 4.1. A constant light was introduced to shine onto one of the light sensors, and as this stimulus was novel, the robot oriented towards it (step (a) of figure 4.3). Once the robot had stopped moving, directed towards this light, a second stimulus, a flashing light, was added (step (b)). As this stimulus is more novel, the robot turned towards it. A further, faster flashing light was then introduced and the robot again turned towards this more novel stimulus (step (c)).

Finally, the constant light source was turned off. The response to this depended on whether or not the forgetting described in section 3.4.1 was being used. If the robot was not forgetting then the constant stimulus was not novel, and so the robot did not respond. However, if forgetting was used then the lack of a stimulus on this sensor was novel and so the robot turned to face the position where the light used to be. This is shown in step (d).

Second experiment Steps (a) and (b) of the first experiment were again performed, but this time a second flashing light of the same frequency as in step (b) was used in step (c), instead of a faster flashing light. How the robot responded to this depended on how long the robot had seen the flashing light for. If it was sufficiently long for the robot to stop finding the light novel, then the robot did not respond, otherwise it turned towards the newer input.

Third experiment In the third experiment a wider variety of flashing lights were used. Two additional patterns were used, short-short-long-long and short-long-short-long. These were obviously more complicated patterns for the novelty filter to classify, and were used to differentiate between the novelty filters based on the HSOM, the temporal Kohonen map (TKM) and the *k*-means algorithm. All three algorithms had some problems whenever the patterns of lights changed, but the TKM took longer to learn new patterns than the HSOM and *k*-means did. It was also found that occasionally the *k*-means algorithm could respond to spurious inputs due to the noise inherent in the sensors.

Fourth experiment In the final experiment a moving light was introduced in step (c) instead of a flashing light. For this experiment the robot was equipped with a linear array of photocells, and once the robot had detected the moving light, the robot attempted to follow the light as it moved. The TKM performed well at this task, with the HSOM taking some time to detect slow movement as novel. The

k-means algorithm was very jumpy, detecting novelty many times when there was none.

These last two experiments were used to test how well the networks could deal with a variety of inputs. It is on the basis of these results that the HSOM was chosen rather than the novelty filter based on the Temporal Kohonen Map or the *k*-means algorithm.

Experiment	Forgetting	Stage	Action
1	On	Constant On	Robot turns towards it
		Slow Flashing On	Robot turns towards it
		Fast Flashing On	Robot turns towards it
		Constant Off	Robot turns towards it
	Off	Constant On	Robot turns towards it
		Slow Flashing On	Robot turns towards it
		Fast Flashing On	Robot turns towards it
		Constant Off	Robot does not respond
2	On	Constant On	Robot turns towards it
		Slow Flashing On	Robot turns towards it
		Slow Flashing On	If on a different sensor, robot turns towards it
	Off	Constant On	Robot turns towards it
		Slow Flashing On	Robot turns towards it
		Slow Flashing On	If on a different sensor, robot turns towards it
3	On	Constant On	Robot turns towards it
		Slow Flashing On	Robot turns towards it
		Constant On	If on a different sensor, robot turns towards it
	Off	Constant On	Robot turns towards it
		Slow Flashing On	Robot turns towards it
		Constant On	If on a different sensor, robot turns towards it

Table 4.1: A description of the behaviour of the robot during the experiments in which the robot turned its attention towards the most novel stimulus in the environment.

4.2.3 Discussion

One problem with the system used here is that each of the sensors had a separate novelty filter. This means that a stimulus that is normal to one sensor would remain novel to the others. To deal with this, each of the novelty filters shared its input with

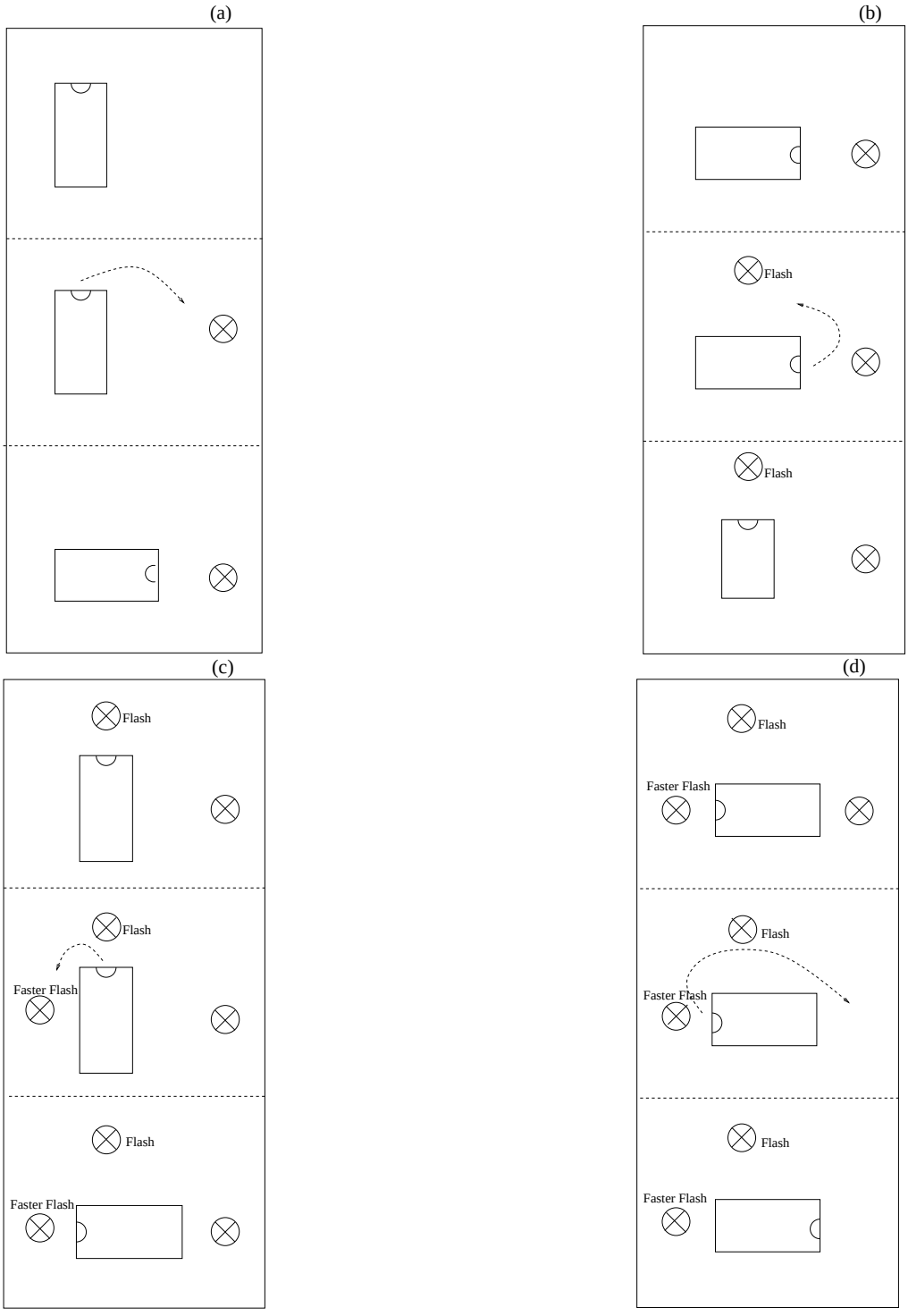


Figure 4.3: Figures showing the behaviour of the robot during the four stages of the first experiment. In (a) the robot turns towards the new light, in (b) it turns towards the newer flashing light, and then in (c) to the faster flashing light. Finally, in (d) it turns back to the point where the light has been turned off.

the others at the end of each cycle, so that all the filters perceived the same stimuli. This is equivalent to using one filter and classifying four inputs with it on every cycle. However, conceptually, using four separate filters is clearer.

The results that are demonstrated in figure 4.3 and table 4.1 show that the novelty filter can be used in this way and that the behaviour of the robot is as expected – the robot detects the most novel stimulus in its perceptual range, and turns towards it. What is not clear from the purely qualitative results is how well the filter performed. Whenever a flashing light was introduced the first few perceptions were found to be novel. This is because of the time that the first input took to work through the lag vector, and is a common problem in dealing with temporal information.

All three of the filters that were used in the robot experiments – based on the TKM, HSOM and *k*-means algorithms – had some problems dealing with the inputs. Overall, the HSOM had less difficulty with noise than the other two, and it is for this reason that it is used for further experiments in the remainder of this chapter. The main reason why it performed well compared to the *k*-means algorithm is the use of neighbourhoods, so that similar perceptions could mutually habituate each other.

These experiments used a novelty filter for each sensor separately to decide which input should be attended to. It would be more useful to decide what area of the environment – which could extend across several sensors – to focus on. Similar systems could be built to do this work, although it is not done in this thesis, which focuses only on novelty detection.

4.3 Inspection

The experiments described in section 4.2 demonstrated that the novelty filter based on habituation enabled the robot to respond to the most novel stimulus in its perceptual range at each stage. However, the Fischertechnik robot that was used in the experiments had only very limited sensory capabilities. For that reason, further experiments were performed on a more complex robot. This is the Nomad 200 robot shown in figure 4.4, a fully autonomous mobile robot. The Nomad 200 is equipped with an onboard PC, and has a ring of 16 sonar sensors, a ring of 16 infra-red sensors, a monochrome camera and a set of bumper sensors. The turret and base can rotate independently of each other. The translation and rotational movement of the robot are controlled by electric motors located in the robot's base.

In the experiments reported in this section the infra-red sensors were used to control

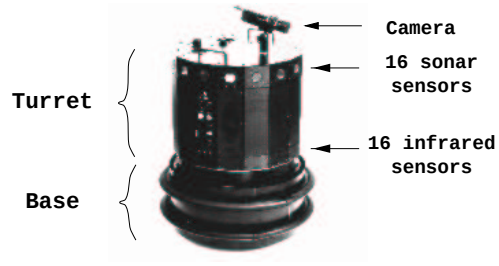


Figure 4.4: The Nomad 200 mobile robot.

the motion of the robot and the sonar sensors were used to perceive the environment for the novelty detection. This meant that the perceptions of the environment that were used by the novelty filter did not also control the movement of the robot. Although this means that the robot cannot move towards more novel parts of the environment, it also means that an independent behaviour determines the perceptions that the novelty filter sees. The use of the camera as input for the novelty filter is described in chapter 7.

The task chosen was a simple on-line inspection task. The robot explores an environment and uses the novelty filter to build a model of that environment as it is perceived through the sonar sensors. Once the model is complete (so that the robot does not find any perception in the environment novel) the robot is moved to another environment, and explores it, highlighting perceptions that were not found in the first environment and are therefore novel.

4.3.1 A Description of the Experiments

Experimental Procedure

Each experiment consisted of a number of passes through an environment. The robot was positioned at a starting point chosen arbitrarily within the environment. From this starting point the robot travelled for 10 metres using a wall-following behaviour. As the robot travelled it took continuous sonar scans. Every 10 cm the robot computed the average sonar scan over the last 10 cm of travel. Each element of this vector was thresholded at about four metres, above which the readings were unreliable, was linearly scaled so that each element had a value between 0 and 1, and was inverted so that perceptions of nearby objects were close to 1, and perceptions of far away objects were close to 0. This vector was then presented to the HSOM as an input vector. The HSOM categorised the perception and produced an output novelty value for that perception with respect to the model learned so far.

At the end of the 10 m of corridor that comprised each environment the robot stopped and saved the weights of the HSOM. It was then returned to the start of the environment using the joystick, and the process repeated using the weights learnt during the previous run, until none of the robot's perceptions were found to be novel.

The passes through the environment were performed in pairs. In the first of the two, the novelty filter was learning and the new perceptions were added to the model, while in the second trial the novelty filter was not learning. The filter was not therefore trained on the new perception, although it did still produce a novelty value for each input with respect to the model learned so far. This test run was used to discover how well the filter was learning and to see what was learnt during the previous pass. With the standard HSOM each perception is considered independently of the previous perceptions, so that there is no temporal information. This means that the order in which perceptions are seen does not matter, and that the robot would not have to start in the same place on every run.

When the novelty filter was first used, every perception was novel, since the filter evaluates the novelty of each perception with respect to the model acquired from previous inputs to the filter, but the filter soon learnt to recognise perceptions that were seen repeatedly.

The Environments

Three different environments in the Computer Science Department at the University of Manchester were used for these experiments. They are shown in figure 4.5. The first two environments (A and B) are similar sections of corridor on the second floor of the Computer Science Department. The corridors are 1.7 m wide and have breezeblock walls with wooden doors set into them. The third environment (environment C in figure 4.5) is rather wider than the other environments (2.1 m), and has bricks on one wall (the wall that the robot followed) and glass windows on the other wall.

Environment A can be altered by opening the door that is shown on the right of the robot (labelled door D) in figure 4.5. This alters the sonar perceptions at this stage, since the sonar pulses are no longer reflected back from the door, but from a wooden blockade approximately 1.5 m further away. This environment is referred to as environment A*. In order to prevent the robot from going through the doorway, a cardboard box was placed there. This box was of sufficient height to be seen by the infra-red sensors that controlled the wall-following behaviour, but was too low to be seen by the sonar sensors. Using an environment that has just one change made is a

useful test because every other part of the environment is absolutely identical, so only the area around the doorway should be found to be novel if the filter is working as expected.

4.3.2 Results

In order to investigate the behaviour of the novelty filter on the inspection task a number of different experiments were performed. For each experiment the output of the novelty filter (i.e., the activity of the output neuron) is recorded for each input. These outputs are shown in two different ways. The first method plots the output of the novelty filter as the robot travels through an environment during each run, while the second technique plots the cumulative output of the novelty filter for each run. This means that the first method demonstrates which perceptions of an environment are novel, and how novel they are, while the second method shows how the amount of novelty found in an environment changes as learning occurs.

After some experimentation a set of parameters for the network and also for the habituation dynamics were chosen. The HSOM network was made up of 100 nodes arranged in a 10×10 grid. A learning rate of $\eta = 0.25$ was used, and a neighbourhood size of the eight closest nodes selected. These parameters remained constant throughout the experiments, so that the learning behaviour did not change. This meant that the network was always learning at the same rate no matter how many perceptions had been seen. The habituation dynamics were again controlled by equation 2.2. The values of the parameters here were $y_0 = 1$, $\alpha = 1.05$ and $\tau = 3.33$ for the winning node, meaning that a synapses decreases to below 10% of its original value within five iterations. The neighbourhood nodes, which recognise similar perceptions, have a smaller amount of habituation, $\tau = 14.3$, meaning that about 30 presentations of a stimulus are needed for the efficacy of the synapse to drop below 10% of its original value.

How Novel is Novel?

The novelty of each perception takes a value between 0 and 1, with 0 meaning that similar perceptions have been seen frequently, and 1 meaning that no similar perception has been seen by the network. The question is at what stage a perception stops being novel and becomes normal or usual. In the experiments into directing attention described in section 4.2 above a ‘boredom’ threshold of 0.4 was used, below which an input was not considered to be novel. This is not done in the following experiments,

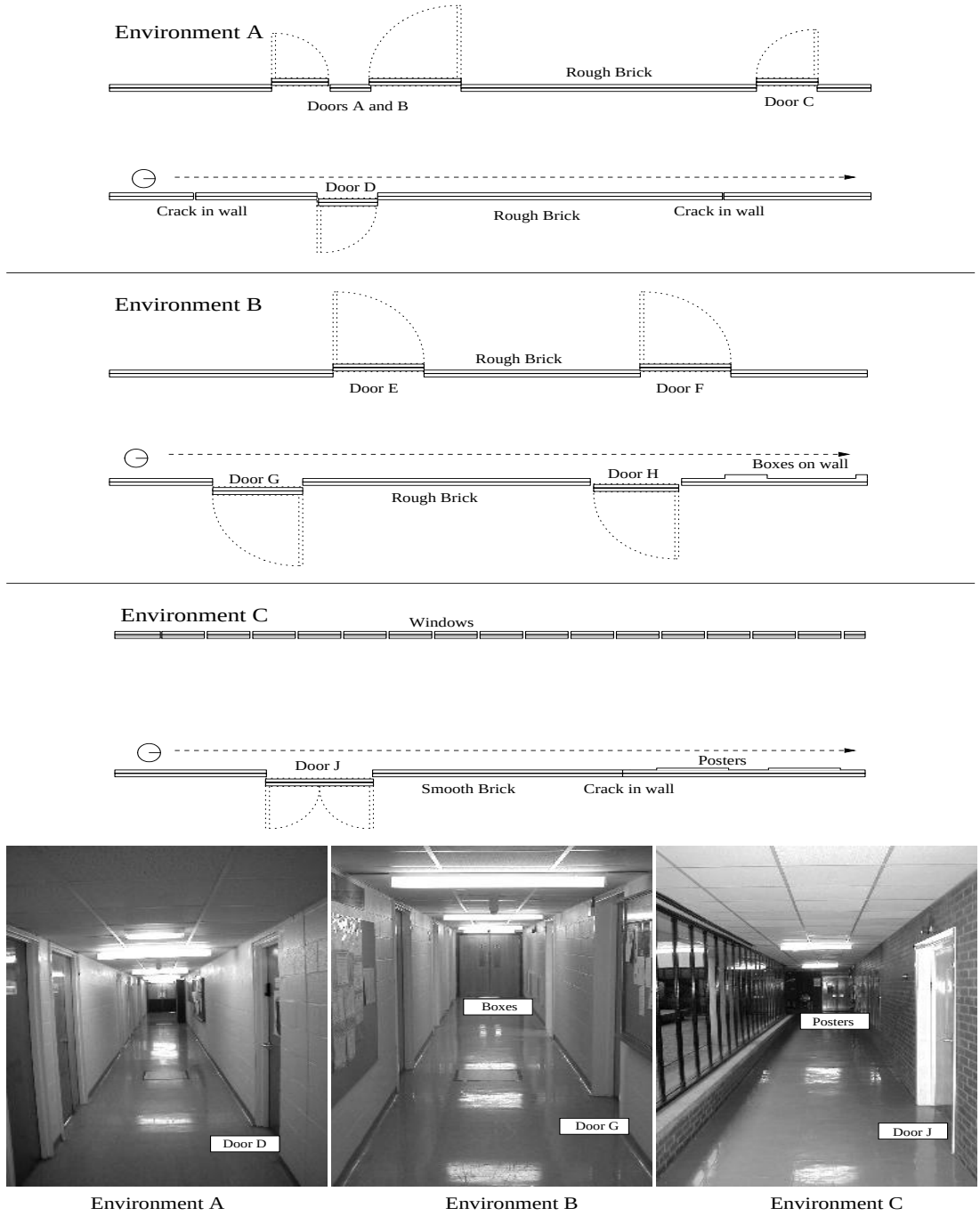


Figure 4.5: Diagrams of the three environments used. The robot is shown facing in the direction of travel adjacent to the wall that it followed. Environments A and B are two similar sections of corridor, while environment C is wider and has walls made of different materials, glass and brick instead of breezeblock. The photographs show the environments as they appear from the starting position of the robot. The notice boards that are visible in environment B are above the height of the robot's sonar sensors, and are therefore not detected.

no threshold is given.

If the amount of novelty was to be used to control the behaviour of the robot then a threshold would have to be used. The value of such a threshold would depend primarily on the parameters used in the habituation equations and the particular task. It could be chosen empirically or could be some function of the average amount of novelty found in a set of stimuli. For the experiments demonstrated in this section a threshold of around 0.3 would highlight all of the novel features and would remove most of the responses that are due to noise.

Environments A and A*

The diagram in figure 4.6 labelled 'Environment A' shows the output of the novelty filter as the robot explores environment A, with the network randomly initialised. The first plot shows that the robot initially finds all perceptions novel, but quickly learns to recognise the perceptions of the wall on both sides of the robot, since these perceptions are seen frequently. The first perceptions of the doorway are also found to be novel, but the wall that comes after it is not (since this is the same as has been seen previously).

In the second pass through the environment, where the novelty filter is not learning, it can be seen that the internal representation of the environment made during the first pass is fairly good, only some perceptions of doorways (the one on the right of the robot near the start of the environment, and the one on the left of the robot towards the end) are still novel. These are the features that are seen less often and therefore take longer to learn. It is particularly true of doorposts, which produce very different sonar readings depending upon what angle they are seen from.

After the second learning pass (Pass 3), even less is found to be novel, and after a third pass nothing is at all novel. One interesting point is that during pass 3 there is a peak near the start of the run that is not seen in the second pass. This is the perception of the crack in the wall (which can be seen in figure 4.5). The crack is a gap approximately 5 mm wide. When a sonar does see it, it produces a very large reading, and is therefore very noticeable, but mostly it is not seen.

Once the output of the novelty filter showed that the environment had been learnt well, a change was made to the environment. This was the opening of door D on the right of the environment. The novelty filter trained to recognise environment A was used in this modified environment (environment A*), and the outputs of the novelty filter after this change are shown on the right of figure 4.6.

As the novelty filter has already been trained, the first pass through the modified

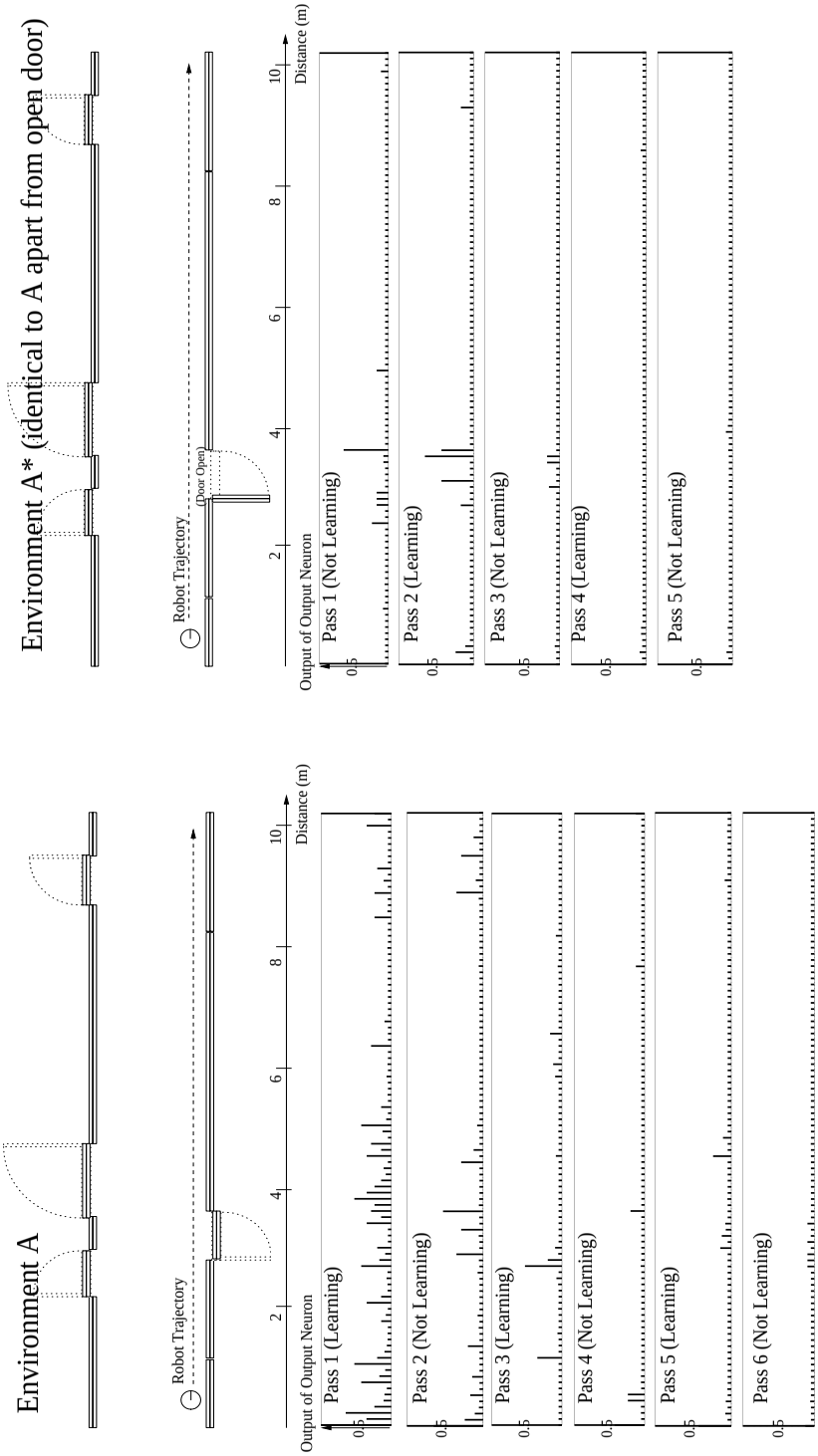


Figure 4.6: The results of the first experiment. The graphs on the left show the output of the output neuron of the novelty filter as the robot moves within environment A when learning and not learning. Once it has stopped detecting novel features (so that the activity of the output neuron is small), the environment was changed by opening a door (environment A*). Then only the perceptions around the doorway are novel.

environment was with learning turned off, to highlight the novel features. It can be seen that the perceptions that are found to be novel are those around the doorway, and that nothing else is found to be novel. The doorway is seen before the robot reaches it, and is still found to be novel after the robot has moved past it. This is because the sonar sensors are arranged in a ring around the robot, and so the perceptions of the doorway (and particularly the doorpost) are still seen by front- or rear-facing sensors when the robot approaches the doorway, and after it has passed it.

Figure 4.8 gives a different way of looking at the data, by plotting the total amount of novelty detected in each testing (i.e., non-learning) run through an environment. This is computed by summing the output of the novelty filter for each input as the robot travelled through the environment. It can be seen that the amount of novelty decreases in each run, as is expected. The left hand figure shows how the learning of the filter reduces the amount of novelty found as the robot explores environment A three times, and how the amount of novelty initially increases a little when environment A is changed to environment A* by opening the door, but decreases as the filter learns again. It can be seen that the amount by which the novelty increases when the robot is placed in environment A* after training in environment A is small, as would be expected since only one change to the environment is made.

Testing in Environments B and C

The novelty filter that was trained on environment A was also used to evaluate environments B and C. The results of these investigations are shown in figure 4.7. Environment B, shown on the left of figure 4.7, is a very similar environment to A, being in the same area of the building. The graphs of the output of the novelty filter show that only the first two doorways (one on either side of the robot) were found to be novel by the filter. On inspection of the environment, it was discovered that these doorways were inset deeper than those in the first corridor. The effects of this can be seen in figure 4.5.

The final graph on the left hand side of figure 4.7 shows the novelty values of the filter on a control trial. For this pass through the environment the novelty filter that was used was one that had been pre-trained in a large, open environment. The robot was steered about a course so that it travelled close to a wall for three metres, turned and moved into the middle of the environment and then returned to the wall. It can be seen that much more novelty was found when the novelty filter trained in this environment was used, rather than the one trained in environment A. Environments A and B are very similar, being sections of two corridors in the same part of the building.

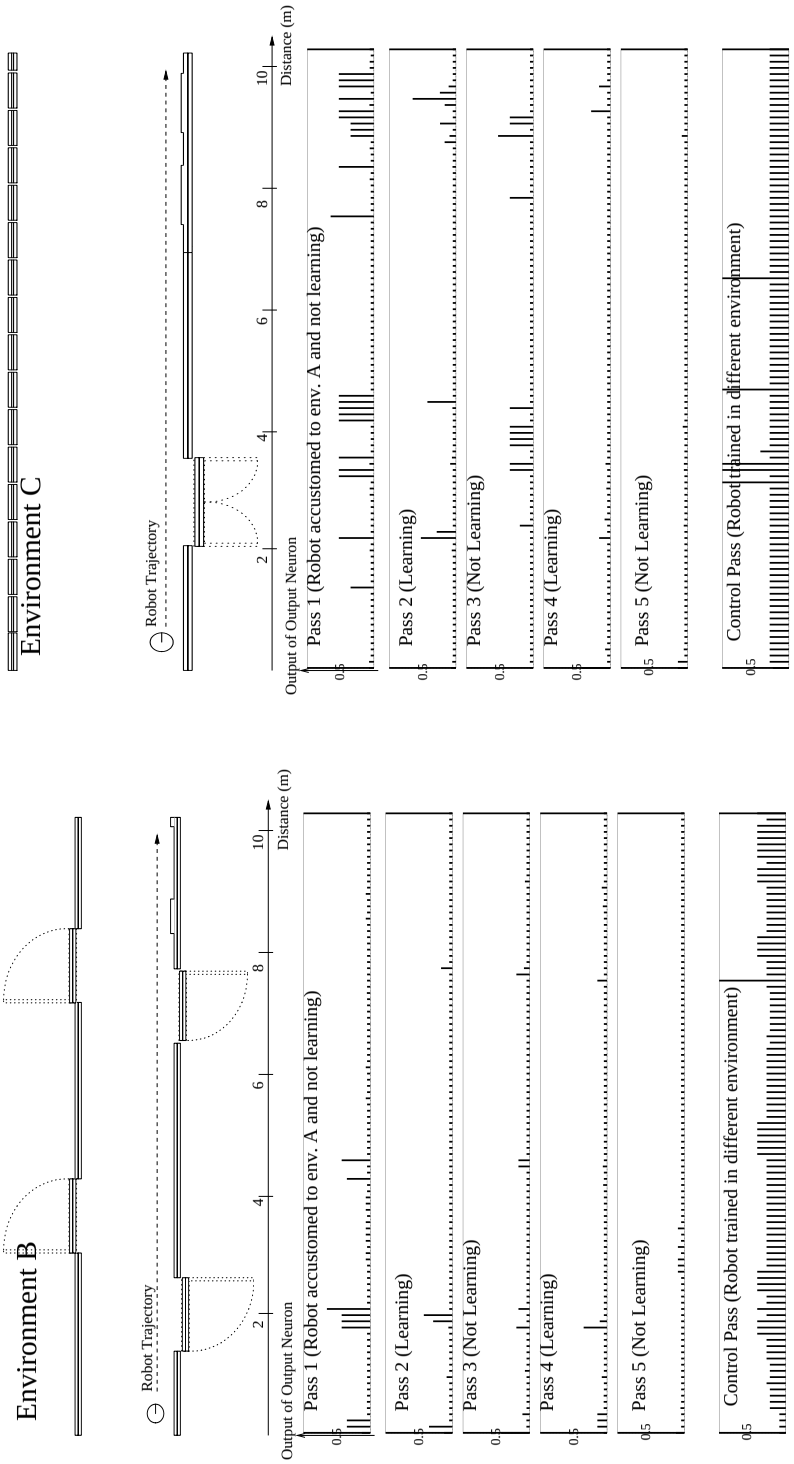


Figure 4.7: After the novelty filter had been trained in environment A (shown on the left of figure 4.6) the robot was allowed to explore two further environments using the weights of the trained novelty filter. The left of the figure shows the results when a environment similar to A was used, in which case very little is found to be novel. However, when an environment that is different is used for testing, shown on the right, considerably more novelty is found. Control trials, where the novelty filter was initially trained in a non-corridor environment, are shown in the final graphs.

Comparing the middle graph of figure 4.8 to the left hand graph, it can be seen that the amount of novelty found when the novelty filter is placed in environment B is more than when environment A is changed by opening the door. However, it is still less than if the robot were placed in a completely new environment. This is shown clearly by the graph on the right hand side of figure 4.8, where the amount of novelty that is seen in environment C (a very different environment to A and B) is similar to the amount seen when the robot first explores environment A.

The reasons for this can be seen on the right hand side of figure 4.7, which gives the output of the novelty filter when it is tested in environment C after training in environment A. It can be seen that several perceptions are found to be very novel, and because the robot is not learning it finds them novel every time it sees them. The photograph at the bottom of figure 4.5 shows how different this environment is. In particular, the doorway that can be seen on the right of the robot is very deeply inset (door J in figure 4.5), and produces a high output from the HSOM, as do the posters at the end of the environment, which are made from a material that is very reflective to sonar.

However, the control trial shows that there is some similarity between the two environments, as when the robot is trained in a non-corridor control environment (the output is shown at the bottom of figure 4.7), more novelty is found. This can also be seen from the top graph of figure 4.7, since only the perceptions in the area of the door and the posters are found to be novel – the areas that are just corridor are normal. However, the perceptions of the doors and posters are found to be very novel, and there are more of these perceptions than there are in environment B.

4.3.3 Using Dishabituation

Forgetting, or dishabituation, was described in section 3.4.1. By allowing synapses to dishabituate when the neurons that they are connected to do not fire, stimuli that are seen only infrequently will always be considered to be novel. This means that the network can be made robust to features that appear occasionally in the training set, even though they should be considered novel. An example of this in the mobile robot experiments described here would be a person walking past the robot. This is one of the features that makes the novelty filter suitable for on-line use.

The experiments that are used to demonstrate how forgetting works parallel those of the previous section. Environments A and B shown in figure 4.5 were used, along with environment A* (the modified version of environment A with the door on the

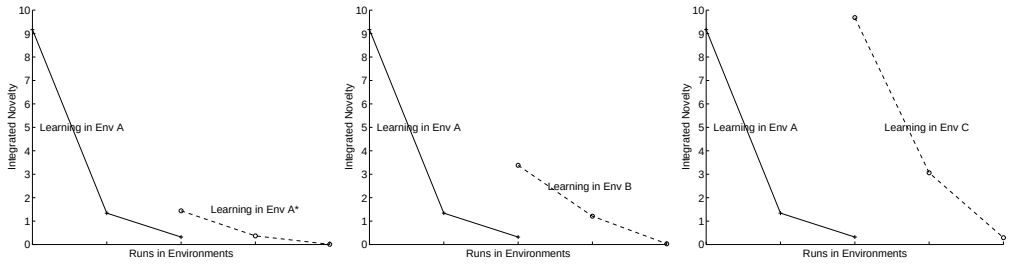


Figure 4.8: Graphs showing how the amount of novelty in an environment decreases as the novelty filter learns over three runs in an environment. The graph on the left shows the robot learning first in environment A and then in environment A*. It can be seen that once the robot has learnt about environment A, very little in A* is novel. Figure 4.6 shows that what is novel is the area around the (now open) door. The middle graph shows how the amount of novelty increases when the robot explores environment B after learning about environment A, and finally the graph on the right shows how the novelty increases when the robot explores environment C.

right hand side of the robot, door D, being open).

For each of the experiments the novelty filter was first trained in environment A, until it explored this environment without finding any novelty, as was shown on the left of figure 4.6. From this starting point two experiments were performed. In the first, the novelty filter was trained in environment A* with forgetting turned on, i.e., synapses that were not used dishabituated a little (a value of $\tau = 100$ was used for the dishabituation, this compares with $\tau = 3.33$ for the winning node). After each training run in environment A*, the robot was returned to environment A, which it explored without learning. This enabled the novelty filter to evaluate the perceptions of the closed door in environment A that it had learnt about previously. As these perceptions had not been seen during this training, they should be forgotten. The results are shown on the left of figure 4.9, and the cumulative novelty found in each environment is shown on the left of figure 4.10.

Passes 2, 4 and 6 in figure 4.9 show the standard learning of the novelty filter as it explores an environment. This is the same as was seen on the right of figure 4.6. Passes 1, 3 and 5 demonstrate the effects of the forgetting. In pass 1 the filter did not find door A to be novel, as it had just learnt to recognise it. However, during the learning runs shown in figure 4.9, the robot did not perceive a closed door on its right, and so the synapses that responded to this perception dishabituated. This means that even though the nodes in the HSOM that recognise the perception still fire strongly, as they match the input well, the habituation synapse has increased in strength again and so the novelty filter considers the perceptions to be novel. The left of figure 4.10

shows the amount of novelty detected over each of the two environments, plotted as the network learns about environment A* and consequently forgetting about the closed doorway that was only perceived in environment A.

On the right of figures 4.9 and 4.10 a second experiment is shown. In this experiment the novelty filter trained in environment A learns about environment B, so that it forgets about any perceptions of environment A that are not also seen in environment B. As environments A and B are very similar, very few things should be forgotten about. Passes 1, 3 and 5 show ordinary learning of environment B, whereas passes 2, 4 and 6 show similar effect to those shown on the left of the figure for environment A*. It is the doorway that is found to be novel, and also the crack in the wall (the first spike in pass 6). This tallies with what was found to be novel in the previous section without forgetting (shown on the right of figure 4.6) when the novelty filter trained in environment A was exposed to environment B. It is interesting to note that in pass 4 nothing is found to be novel, which was not true when the robot learned about environment A*. This is probably because there are similar perceptions in environment B for almost all of the perceptions in environment A, and so the synapses did not dishabituate.

4.4 Scalability of the HSOM

A number of problems with the HSOM were discussed in section 3.5. The most important was that it was possible for the network to saturate, meaning that any perception seen by the novelty filter would be considered to be normal, no matter what it was.

In order to demonstrate this effect clearly, data was collected from a much larger environment than those used in the previous sections, which consisted of just 10 m of corridor. The new environment was a 200 m loop of corridor, one of the two loops that comprise the second floor of the Computer Science department at the University of Manchester. Again, the Nomad 200 robot shown in figure 4.4 was used, and the same wall-following program using the infra-red sensors controlled the travel of the robot. Just as in the previous experiments, the robot took continuous sonar scans of the environment and after every 10 cm produced an input vector for the novelty filter made up of the average readings over that 10 cm of travel. This was then classified by the HSOM, which produced a novelty value for that perception with respect to the model learned so far.

As the aim of the experiment was to investigate whether or not the HSOM saturated or not, a test for saturation had to be devised, otherwise it is difficult to tell the

Figure 4.9: The results of the forgetting experiment. The perceptions that are not seen in the environment that the novelty filter is being trained in become novel when the robot explores environment A again. On the left, the novelty filter is exploring environment A*, and on the right it is exploring environment B.

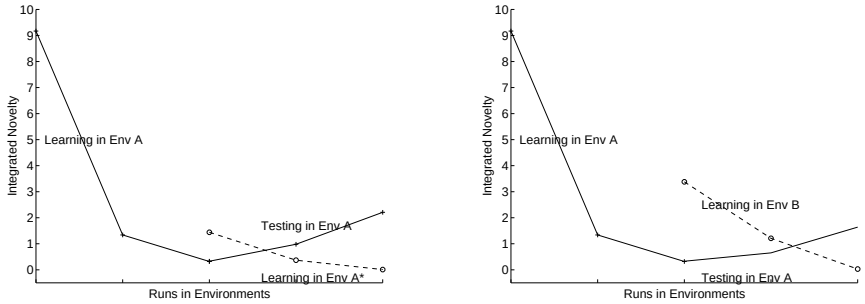


Figure 4.10: The effects of forgetting on the amount of novelty found in an environment. In the left figure the robot explores environment A for three runs, as previously. Then the robot explores environment A*, with forgetting turned on. After each exploratory pass through environment A*, the robot explores environment A. It can be seen that as the novelty filter finds environment A* less novel, so the amount of novelty found in environment A increases, as the robot forgets about these perceptions. The same is shown on the right for the robot learning about environment B.

difference between good learning and saturation. This test was that during the final exploration lap of the environment, the environment was changed significantly, so that this should be detected as novel. At the end of the final run a collection of cardboard boxes were placed in the corridor. These cardboard boxes meant that the robot reached a dead end and that the corridor was considerably narrower than previously. If the novelty filter did not find these perceptions to be novel, then clearly the network had saturated. The saturation of the network could also be checked by looking at the saved weights and checking the value of the habituation synapses. If all of the habituation synapses are low then there is no space for novel perceptions and so the network has saturated.

Two different sized HSOM novelty filters were run simultaneously by the robot as it travelled. The first, smaller HSOM comprised 49 neurons arranged in a 7×7 grid, while the second was a 23×23 network of 529 neurons. For both networks a learning rate of $\eta = 0.25$ was used and kept constant, so that the learning behaviour did not change over time. The neighbourhood was also kept fixed, being the 8 immediate neighbours of each node. The habituation values were those used in the experiments in section 4.3.2.

On the left of figure 4.11 the output of the novelty filter based on the smaller HSOM for each perception is shown over the five passes through the large environment, while on the right of the figure the output of the large HSOM is shown. In fact, to make the graph easier to read, what is shown is the maximum output of the novelty filter over each 1 m of travel. That is, the maximum of the 10 outputs of the novelty filter over

each metre.

Looking at the output of the small HSOM first (on the left of figure 4.11), it can be seen that from halfway through pass 3, nothing is found to be novel. This could be because the novelty filter had learnt a satisfactory representation of the environment; the results of the first two passes show that the HSOM was learning satisfactorily up to this stage. However, this novelty filter fails to find any of the perceptions at the end of pass 5 to be novel, which implies that the network has indeed saturated. An examination of the network weights demonstrated that this was so – all of the 49 synapses in the network were fully habituated.

The right of figure 4.11 shows the output of the novelty filter based on the large HSOM for exactly the same 5 passes through the large environment. Starting at the end of pass 5, it can be seen that the perceptions at the very end of the run are found to be novel, which is satisfying. Unfortunately a lot of other perceptions are also found to be novel, too. This is disappointing. Many of the perceptions that are found to be novel are places where the robot is turning a corner. This is because the wall-following control program allows the robot to take a variety of paths around a corner, and so the same part of the environment can appear different each time.

However, even taking this into consideration, the large HSOM network does seem to have some problems with noise in the sensors. This is partially to be expected since the network is so large, so that there is enough space in the network for each perception to be stored separately, so that the network does not generalise. In fact, 271 of the 529 synapses had received some habituation, but many only as neighbours of nodes. Nearly half of the nodes had never been activated, nor had any of their neighbours.

Using the data collected during the experiment, a number of different sizes of HSOM were also trained off-line, and different neighbourhood sizes and learning rates were tried. It was found that a 15×15 network with a neighbourhood of the nearest 24 nodes performed well on this dataset. This network did not saturate, as it found the boxes at the end of the run novel, but it learnt satisfactorily and no other perceptions in the final run were found to be novel.

Unfortunately there does not seem to be any reliable way of predicting how large the network should be in advance of collecting the data. This is because it is difficult to tell how many different perceptions will be seen. Therefore the filter cannot be used on-line, as intended, for large environments. A solution to this problem is described in the next chapter, a different form of neural network that adds new nodes into the map field as the network learns, so that the size of the network does not have to be selected

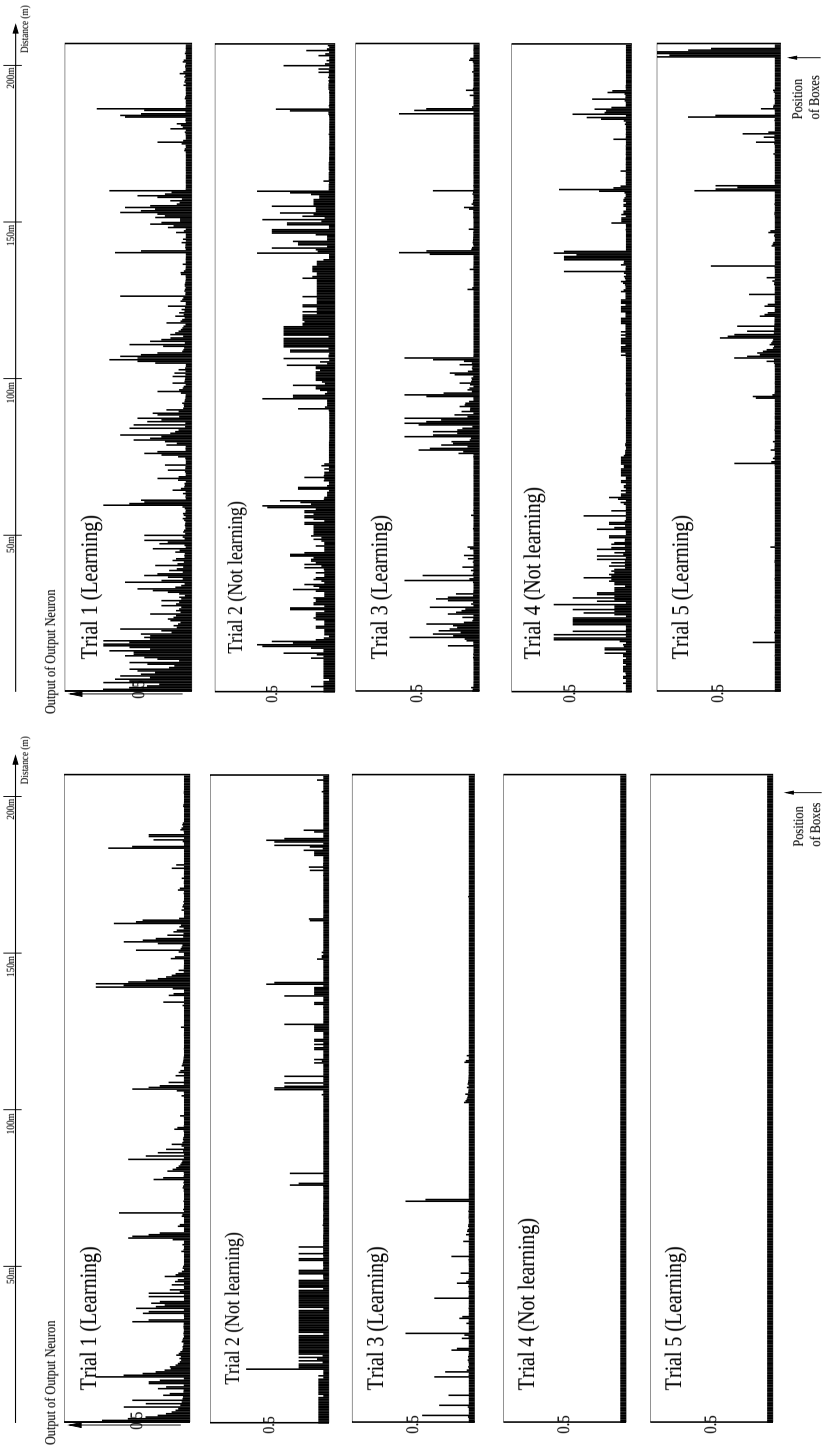


Figure 4.11: *Left:* The output of the novelty filter using the small HSOM. The network saturates so that after partway through pass 3 nothing is found to be novel. *Right:* Using the large HSOM as the basis for the novelty filter means that the network does not saturate. However, the filter does not appear to generalise well.

in advance.

4.5 Summary

In this chapter the novelty filter based on the HSOM, which was described in chapter 3, has been investigated. First of all, the effects of using the novelty filter on a simple robot to perform the task of turning to face the most novel stimulus were investigated. It was shown that this was successful, although as the capabilities of the sensors were limited, there was no evidence that the system would scale.

Then a second problem, that of inspection, was investigated. A robot explored an environment, perceiving the environment through its sonar sensors. Sonar scans taken as the robot travelled were used as the input vector to the novelty filter, after some simple preprocessing. The novelty filter evaluated the novelty of each scan with respect to the perceptions seen so far, and then added the new perception to the model.

The performance of the novelty filter on this task was investigated for a range of corridor environments in the Computer Science Department of the University of Manchester. It was shown that the features that were highlighted when the robot explored one environment after being trained in a different one were those features that were not present in the training environment, or were seen only rarely. Once the concept had been shown to work, the effects of using dishabituation, or forgetting, were demonstrated. This was done by training the robot in an environment until none of the perceptions were novel, and then showing that if the robot was then trained in an environment where some feature was missing then the perceptions of the missing features were found to be novel when the robot explored the first environment again.

Finally in this chapter, the problem of scalability to larger environments was considered. It was demonstrated that the HSOM network could saturate if the size of the network was small and the environment large. This is a problem with using the HSOM network for on-line learning, as the network is not suitable for continuous learning. It has been shown that the network works well for the task, providing that the size of the network is chosen carefully. However, it would be nice if this investigation was not required, so that the novelty filter could be used on-line as originally intended. A solution to this problem is the subject of the next chapter.

Chapter 5

The 'Grow When Required' (GWR) Network

A solution to the problems described at the end of the previous chapter is introduced here. Details of the new network are given, and the behaviour of the network is analysed on a number of non-novelty detection tasks.

5.1 Introduction

This chapter describes the 'Grow When Required' (GWR) network that is proposed to solve the problems of scalability demonstrated at the end of the previous chapter. This scalability problem exists for any algorithm where the number of nodes in the network is fixed before learning begins. The principal reasons for this are that the single fixed point of the habituation dynamics occurs when the synapse is fully habituated, and that the algorithm is trained on-line, as was discussed in section 3.5.

The proposed solution is a growing network, that is, a network that adds nodes into the map space in order to reduce the mapping error over the set of inputs. A number of algorithms that use a growing process during self-organised learning have been described in the literature, and these are described in section 5.2. Most of these networks add new nodes into the map field to support the node that has accumulated the highest error during previous iterations, or to support topological structures. This means the new nodes are added only when the number of iterations is an integer multiple of some pre-defined constant, λ. This approach is not useful for novelty detection, because unless the novel input occurs just before the new node is added, the response to the input will be delayed until the novel perception is encountered a second time. Even if

the novel input happens when the network is about to add a new node, it is unlikely that one novel input alone will have sufficient impact on the accumulated error at the nearest node, and so the network will not act as a novelty filter.

The GWR network avoids this problem by adding new nodes into the network structure whenever the activity of the current best-matching node is below some threshold, which implies that the best-matching node is not trained to deal with that particular input. This means that the network grows very quickly when new data is presented, but stops growing once the network has matched the data to some definable accuracy. The GWR algorithm is described in section 5.4.

Some analysis of the algorithm is given in section 5.5, in particular two cost functions are described that aim to evaluate the performance of the network. Two separate cost functions are used because the network is attempting to minimise two different measures – the network should model the data well, so that the distance between each input and the best-matching node should be small, but equally the network should be as parsimonious as possible, so that the number of nodes that are added is kept small. This means that the computational costs are kept to a minimum. These cost functions are evaluated in section 5.6 on two different test problems and the results are compared to the Growing Neural Gas (GNG) network that is closely related to the GWR network. Two further datasets are also introduced in order to demonstrate that the GWR network can produce *perfectly topology-preserving networks* (Martinetz, 1993), where the dimensionality of the map space models that of the data manifold. Demonstrations of the GWR network performing novelty detection tasks, together with comparison data for the GNG and Reduced Coulomb Energy (RCE) networks, are given in chapter 6.

5.2 Literature on Growing Neural Networks

The Cascade-Correlation Learning Architecture (CCLA) (Fahlman and Lebiere, 1990) is a supervised learning network that aims to speed up learning substantially. Like most of the growing networks described in this section, CCLA starts with a minimal network and adds units into the network architecture, but unlike most of the networks described here, CCLA is a supervised network and nodes are added into the hidden layers of the network. The new units are intended to act as feature detectors and are added when no error reduction has occurred over several training iterations. The candidate unit is created and its input weights are trained to maximise the correlation between the output of the node and the residual output error before the node is added to the network. Once

the node is added, these weights are frozen.

One of the first unsupervised growing neural networks was the Growing Cell Structures (GCS) network of Bernd Fritzke (1994). In the GCS, the network is made out of k-dimensional simplices. The value of k is pre-defined, and is usually $k = 2$, so that the simplices are triangles and the map is two dimensional. A new node is inserted every λ iterations, where λ is a positive constant, with the new node positioned to support the node that has accumulated the highest error during previous steps. The network continues to adapt and grow until some stopping criterion is met. This can take the form of a prescribed network size or some acceptable minimum for the accumulated error in the network.

The GCS is one of a number of networks that are perfectly topology-preserving (Martinetz and Schulten, 1994), in that if k is the dimension of the manifold on which the data lies, the positions of the map nodes conform to the topology of the input space. Another network with this property is the Dynamic Cell Structures network (Bruske and Sommer, 1995), which adds nodes to maintain the topology preservation rather than to equalise the expected error at each node, as in the GCS. A different approach is the Growing Self-Organising Map (Bauer and Villmann, 1997; Villmann and Bauer, 1998), which attempts to reproduce the topology of the input space by adding complete rows or columns that maintain the rectilinear structure of the SOM architecture. The map field can also be extended into new dimensions. The choice of which of these steps to take at each stage depends upon the dynamics of the Voronoi cells in the network.

In addition to the GCS, Fritzke (1995) has also proposed the Growing Neural Gas (GNG) network. As in the GCS, new nodes are added every λ iterations to support the node with the highest accumulated error, but in the GNG the structure of the network is not constrained, with links being created between the two nodes with the highest activity for each input. For each data sample presented to the network, the two best-matching nodes are selected, that is the two nodes whose weights are closest to the input in the Euclidean sense. A neighbourhood connection is made between the two nodes if it does not already exist, and the positions of these nodes – together with the other neighbours of the winning node - are moved so that their weights better match the input. Edges that are not used increase in age, while ages that are used have their age reset to zero. Once the age of an edge exceeds a threshold, that edge is deleted. This concept of edge creation is known as competitive Hebbian learning and was developed by Martinetz and Schulten (1991). After λ iterations, the node that has accumulated

the highest error during the previous steps is calculated, and a new node is added to support it. The new node is positioned between the node with the highest error and whichever of its neighbours has the next highest error. The algorithm continues until some stopping criterion is reached.

Fritzke (1997) has also investigated modifying the GNG so that it can be applied to non-stationary datasets. This is done by adding the concept of the 'utility' of a node, the amount that the global error would increase if that node were removed. Nodes with low utility are removed by the algorithm, which is known as the Growing Neural Gas with Utility (GNGU).

A number of authors have proposed variations on the GCS or GNG. For instance, Burzevski and Mohan (1996) propose a hierarchy of GCS networks arranged in a tree to avoid the massive changes that deleting a node can have on the structure of the GCS network due to the constraint that the network must form k-dimensional simplices. Vlassis et al. (1997) reset the GCS into a probabilistic framework, which improves the way the network handles correlated inputs, and Cheng and Zell (2000) allow for more than one node to be inserted at each insertion step to increase the rate of convergence of the network.

A different form of growing network grows new nodes without generating neighbourhood relations between the nodes, to perform categorisation. The simplest of these is the unsupervised RCE network (Reilly et al., 1982; Kurz, 1996), which uses prototype vectors to describe particular classes. If none of the prototype vectors already in the map space is sufficiently close to the current input, a new class is generated and the input used as the prototype for that cluster. This network was described in section 2.7.3. Different examples of this type of network, although rather more complicated, are the Adaptive Resonance Theory networks (ART) (Carpenter and Grossberg, 1988). These networks (described in section 2.7.2) also add new categories when a mismatch is found between the current input and the set of categories in the network. The degree of mismatch allowed, which controls how many categories are made in the network for a given dataset, is controlled by a parameter known as the vigilance.

Another approach is the Contextual Layered Associative Memory (CLAM) of Thacker and Mayhew (1990), which uses a multi-layered network that has feedback between the layers and resonance within a layer. Patterns are classified over a group of nodes rather than using a winner-takes-all approach. Nodes are added when a probability measure indicates that the region of input space that the current input comes from has low density.

5.3 The Need For a New Growing Network

A neural network suitable to be the basis of an on-line novelty filter requires several properties. Primarily, the network needs to be able to learn continuously, so that if new data is provided the network can learn about it without needing to be reinitialised and retrained. This retraining would mean that all data had to be stored externally and that the network could not learn on-line. In order to learn continuously the algorithm must have no time-dependent parameters. This was the problem with the HSOM. The SOM is not designed for continuous learning and continually reduces the neighbourhood size and learning rate over time. In the HSOM these parameters were kept constant, which was what enabled the HSOM to saturate, as was discussed in section 3.5.

Continuous learning also has implications for the stability-plasticity dilemma. If a network is to learn continuously then the network has to remain plastic, so that new memories can be added. However, old memories must be recalled reliably and not overwritten by new, i.e., the memories must be stable. The way to accommodate these requirements is to use an algorithm that allows nodes to be added into the network architecture. A number of networks have been introduced in the literature that add nodes into the network. Examples include the Cascade-Correlation algorithm, CLAM, ART and the Growing Cell Structures and Growing Neural Gas algorithms, all of which were described in section 5.2.

Novelty detection algorithms have to be able to deal with data for which targets are not available. This means that supervised methods are not suitable, which rules out the Cascade-Correlation algorithm. In addition, the novelty filter described in this work, which is based on a set of habituating synapses, uses the concept of neighbourhood connections so that similar perceptions can habituate each other. This requires that the mapping generated by the network is topology-preserving, so that similar perceptions are mapped onto network nodes that are close together. The ART and CLAM networks do not satisfy this constraint.

While the GNG and GCS networks that were discussed in the previous section are suitable for continuous self-organised learning, they are not suitable for use in the novelty filter for one important reason – the networks add nodes into the map field in such a position as to support the node that has accumulated the highest error during the previous λ iterations, where λ is a positive constant. This means that the number of nodes that the network adds, and when these nodes are added, is controlled by the parameter λ, and not by the data that is presented. A novelty detector has to react as soon as a novel input is presented, and these networks do not allow that. These

considerations motivated the design of the 'Grow When Required' network that is described in the next section.

5.4 The GWR Network

5.4.1 A Description of the Algorithm

This section describes the algorithm used to implement the GWR network. The network is designed to fill the gap in the growing neural network literature that was described above – responding quickly to new data, but maintaining a topologically correct set of neighbourhood connections in the map space. There are therefore two processes being performed by the algorithm – adding and repositioning nodes in the map space, and generating neighbourhood connections between network nodes.

The technique used for creating and destroying network edges is the competitive Hebbian learning method used by Martinetz and Schulten (1991) and Fritzke (1995). For each input an edge connection is generated between the node that best matched the unit and the second best-matching unit. These edge connections have an associated 'age'. This is originally set to zero, and is reset to zero whenever the two nodes that it connects are the best match and second best match. If one of the two nodes is the best match, but the other is not the second best, then the age of the edge is incremented. Edges whose age exceeds some constant $a_{\max}$ are removed. Any node that has no neighbours, i.e., that has no edge connections, is removed, as it is an unused ('dead') node.

The new part of the algorithm is the way that the growing process is carried out. Rather than adding a new node after every λ inputs, new nodes can be added at any time. For example, several may be added one after another and then no more added for the next hundred iterations. The position of a new node is dependent on the input and the current winning node, rather than being added where the accumulated error is highest, as in Fritzke's algorithm.

A new node is added when the activity of the best-matching node (which is a function of the distance between the weights of the node and the input, see equation 5.6) is not sufficiently high. The activity of nodes is calculated using the Euclidean distance between the weights for the node and the input. It is assumed that the input vector lies on the unit hypersphere. To allow for the fact that recently created nodes may not yet have been trained to match their intended output correctly, which would mean that the

node should be trained more rather than creating a new node, each node is equipped with a way of measuring how often it has fired. This could be done using a simple counter for each node, which is incremented whenever that node is the best match. However, habituation, which is the basis for the novelty detection, provides an alternative. The habituation synapses connecting each node to the output decrease in strength as the node that they are associated with fires or – to a lesser extent – whenever a node with a neighbourhood connection to that node fires. This enables the growing process to be combined with the novelty detection process. It also means that the neighbours of a node, which are trained along with the winning node, also have their counters changed a little.

Networks that learn continuously can have a problem that the weights of well-trained nodes continue to move slightly. Habituation also provides a simple measure of when to freeze the weights, if that is desirable. If a node is fully habituated then the node is presumably well trained, and so the weights of that node can be frozen.

So when an input is presented to the network, the activity of each node in the map space is calculated and a winner picked. If this best-matching node represents the input well then the activity of that node will be close to one (calculated using equation 5.6 on page 120). In that case the best-matching node is trained a little, as are its neighbours. However, if the activity of the network is below the insertion threshold a_T, then either the node has only recently been added to the map and is still being trained, or there is a mismatch between the node and the input. If the node is a new one then the habituation synapse that links it to the output node will be high, and so the node is trained a little (and the habituation synapse decreases in strength, as it did with the HSOM). Otherwise, a new node is needed to represent the input better. This node is added between the winning node, which caused the problem, and the input; with the weights of the new node being initialised to be the mean average of the weights for the best-matching node and the input. This method of node generation, and particularly the insertion threshold a_T, can be thought of as tunable generalisation, that is, the amount to which the network generalises between similar perceptions is controlled by the amount of discrepancy between perceptions that triggers a new node.

In addition to the insertion threshold, a threshold is also required to decide at what level of habituation an input is considered to be sufficiently trained, so that low activity signifies a mismatch. In practise the value of this threshold does not seem to affect the behaviour of the network significantly. Using Stanley's habituation equation (reproduced as equation 5.14 below) with the values of the parameters reported in

section 4.3.2, ($y_0 = 1$, $\alpha = 1.05$ and $\tau = 3.33$ for the winning node, $\tau = 14.3$ for the neighbours, $S(t) = 1$ for the winning node) the threshold was set so that if the node had fired five times then it was considered to be trained.

The value of the insertion threshold a_T does make a large difference, however. If the value is set very close to 1 then more nodes are produced and the input is represented very well. For lower values of a_T fewer nodes are added. The effects of changing the parameter are investigated in section 5.6.

The precise steps of the algorithm are now described. In order to facilitate comparisons, the new algorithm is described using notation based on that used by Fritzke (1995) to describe the GNG.

5.4.2 The GWR Algorithm

Let A be the set of map nodes, and $C \subset A \times A$ be the set of connections between nodes in the map field. Let the input distribution be $p(\xi)$, for inputs ξ. Define w_c as the weight vector of node c.

Initialisation

The network is initialised using:

- Create two nodes for the set A,

$$A = \{c_1, c_2\} \tag{5.1}$$

  with their weights w_{c_1}, w_{c_2} initialised randomly from $p(\xi)$

- Define C, the connection set, to be the empty set,

$$C = \emptyset \tag{5.2}$$

- Let $y(0) = y_0$

Iteration

Each iteration of the algorithm follows the following steps:

1. Generate a data sample ξ for input to the network

2. For each node in the network, calculate the distance from the input $\|\xi - w_i\|$

3. Select the best matching node, and the second best, that is, the nodes $s,t \in A$ such that

$$s = \arg\min_{c \in A} \|\xi - w_c\| \tag{5.3}$$

and

$$t = \arg\min_{c \in A/\{s\}} \|\xi - w_c\| \tag{5.4}$$

where w_c is the weight vector of node c.

4. If there is not a connection between s and t, create it with age 0

$$C = C \cup \{(s,t)\}, \tag{5.5}$$

otherwise, set the age of the connection to 0.

5. Calculate the activity a of the best matching unit (s),

$$a = \exp(-\|\xi - w_s\|) \tag{5.6}$$

6. If we should add a node, i.e., if activity $a <$ activity threshold a_T and habituation $<$ habituation threshold h_T

   - Add the new node, r

$$A = A \cup \{r\}. \tag{5.7}$$

   - Create the new weight vector, setting the weights to be the average of the weights for the best matching node and the input vector

$$w_r = (w_s + \xi)/2. \tag{5.8}$$

   - Insert edges between r and s and between r and t

$$C = C \cup \{(r,s),(r,t)\}, \tag{5.9}$$

   - Remove the link between s and t

$$C = C/\{(s,t)\} \tag{5.10}$$

7. Adapt the positions of the winning node and its neighbours, i, that is the nodes to which it is connected.

$$\Delta w_s = \varepsilon_b(\xi - w_s) \tag{5.11}$$

$$\Delta w_i = \varepsilon_n(\xi - w_i) \tag{5.12}$$

where the learning rates are such that $0 < \varepsilon_n < \varepsilon_b < 1$

8. Age edges with an end at s.

$$age_{(s,i)} = age_{(s,i)} + 1. \tag{5.13}$$

9. Habituate the winning node and its neighbours using

$$\tau\frac{dy(t)}{dt} = \alpha[y_0 - y(t)] - S(t), \tag{5.14}$$

where $y(t)$ is the strength of the synapse, y_0 is the initial strength, and $S(t)$ is the stimulus strength, usually 1. α and τ are constants controlling the behaviour of the curve. The winner habituates faster than its neighbours. Values of the parameters used in the experiments in section 5.6 are $\alpha = 1.05$, $y_0 = 1$ and $\tau = 3.33$ for the winning node, $\tau = 14.3$ for the neighbours.

10. Check if there are any nodes or edges to delete, i.e., if there are any nodes that no longer have any neighbours, or edges whose age is greater than $a_{\max}$.

5.5 Analysis of the Network

This section discusses the properties of the self-organisation and learning processes taking place as the GWR network learns. Section 5.5.1 introduces the notation that is used in the discussion, and then neighbourhood preservation is discussed in section 5.5.2 and three different measures that have been proposed in the literature to measure it are described. Neighbourhood preservation is not the only way to evaluate the performance of the mapping learnt by a network. The accuracy with which the network models the data and the length of the neighbourhood connections are also possible measurements. Measures of these properties are described in section 5.5.3. These different measures are evaluated when the GWR network learns representations of a number of different datasets in section 5.6.

5.5.1 Notation

This section introduces the notation that is generally used in the literature on self-organisation and topology preservation.

A network A comprises a number N of nodes and receives inputs sampled from a data manifold $M \subseteq \mathbb{R}^d$. Every node i in A has a synaptic weight vector $w_i \in \mathbb{R}^d$. The representation of M formed in A is defined by the mappings $\mathcal{M}_A = (\Psi_{A\to M}, \Psi_{M\to A})$, the mapping from M to A and its inverse, which are defined by:

$$\mathcal{M}_A = \begin{cases} \Psi_{M\to A} : M \to A; v \in M \mapsto i^*(v) \in A \\ \Psi_{A\to M} : A \to M; i \in A \mapsto w_i \in M \end{cases}, \tag{5.15}$$

where $i^*(v)$ is the neural unit with weight vector $w_{i^*(v)}$ closest to v. A connection matrix C is defined on the network A by putting non-zero matrix entries between nodes that are connected in the network, i.e.,

$$C_{ij} = \begin{cases} 1 & \text{if } i \text{ and } j \text{ are connected} \\ 0 & \text{otherwise} \end{cases}. \tag{5.16}$$

5.5.2 Measuring Neighbourhood Preservation

The preservation of neighbourhood relations (also known as topology preservation) is a very useful property of self-organising networks and has attracted a great deal of interest. Particularly useful papers are (Martinetz and Schulten, 1994; Villmann et al., 1997; Goodhill and Sejnowski, 1997).

Loosely speaking, a mapping preserves neighbourhood relations if nearby points in input space remain close in the map space. This has been formalised by Martinetz (1993) through the definition of the perfectly topology-preserving map. A mapping between input manifold and network is perfectly topology-preserving if connected nodes i, j that are adjacent in A have weight vectors w_i, w_j adjacent in M.

In general, a network can only perform a perfectly topology-preserving mapping if the dimensionality of the map space reflects the dimensionality (or at least, the *intrinsic dimensionality*) of the input space. This is demonstrated in figure 5.1. At the bottom of the figure, three different square-shaped manifolds M are shown. Only in figure (b), where the dimensionality of the map space and input space are the same, can a perfectly topology-preserving map be generated between M and A. For this reason, for a network to be perfectly topology-preserving it is necessary for the network to evolve to reflect the dimensionality of the dataset, or to have this dimensionality preset.

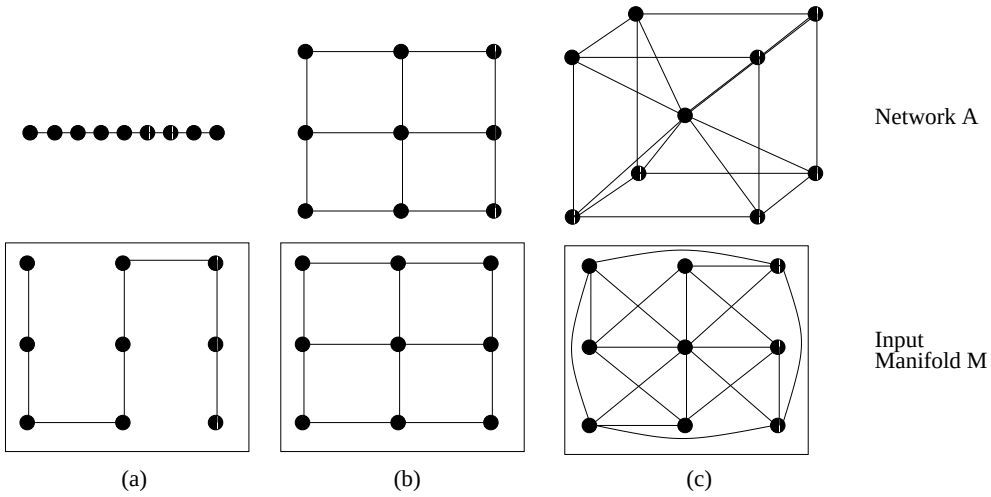


Figure 5.1: Networks A of three different dimensionalities receive mappings from a square-shaped manifold M. Only in (b) where the dimensionality of the manifold M and map A are the same is a neighbourhood preserving map generated. Adapted from Martinetz and Schulten (1994).

The question of how topology preservation can be measured has received a lot of attention. Several authors have described ways of quantifying neighbourhood preservation. These can be categorised into two categories, measures of similarity and measures of similarity ordering. In the first class are those measures that evaluate the similarity of pairs of points before and after the neighbourhood mapping, and require that the two similarity measures are at least correlated, while in the second class it is only required that the relative ordering of the similarities is preserved. A useful review is given by Goodhill and Sejnowski (1997). Some of the more relevant measures are described below.

The C Measure (Goodhill and Sejnowski, 1997)

The cost function C is a member of the first class, requiring that the two similarity measures are at least correlated. It is defined as

$$C = \sum_{i=1}^{N} \sum_{j<i} F(i,j) G(\Psi_{M\to A}(i), \Psi_{M\to A}(j)), \tag{5.17}$$

where $\Psi_{M\to A}$ is the mapping from the input space M to the map space A, and F and G are symmetric similarity functions defined on M and A respectively, that return a scalar indicating how similar two points i and j are. For Euclidean spaces these could

be Euclidean distances.

The Topographic Product P (Bauer and Pawelzik, 1992)

The topographic product measures the neighbourhood preservation of a SOM learning a mapping between spaces of possibly different dimensionality.

For each node in the map i, two sequences are measured, $n_k^A(i)$, the kth neighbour of i measured using the Euclidean distance in the map space A, and $n_k^M(i)$, the kth neighbour of i measured using the Euclidean distance in the input space M (i.e., the distance between the weights $\mathbf{w}_i$ and $\mathbf{w}_{n_k^M(i)}$). The sequences are used to define the quantity P_3,

$$P_3(i,k) = \left(\prod_{l=1}^{k} \frac{d^M\left(\mathbf{w}_i, \mathbf{w}_{n_l^A(i)}\right)}{d^M\left(\mathbf{w}_i, \mathbf{w}_{n_l^M(i)}\right)} \cdot \frac{d^A\left(i, n_l^A(i)\right)}{d^A\left(i, n_l^M(i)\right)} \right)^{\frac{1}{2k}}, \tag{5.18}$$

where d^A is the Euclidean distance in A and d^M is the Euclidean distance in M. The logarithm of P_3 is then averaged over the neighbourhood orders k and nodes i to compute the topographic product P (where N is the number of nodes in A),

$$P = \frac{1}{N(N-1)} \sum_i \sum_{k=1}^{N-1} \log(P_3(i,k)). \tag{5.19}$$

If P is close to zero then the dimensionality of the map space is correct, while if $P < 0$, the dimensionality of the map space is too low, and vice versa for $P > 0$.

The Topographic Function Φ_A^M (Villmann et al., 1997)

The topographic product is limited to linear data manifolds, as the neighbourhood relations are measured using the Euclidean metric within the embedding space of the weight vectors. A way around this problem is proposed by the topographic function, which evaluates the topology preservation of the SOM mapping taking the structure of the data manifold into account using the Delaunay triangulation induced onto it by the mapping. The topographic function Φ_A^m of map $\mathcal{M}_\mathcal{A}$ is defined by

$$\Phi_A^M(k) = \begin{cases} \frac{1}{N}\sum_{j\in A} f_j(k) & k < 0 \\ \Phi_A^M(1) + \Phi_A^M(-1) & k = 0 \\ \frac{1}{N}\sum_{j\in A} f_j(k) & k > 0. \end{cases} \tag{5.20}$$

where $f_i(k)$ measures the neighbourhood preservation of the mapping $\Psi_{M\to A}$, and $f_i(-k)$ the neighbourhood preservation of the mapping $\Psi_{A\to M}$. These two quantities can be computed using the induced Delaunay triangulation, that is, the graph connecting points with adjacent Voronoi polyhedra (for further details, see Martinetz and Schulten (1994) and the discussion below). $\Phi_A^M(0) = 0$ if and only if the map is perfectly topology-preserving.

Discussion

In the forms described above, both the topographic product and the topographic function are only specified for rectangular lattices. This means that they are not applicable to the GWR network and other networks where the network structure is not of this form. For the topographic function the problem is in the measurements of $f_i(k)$ and $f_i(-k)$. A description of how these measurements can be made more general is given in Villmann et al. (1997), and is developed further here.

The structure of A is defined by the connectivity graph C that is generated by the competitive Hebbian rule. A discrete topology (Berge, 1997) can be induced on this space using the graph metric in $C(i)$, where $C(i)$ is C with node i taken as the root. A second discrete topology can be induced on A by considering the graph metric of the Delaunay graph, $\mathcal{D}(i)$, again with node i taken as the root. These two topologies are referred to as $\mathcal{T}_A^-(i)$ and $\mathcal{T}_A^+(i)$ respectively.

A neighbourhood topology also needs to be induced on the data manifold M, or at least that subset of it $M^A = \{w_i \in \mathbb{R}^d \,|\, i \in A\}$. By generating the Voronoi diagram of M using M^A and constructing the dual Delaunay graph of this $\mathcal{D}$, the required topology (labelled $\mathcal{T}_{M^A}$) is induced.

Thus, three discrete topological spaces have been created, two on A - $(A, T_A^-(i))$ and $(A, T_A^+(i))$, and one on M^A, $(M^A, T_{M^A}(i))$. The map $\mathcal{M}_A = (\Psi_{A\to M}, \Psi_{M\to A})$ can then be defined as topology preserving if the maps $\Psi_{M\to A}$ and $\Psi_{A\to M}$ are both continuous mappings for all nodes $i \in A$ on their respective topological spaces,

$$\Psi_{M\to A} : (M^A, \mathcal{T}_{M^A}(i)) \to (A, \mathcal{T}_A^-(i)) \tag{5.21}$$

$$\Psi_{A\to M} : (A, \mathcal{T}_A^+(i)) \to (M^A, \mathcal{T}_{M^A}(i)). \tag{5.22}$$

Using these relations, general forms for $f_i(k)$ (which measures the continuity and hence the neighbourhood preservation of $\Psi_{M\to A}$) and $f_i(-k)$ (which does the same for

$\Psi_{A\to M}$) can be derived, and are given below:

$$f_i(k) = \#\left\{ j | d_{\mathcal{T}_A^-(i)}(i,j) > k; d_{\mathcal{T}_{MA}(i)}(i,j) = 1 \right\} \tag{5.23}$$

$$f_i(-k) = \#\left\{ j | d_{\mathcal{T}_A^+(i)}(i,j) = 1; d_{\mathcal{T}_{MA}(i)}(i,j) > K \right\} \tag{5.24}$$

where $\#\{\cdot\}$ is the cardinality of the set, $k = 1, \cdots, N-1$ and $d_{\mathcal{T}(i)}(i,j)$ is the distance metric based on each of the topologies. These measurements can then be used in equation 5.20 and the topology preservation of the mappings learnt by the GWR network measured.

Evaluating the measures is a computationally expensive task, as the Delaunay triangulation has to be computed and then the graph of connections between datapoints has to be created and searched. The topographic function has been evaluated for some of the datasets that are described in section 5.6, and the results are given there.

5.5.3 Further Performance Measurements

The measurements described in the previous section aimed to evaluate the topology preservation of self-organising networks. This is only one of the properties that a network mapping should display. In this section two complementary cost measures are described that aim to evaluate the mapping between input space and map space generated by the algorithm. Any number of cost measures that evaluate the desired properties can be generated; these particular ones were chosen for their simplicity. The two measures represent a trade-off between a complex network with many nodes that represents all possible inputs very accurately and an efficient network that generalises well.

The network should be as parsimonious as possible, meaning that the length of edges should be short and the number of nodes reasonably small, but equally the network of nodes must accurately model the data, so that the distance between an input and the node that best represents it should be small. The cost measures evaluate each of these aims separately. The first measure, E_1, given in equation 5.25, penalises the network for neighbourhood connections between nodes that are placed far apart on the graph.

$$E_1 = \sum_i \sum_{j<i} C_{ij} G(v_i, v_j), \tag{5.25}$$

where the sum is over all the nodes in the network, and C is the connection matrix defined in equation 5.16. For the implementation used in the next section, $G(v_i, v_j) = \|w_i - w_j\|^2$, the Euclidean distance. The second measure, E_2, given by equation 5.26, shows how the network aims to minimise the distance between each data point d_μ and the weight vector w of the node v that best represents it.

$$E_2 = \sum_\mu \sum_i G(d_\mu, v_\mu) \cdot f_i(d_\mu, \{v\}), \tag{5.26}$$

where the first sum is over each element of the dataset, and

$$f_i(d, \{v\}) = \frac{e^{-\lambda G(d, v_i)}}{\sum_j e^{-\lambda G(d, v_j)}}. \tag{5.27}$$

In the limit as $\lambda \to \infty$, this reduces to winner-takes-all, which is the implementation that is used in the results reported in the next section.

5.6 Demonstrations of the GWR Network

This section demonstrates the learning ability of the GWR network on four separate datasets. For two of the problems the performance is compared to the Growing Neural Gas (GNG) network, which was described in section 5.2. The GNG is the network that is most similar to the GWR network.

5.6.1 The Four Squares Dataset

The first dataset is very simple. Data is drawn randomly with uniform distribution from the unit square, with inputs coming with non-zero probability from four squares and a line joining two of the squares. This is shown in the top left picture of figure 5.2. The remaining seven pictures of figure 5.2 show the appearance of the network after increasing number of data presentations when the GWR network is trained on this dataset. The points plotted are the positions of the weights vectors for each node, together with lines linking nodes that are neighbours.

The figure shows that the network initially grows very quickly, after 80 samples from the input space have been presented to the network (the second picture of figure 5.2), 40 nodes have been added to the network. From this start, only three further nodes are added during the next 40 data presentations, and one node during the remainder of the learning process. This is markedly different to other growing networks,

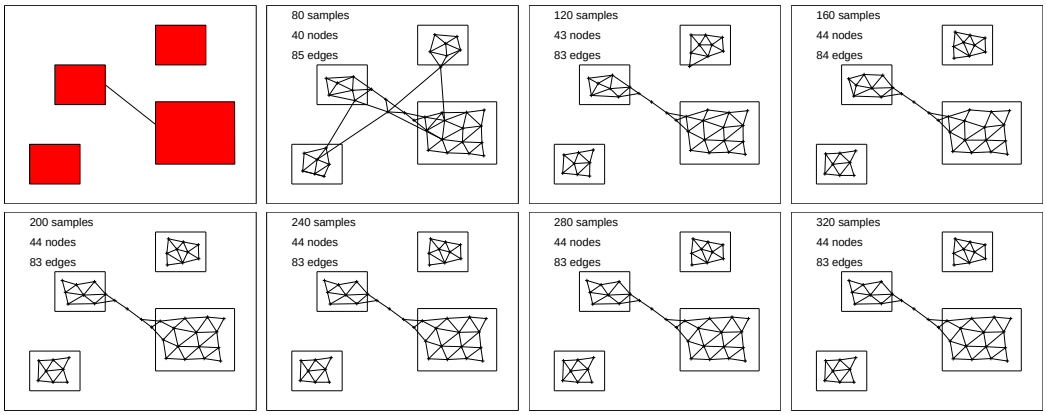


Figure 5.2: The Grow When Required Network learns a dataset consisting of four squares and a line connecting two of the squares. The top left diagram shows the layout of the dataset, the grey squares being the areas from which data can be sampled with non-negligible probability. The other diagrams show the positioning of nodes and edges while the network learns, with insertion threshold $a_T = 0.99$. Initially the network consists of two nodes placed at random within the space. It can be seen that where the input is two-dimensional, the structure of the network is two-dimensional, and where the input is one-dimensional, the network reflects this.

which would have grown at the same constant speed throughout this training. The result is that from very early on in training the only changes that occur in the network structure are that the weights of the nodes are adjusted to form a better representation of the space, and additional neighbourhood links (edges) that were created by the growing process are pruned. After 200 samples from the dataset the network has learnt an accurate representation of the input space and the network does not change again.

Figure 5.2 also demonstrates that the GWR network is perfectly topology-preserving. The dataset has two separate dimensionalities – the squares are obviously two-dimensional, but the line joining the top left and bottom right squares is one-dimensional. The neighbourhood structure of the trained network represents this, with a string of neurons representing the line and a lattice of nodes representing the squares. This is reflected in figure 5.3, which plots the values of the two cost functions described in section 5.5 for both the GWR network and the GNG network.

The first cost function (shown on the left of figure 5.3) measures the length of the connections between nodes. As nodes are added to the map this value increases for both the GWR and GNG networks, but for the GWR it decreases to a constant value when the network stabilises, while for the GNG network it decreases a little as further nodes are added. The second function evaluates the distance between each datapoint and the corresponding best-matching node and shows similar results – the mapping

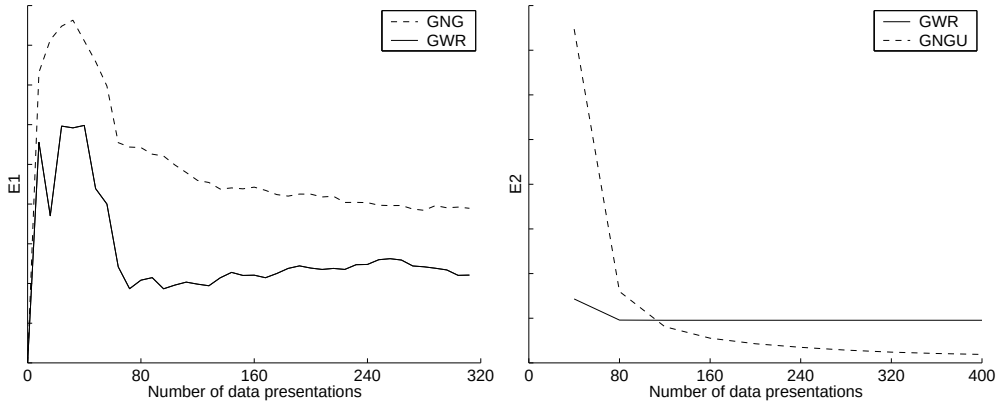


Figure 5.3: Evaluation of the two cost measures described in section 5.5 against the number of data samples for the GNG and GWR networks while the networks learn the four squares dataset.

Table 5.1: The number of nodes produced for the four squares problem, together with the number of data presentations required for the network to stop growing for different values of the activity threshold a_T.

a_T	Number of Nodes Created	Number of Data Presentations to Stop Growing
0.8	6	<40
0.85	6	<40
0.9	6	<80
0.95	10	<80
0.99	42	<120
0.999	204	<600

decreases very rapidly to a constant level. Obviously, the level at which the measure stabilises is controlled by the insertion threshold a_T. The topographic function was also evaluated for the GWR network on this dataset (see section 5.5.2) and a value of $\Phi(0) = 0.0023$ was recorded, which is very close to 0, as it should be.

Table 5.1 shows how the value of the insertion threshold affects the number of nodes that the algorithm generates and the number of data presentations that were required before the algorithm stopped adding nodes. The network shown in figure 5.2 was created using a value of $a_T = 0.99$. It can be seen that only when the insertion threshold is set to be very high does the network take a large number of data samples to stabilise.

5.6.2 A Dynamic Dataset

The Growing Neural Gas does not follow dynamic data distributions. However, Fritzke (1997) did suggest a variation, the Growing Neural Gas with Utility (GNGU) that could track a dynamically changing distribution. The difference between the GNG and GNGU networks is that the GNGU deletes nodes that do not affect the performance of the network on the current data. This was described in more detail in section 5.2.

The dataset used by Fritzke (1997) to demonstrate the GNGU network consisted of two squares positioned initially in the top-left and bottom-right corners of the unit square (as can be seen on the left of figure 5.4). At some time after training begins the distribution changes so that the inputs are now sampled from the top-right and bottom-left corners of the square (see the third and fourth pictures of figure 5.4). In the demonstration shown here, the input distribution was changed after 8000 data samples from the top-left and bottom-right squares had been presented, and then another 8000 data samples from the other two squares were presented to the network. The size of the networks was constrained to be no bigger than 30 nodes, in line with Fritzke's experiments. The values of the parameters were those given by Fritzke (1997). The learning rates were set at $\varepsilon_b = 0.05$ for the winning node and $\varepsilon_n = 0.0006$ for its neighbours. The insertion threshold for the GWR network was $a_T = 0.95$ and a new node was added after every 500 data samples for the GNG and GNGU networks.

Figure 5.4 shows that the GNG network does not follow the change in input distribution well, leaving dead nodes behind in those places where the input signals came from originally. The GNGU network, however, does follow the change in input distribution satisfactorily, as can be seen in figure 5.5. The network learnt by the GWR algorithm is shown in figure 5.6. It can be seen that the GWR network also follows the input distribution without any difficulty.

Comparing figures 5.5 and 5.6 shows that the GWR network grows a simple solution very rapidly, while the GNGU network is still changing the weight vectors of the nodes to produce a sensible ordering. When the distribution changes, which occurred immediately after the second picture, at 8000 data samples, the GWR network very rapidly repositions the nodes that were used previously, as well as adding additional nodes. This means that 4000 data samples after the distribution changes the network has learnt a reasonable representation, while the GNGU requires further training to delete a few superfluous nodes.

The evaluations of the two cost functions for the GWR and GNGU networks are shown in figure 5.7. Although both networks perform well on this task, it can be seen

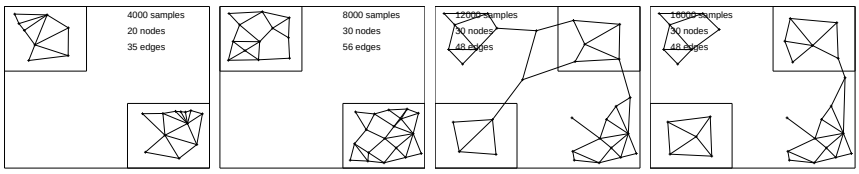


Figure 5.4: The Growing Neural Gas learns a dataset that changes suddenly after 8000 data samples (i.e., immediately after the second picture). It can be seen that the network does not track the change in distribution well, leaving a number of dead nodes behind.

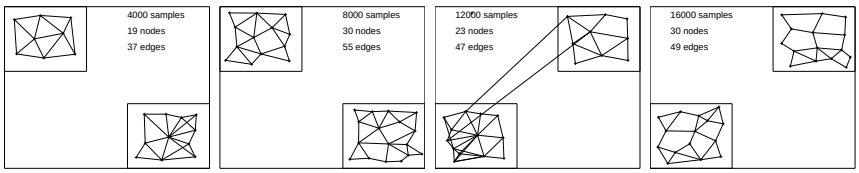


Figure 5.5: The Growing Neural Gas with utility learns a dataset that changes suddenly after 8000 data samples (i.e., immediately after the second picture). This network does track the change in input distribution, removing those nodes that no longer represent the inputs space.

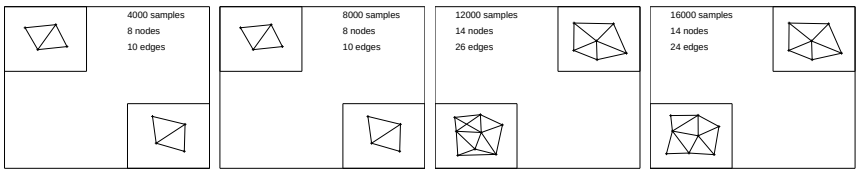


Figure 5.6: The network that Grows When Required learns a dataset that changes suddenly after 8000 data samples (i.e., immediately after the second picture). The GWR network also follows the change in data distribution without problems.

that for cost function E_1, the GNGU network takes considerably longer to recover from the change in input distribution than the GWR, and the maximum cost is considerably larger. The topographic function for the GWR network evaluated when all the learning was finished was $\Phi(0) = 0.001$.

5.6.3 A Multi-dimensional Dataset

The dataset shown in figure 5.8 was used by Martinetz and Schulten (1991) to demonstrate the Neural Gas network. The dataset consists of three parts, with three different dimensionalities – a parallelepiped, a rectangle and a circle connected to the rectangle by a line. It can be seen that the connections between the network nodes shadow the topological structure and dimensionality of the manifold on which the data lies. The

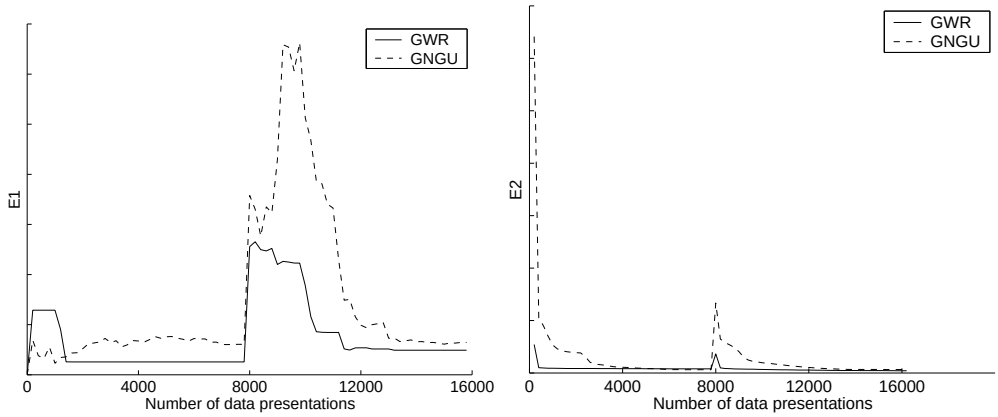


Figure 5.7: Evaluation of the two cost measures described in section 5.5 against the number of data samples for the GNGU and GWR networks while the networks learn the dataset that changes after 8000 data samples.

parameters used were the same as those used by Fritzke (1995) for the GNG, $\varepsilon_b = 0.2$, $\varepsilon_n = 0.006$, $\alpha = 0.5$, $a_{\max} = 50$ and the insertion threshold, $a_T = 0.95$. The GWR network learns considerably faster than the others did.

5.6.4 A Torus

This section introduces another dataset that demonstrates the performance of the GWR network. The data is sampled from the surface of a three-dimensional torus with internal radius 0.2 and external radius 0.5 (so that the entire torus fits into the unit cube).

Figure 5.9 shows the representation generated when an insertion threshold of $a_T = 0.97$ was used, while figure 5.10 shows the same for $a_T = 0.99$. It can be seen that substantially more nodes are added for the higher threshold, but that in both cases the representation that is generated is a good one. The first image of figure 5.9 shows the torus that samples were drawn from, while the first image of figure 5.10 shows 2000 samples, drawn at random.

5.6.5 The Two Spirals Dataset

The two spirals dataset is a benchmark dataset available in the CMU AI repository (`http://www.cs.cmu.edu/Web/Groups/AI/html/air.html`). Originally intended for supervised learning, the dataset is composed of two interleaved spirals, as is shown

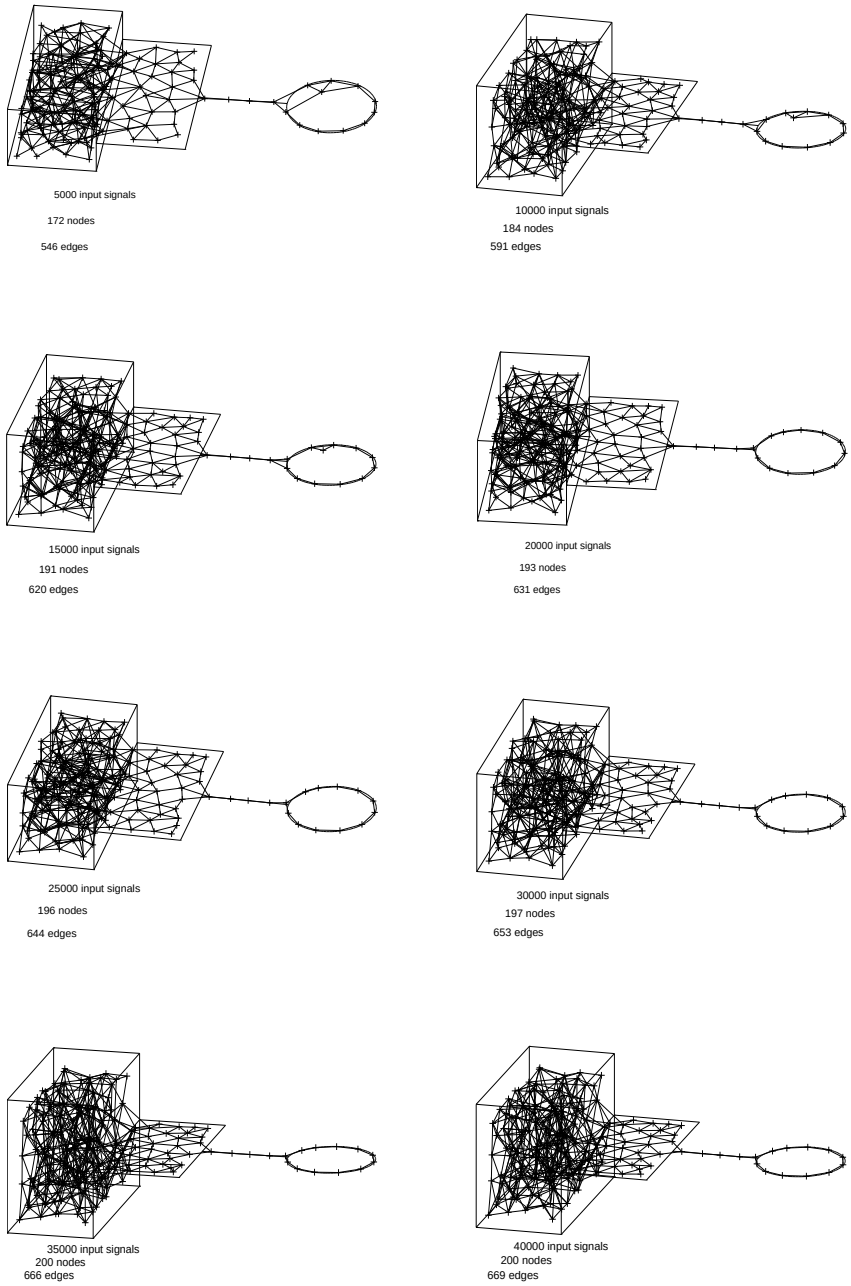


Figure 5.8: The GWR network learns a representation of a signal distribution that has three different dimensionalities. It can be seen that the network has learnt a good representation after 15000 data presentations, and does not change significantly after that.

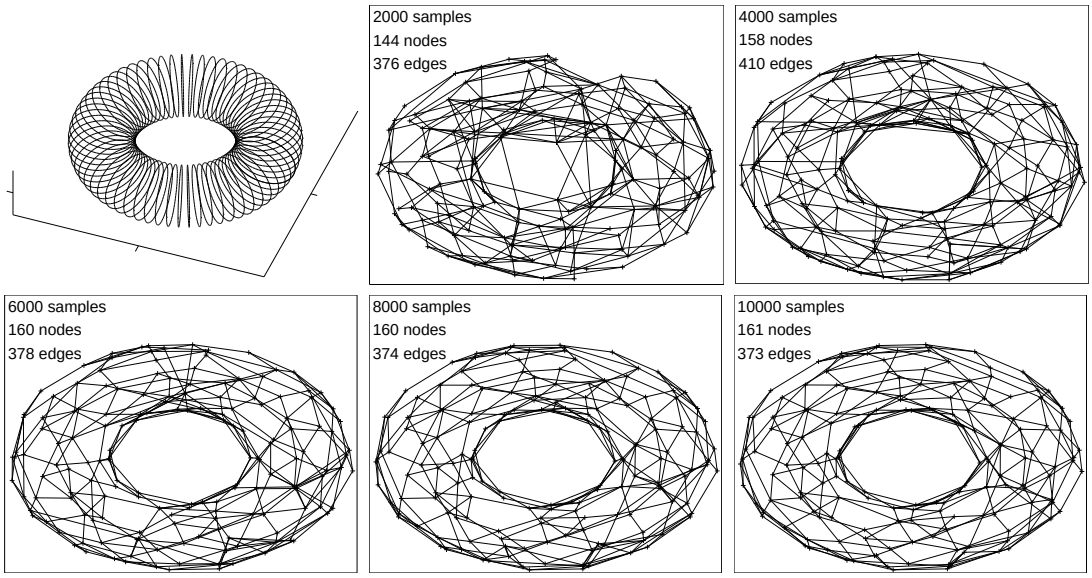


Figure 5.9: The GWR network learns a representation of a three-dimensional torus, $a_T = 0.97$.

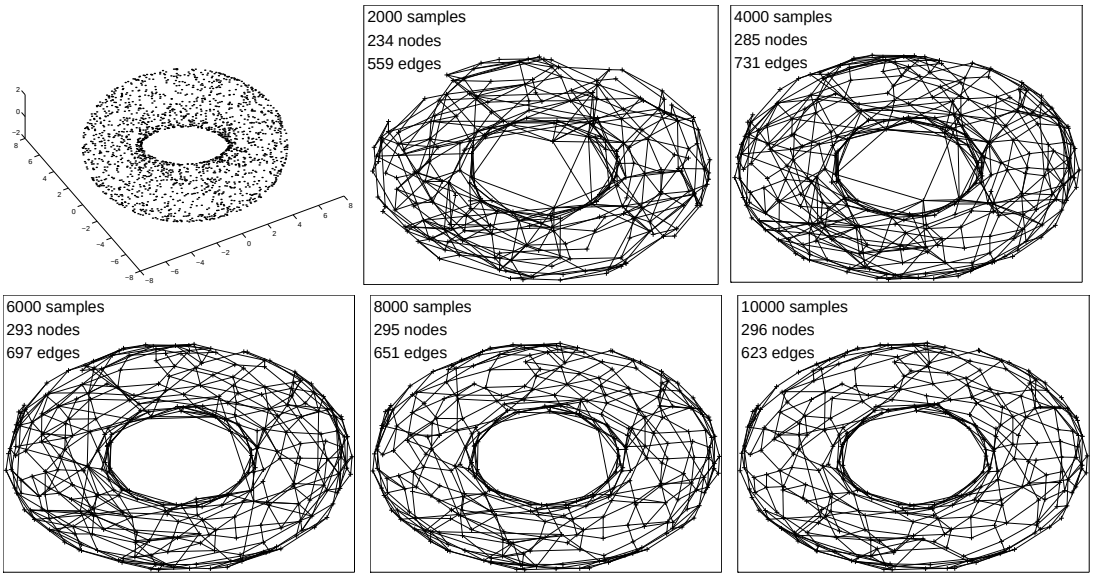


Figure 5.10: The GWR network learns a representation of a three-dimensional torus, $a_T = 0.99$.

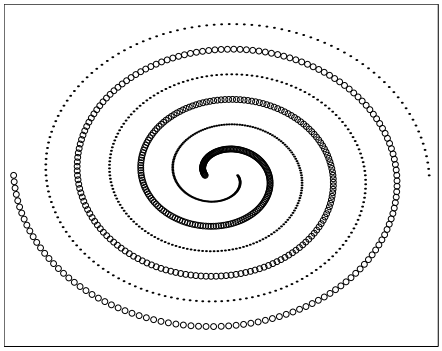


Figure 5.11: The two spirals dataset, which consists of data from two interleaved spirals.

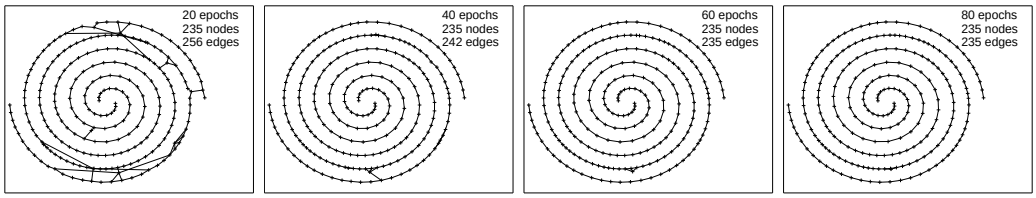


Figure 5.12: The GWR network learns a representation of the two spirals dataset. Note that the learning is unsupervised.

in figure 5.11. Bruske and Sommer (1995) applied their Dynamic Cell Structures algorithm (DCS) to the dataset as an unsupervised problem, in order to demonstrate the topology-preserving nature of their algorithm, described in section 5.2. They report that after 196 runs through the dataset (which contains 962 examples from the two spirals), their algorithm has learnt a perfectly topology-preserving map of the dataset, with the two spirals clearly separated. At this point the network has 198 nodes, since a new node is added at the end of every run through the training set. We applied the GWR network to the same problem, the results can be seen in figure 5.12 ($a_T = 0.8$).

The DCS algorithm, like the GNG, grows slowly. Bruske and Sommer show that after 80 epochs the DCS has only just got the shape of the spirals. By way of contrast, after 80 epochs the GWR network has generated the perfectly topology-preserving map. As usual, the GWR network grows very quickly when trained on this dataset – all the nodes were generated by the sixth epoch, and the remainder of the time was spent removing neighbourhood connections that crossed between the spirals. It is interesting to note that the number of nodes generated by the GWR network is of the order of the number required by the DCS network.

5.7 Summary

In chapter 4 it was demonstrated that combining habituating synapses and an unsupervised neural network produces a filter that highlights novelty, that is, features that do not conform to an acquired model of inputs that it has seen so far. The novelty filter that was used for the majority of the experiments in chapter 4 was the HSOM, which is based on the Self-Organising Map. Unfortunately, it was shown that the necessity for deciding in advance how large the network should be is a significant problem, as it means that too small a network can saturate, so that any input, however novel, would be found to be normal.

This chapter has proposed a solution to the problem. A new self-organising network, the 'Grow When Required' (GWR) network, has been described, where the algorithm adds new nodes into the map space whenever the current input is not matched to some specified accuracy by any of the nodes currently in the map. The algorithm maintains a set of neighbourhood connections (edges) between nodes that represent similar inputs, in a similar way to the Neural Gas (NG) of Martinetz and Schulten (1991) and the Growing Neural Gas (GNG) of Fritzke (1995). These edges are used to form neighbourhoods of nodes.

The principal difference between the NG and GNG algorithms and the GWR is that the NG has a predefined number of nodes in the map space, the GNG adds a new node every λ iterations and, instead, the GWR adds a new node whenever the current input is not matched to sufficient accuracy by the current winning node. This means that several nodes can be added one after another, but then none added for hundreds of iterations, dependent only upon the input data. The benefits of this approach for novelty detection are that the network responds immediately to a novel stimulus (unlike the GNG, which would have to wait until a new node was added), but cannot saturate (like the NG, which has the same problem as the SOM if used on-line).

Section 5.5 discussed some of the theory of neighbourhood preservation in self-organising neural networks and described two cost functions that could be used in order to evaluate the performance of a self-organising network on a dataset. These cost measures were used to compare the GWR and GNG networks on two toy datasets in section 5.6. The first of these was a very simple dataset where the inputs came from four squares situated within the unit square, while the second was a dynamic dataset, where the input distribution changed after some time. Finally, two further datasets have been used to demonstrate that the GWR network is perfectly topology-preserving, that is, that the dimensionality of the map structure represents the intrinsic dimensionality

of the input manifold.

The next chapter demonstrates the application of the GWR network to a number of novelty detection tasks. These include the robotic inspection tasks described in sections 4.3 and 4.4. The GNG and RCE networks are also tested on this data in order to provide some comparison results.

Chapter 6

Novelty Detection with the GWR Network

In this chapter the GWR network is applied to a set of novelty detection tasks, first the robot inspection tasks described in chapter 4, together with an extension to larger environments and then a number of non-robotic novelty detection tasks.

6.1 Introduction

In the previous chapter the GWR network was introduced and the learning abilities of the network were demonstrated on a number of simple datasets. However, none of the examples concerned novelty detection. This chapter examines the novelty detection capabilities of the network. The GWR network has the habituation synapses built in, because they double as the measure of how often a node has fired before, and so the network does not need any alterations to operate as a novelty filter.

Section 6.2 tests the novelty filter based on the GWR network on a simple two dimensional dataset, so that the performance of the network can be visualised. Then, in section 6.3 the GWR network is applied to the robot sensor data that was used in section 4.3 for the small-scale on-line inspection task. Novelty filters based on the GNG and RCE networks are also used in this section to provide a comparison with the output of the GWR network.

In section 6.3.3 the GWR network is tested on the data from the 200 m circuit of the 2nd floor of the Computer Science Department, the environment that was used in section 4.4 in order to demonstrate that the HSOM network could saturate. Section 6.4,

compares the GWR network to another novelty detector, one based on a Support Vector Machine. Two datasets are used to compare the two novelty detectors. The datasets come from the two most typical uses of novelty detection systems – medical diagnosis and fault monitoring in machinery. This is because for both of these applications there are typically fewer examples of the important class – the disease, or machine breakdowns – than of normal operating data, and so a simple classifier cannot be used. A further example on oil pipeline data is also given.

6.2 A Simple Novelty Detection Task

This toy dataset demonstrates the novelty filter with habituation based on the GWR network. The training set, an example of which is shown in figure 6.1, consists of 200 samples from each of three Gaussian clusters. Data from each of the clusters are plotted differently, but this is purely to aid clarity in the figure. The actual data consists only of the two dimensional coordinates, and the network is trained in an unsupervised fashion. The two clusters with the principal axis vertically have covariance matrix $\begin{pmatrix} 0.2 & 0 \\ 0 & 0.3 \end{pmatrix}$ and means at $(2,5)$ for the left-hand cluster and $(8,5)$ for the right-hand cluster. The third cluster, which has the principal axis horizontally, has covariance matrix $\begin{pmatrix} 3 & 0 \\ 0 & 0.1 \end{pmatrix}$ and a mean of $(5,2)$.

The GWR network was trained on the data shown in figure 6.1, with insertion threshold $a_T = 0.95$ and learning rates $\varepsilon_b = 0.2$ for the best-matching unit and $\varepsilon_n = 0.05$ for the neighbours. After this training the network shown in figure 6.3 was produced. This network has 25 nodes and 35 edges. As in section 5.6, each node is plotted at the position of its weights, and lines show edge connections. The network has modelled the entire dataset as one cluster. This is reasonable, since the three distributions overlap.

The trained network was tested on the data shown in figure 6.2. This test data consists of 50 datapoints sampled from each of the Gaussian clusters that generated the training data and also of 30 points from a uniform distribution positioned between the Gaussian clusters (with points plotted as squares). These are the novel data.

During testing any input that would require the addition of a new node (i.e., any node where the activity was below the insertion threshold) was considered to be novel. These points are shown in figure 6.4 by having a box drawn around the datapoint. It can be seen that all the data from the uniform distribution are found to be novel, as

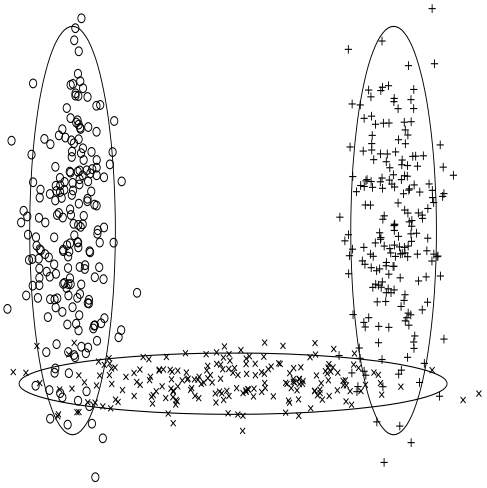


Figure 6.1: The training set, data sampled from the three Gaussians shown. The different shapes of the datapoints show which of the three Gaussians a point was sampled from, although the network was trained unsupervised. The ellipses represent the 95% confidence contour of each Gaussian.

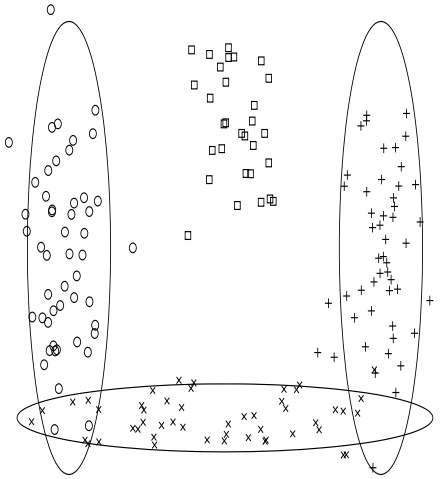


Figure 6.2: The test set, with data sampled from the three Gaussians shown and some additional 'novel' data (the squares in the top-centre of the figure).

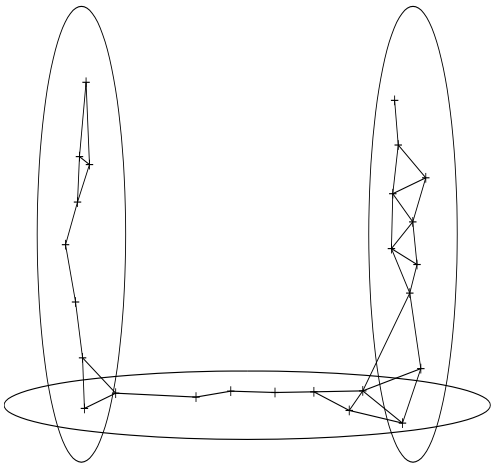


Figure 6.3: The positions of the weight vectors and edge connections of the trained network.

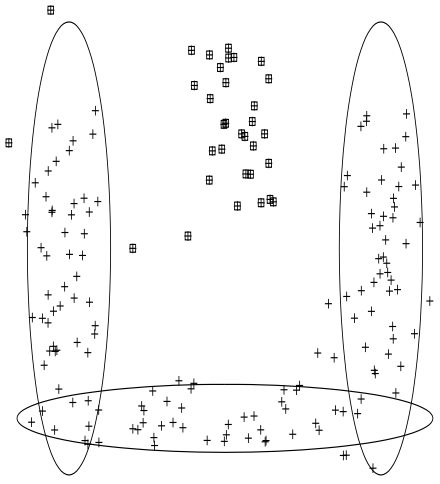


Figure 6.4: The output of the novelty filter. Squares show the positions of datapoints that were considered to be novel.

would be expected, and so are three datapoints from the left-hand Gaussian. These three points are all outliers to the dataset and should therefore be labelled as novel. No other points are labelled as novel, and this appears to be reasonable – there are no other points that appear to lie significantly outside their distributions.

In this case learning was turned off for testing. While this means that the novelty filter is not being used on-line, it does enable the novelty detection ability of the filter to be evaluated. When learning remains switched on, the filter recognises most of the elements of the new distribution to be novel after a few have been seen.

6.3 The Inspection Experiments with the GWR

6.3.1 Description

The sonar data that was used in section 4.3 to investigate the performance of the HSOM on the robot inspection task was stored while the experiments were conducted. This enabled the data from these experiments to be used to test other networks, and also different parameters for the various networks. The environments are those shown in figure 4.5 on page 99, and the same approach was used – the networks were trained from random initialisation in environment A for three training passes through the environment, and then this trained network was tested in three different environments. The first was environment A*, which was the same as environment A except that a door to the right of the robot (door D in figure 4.5) was opened so that the perceptions at this point changed, but otherwise the environment was identical. Environment B, the second environment, was similar to environment A, while environment C, while still part of a corridor, was from a different part of the building.

This section shows the output of the novelty filter based on three different networks, the GWR network described in section 5.4, the GNG network described in section 5.2 and the RCE network described in section 2.7.3, when trained on the sonar data. The figures in the following sections show the activity of the output neuron of the novelty filter when the filter is trained on the sonar data. Only the training runs are shown. The output of the filter based on the GWR network is shown next to the output of both the GNG and RCE networks, to aid comparison.

As in section 4.3, a high spike means that the perception was novel, while short spikes mean that the perception fitted the model of previously seen perceptions. A spike that reaches the top of the graph means that a new node was generated to match

that perception.

6.3.2 The Small-Scale Environments

Environment A

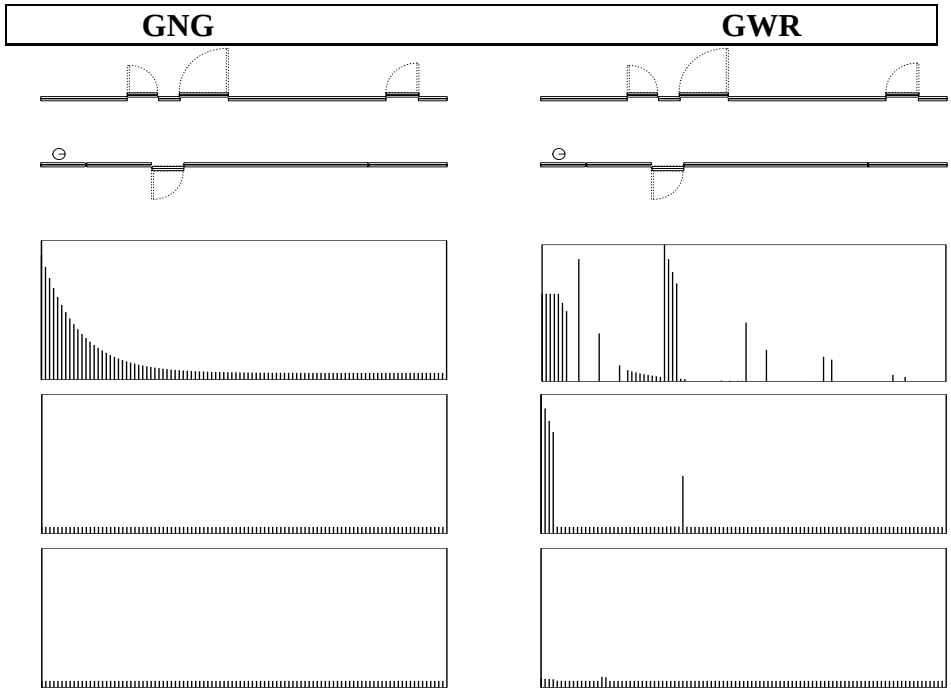


Figure 6.5: The Growing Neural Gas and Grow When Required networks learning about environment A. The networks had no initial training. All of the graphs show the results with learning turned on. The GWR network highlights the perceptions of the door, but quickly learns to recognise perceptions of the wall. By the third run nothing is found to be novel. The GNG does not appear to detect novelty at all. This is because the network only adds nodes in the middle and at the end of each pass through an environment. The reasons for this are discussed in the text.

Figure 6.5 shows the output of the GNG and GWR networks and figure 6.6 shows the output of the RCE and GWR networks when initially untrained networks are trained in environment A. The parameters that the networks had in common were the same for both the GWR and GNG networks, being learning rates $\varepsilon_b = 0.1$ for the best-matching node and $\varepsilon_n = 0.001$ for the neighbours of the best-matching node. In the GNG network a new node was added after every 50 data presentations (i.e., twice in every pass through an environment), while the insertion threshold for the GWR network was $a_T = 0.9$. The sphere of influence for the unsupervised RCE network (see

figure 2.7 on page 59), which defines the radius of the hypersphere around each prototype within which a vector is classified as belonging to that class, was set to be 0.05. The RCE network does not have a learning rate, as the prototype vectors do not move. If the current input is not matched by any of the classes then the input is used as a prototype vector for the new class that is created. Nor does the RCE network have the concept of neighbours, just a set of prototypes, with each prototype representing a class.

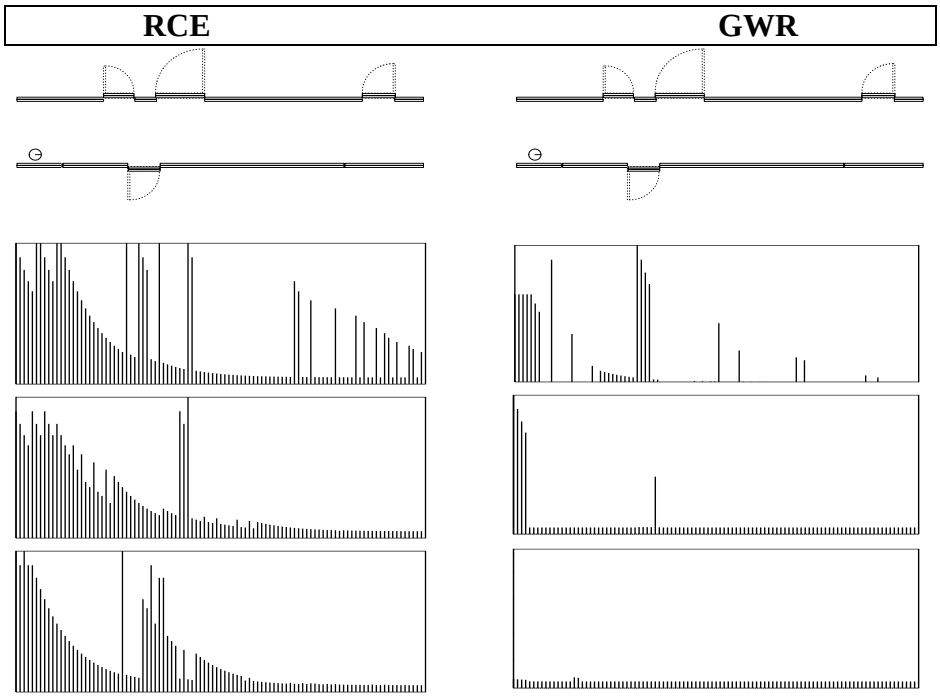


Figure 6.6: The RCE and Grow When Required networks learning about environment A. The networks had no initial training. Both networks highlight the doorway as novel, but the RCE network has some problems learning because of a lack of resilience to noise. Many other features are found to be novel, especially at the beginning of the run.

On the right of figure 6.5, the GWR network behaves as would be expected – initially the perceptions are found to be novel, and then only the perceptions of the doorways are detected. These new perceptions are learnt about during the first run, and only one perception of the doorway is found to be novel during the second run. In the third run, nothing is found to be novel. This is exactly how the output of the novelty filter should be. Unfortunately the GNG network (on the left of figure 6.5) cannot do this. This is because there are only two neurons in the network until after the doorway is perceived, and so the network cannot show this novelty. Therefore nothing is found

to be novel.

The RCE network shown on the left of figure 6.6 appears to have the opposite problem. The network finds the perceptions of the doorway novel, but also finds many other perceptions to be novel and does not appear to generalise well. This is because the setting of the sphere of influence of the network is fairly low, so that the network is not robust to noise. The value of 0.05 that was used was picked so that the open doorway in environment A* (shown in figure 6.8) was reliably detected. When the novelty filter is tested in environment A* after training in environment A, it is expected that a successful novelty filter will act like the HSOM did – finding the perceptions around the doorway novel, but recognising everything else.

The question of how to choose how frequently to add new nodes for the GNG network is critical. If nodes are added twice during each run, as here, no novelty is detected. By adding nodes at shorter intervals, the network would begin to detect features as novel, on those occasions that a novel feature was spotted (and gave the highest error) when a new node was about to be added. So if a new node was added at every iteration, all novel features would be spotted. However, this network would be very sensitive to noise and would grow bigger and bigger without stopping. There is a trade-off between adding nodes sufficiently regularly that most novel features are detected, and not letting the network grow too fast. This is a problem that is inherent in the network architecture.

Environment A*

Figure 6.7 shows the output of the novelty filters based on the GNG and GWR networks. Again, the GNG network does not respond in the desired manner. The perceptions in the middle of the environment, where a new node was added, are found to be novel, although nothing new is situated there. On the other hand, the GWR network shown on the right of the figure behaves exactly as expected – the perceptions of the doorway are found to be novel, and are learnt so that in the third pass through the environment they are recognised. During the second run the sharp spike of novelty is caused by the crack in the wall that was mentioned in section 4.3.

During some runs the GNG does find novelty in the middle of an environment, and sometimes it does not (compare figures 6.5 and 6.7). This is because the node that is added may not be used by the network immediately, and therefore no novelty is shown.

As was mentioned above, the sphere of influence of the RCE network was selected to ensure that the network found the perceptions of the doorway to be novel. It can be

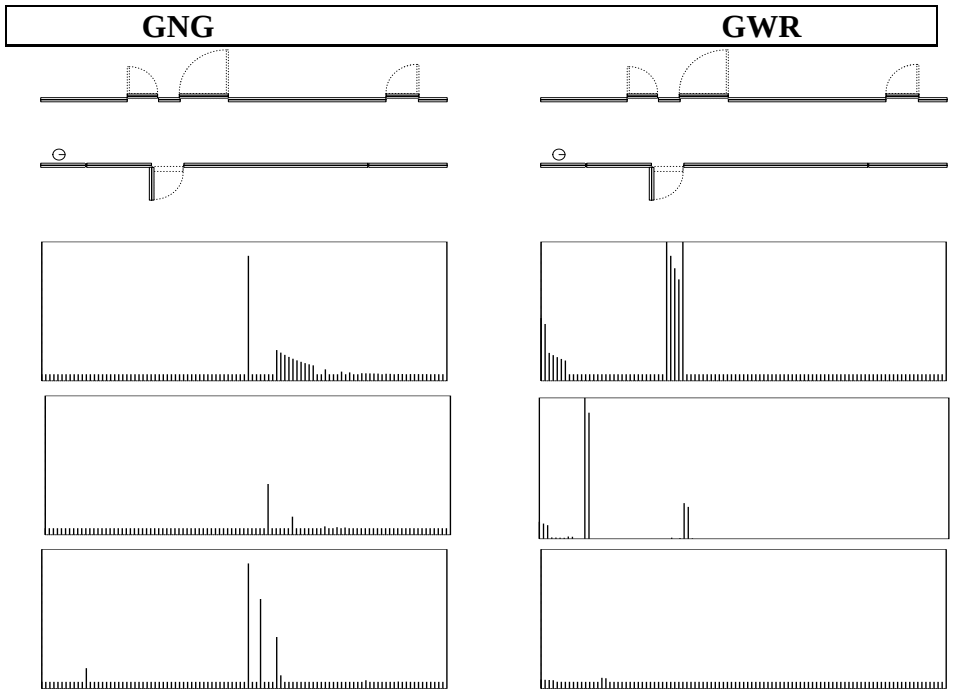


Figure 6.7: The Growing Neural Gas and GWR networks learning about environment A* (door open), after training in environment A with the door closed. Again, the GNG does not respond well, but the GWR network correctly highlights only the perceptions of the doorway as novel.

seen on the left of figure 6.8 that the perceptions of the doorway are indeed indicated to be novel. However, with this value of the parameter, the network remains very sensitive to noise, and so finds many other perceptions novel and does not appear to learn well. This is because the prototypes that are used to define the clusters are made of any input that does not match any stored prototype, and the positions of these prototypes do not change while the network learns. In addition, there are no neighbourhood connections, so nearby clusters do not interact.

Environments B and C

The performance of the GNG network was very disappointing. At no time did the network correctly detect novel features at the right time. This did not improve when the network trained in environment A was tested in environments B and C. Therefore, these figures are not shown. The output of the GWR and RCE novelty filters are shown for testing in environment B in figure 6.9 and for environment C in figure 6.10, in both cases after training in environment A.

Environment B is similar to environment A, and therefore it would be expected that

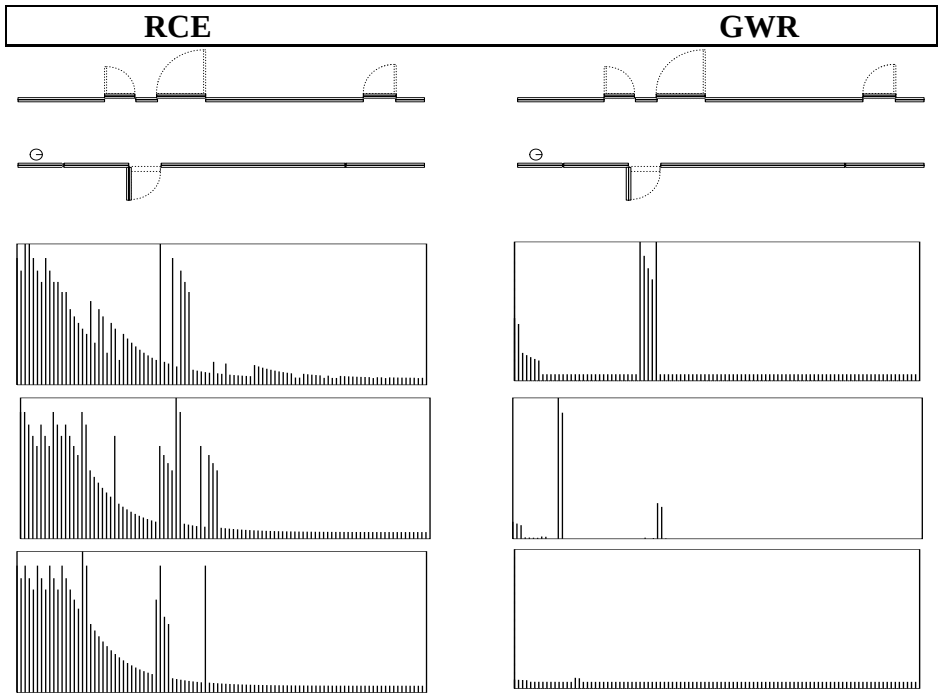


Figure 6.8: The RCE and GWR networks learning about environment A* (door open), after training in environment A with the door closed. Although the RCE network does find the doorway to be novel, it does not appear to learn about it reliably.

very little was found novel. However, in section 4.3 it was found that the doorways were more deeply inset, and therefore are detected. This is also found for both the GWR and RCE networks. However, the HSOM algorithm did not highlight the perceptions at the end of the environment where a number of boxes project approximately 7 cm from the wall. Their position is marked on the diagram of the environment, and the boxes can be seen in the photograph in figure 4.5 (page 99). It appears that the GWR and RCE networks do detect these perceptions as novel. Again, the GWR learns about the environment so that the third run is completely normal, while the RCE network has problems with noise.

For environment C (figure 6.10) the GWR network finds the perceptions at the beginning of the environment (which is made of brick rather than breezeblock as in the other environments) to be novel, then highlights the doorway and the posters at the end of the environment, which appear to be very noticeable to sonar. The performance of the RCE network is very similar, although it does not learn as well as the GWR network.

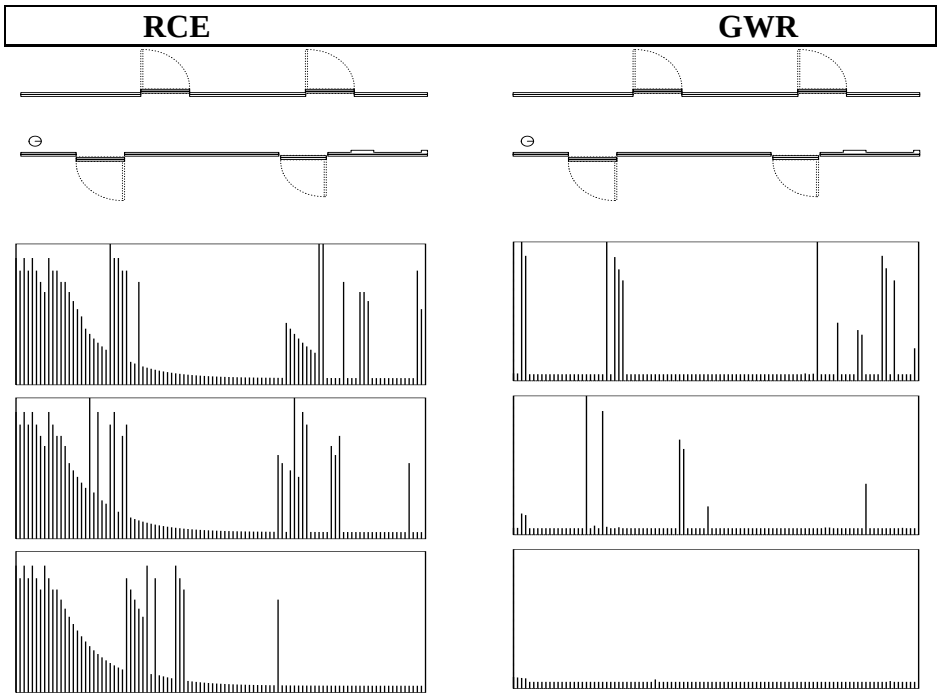


Figure 6.9: The RCE network and Grow When Required network learning about environment B after initial training in environment A. Both networks find the same perceptions to be novel, but the GWR network learns faster.

6.3.3 Scalability: The Large Environment

This section shows how the novelty filter based on the GWR network performs when tested on the data from a large environment, the 200 m circuit of one of the two loops that comprise the 2nd floor of the Computer Science Department at the University of Manchester. In section 4.4 it was suggested that for the novelty filter to be considered to be working correctly, the only features that would be highlighted as novel in the final pass through the environment (after two other training passes through the environment) would be those at the end of the run, where the environment was changed. In figure 6.12 it can be seen that this is exactly what happens when the GWR network is trained in this environment.

The number of nodes added during each of the training runs is given on the right of each graph in figure 6.12. In total, the final trained network comprised 37 nodes, which is fewer than the smaller HSOM network described in section 4.4. This is because the only nodes that are generated are those required to describe the data seen, which is very different to the SOM. As in the output of the HSOM-based novelty filters, much of the novelty is found when the robot turns around a corner, because of the fact that

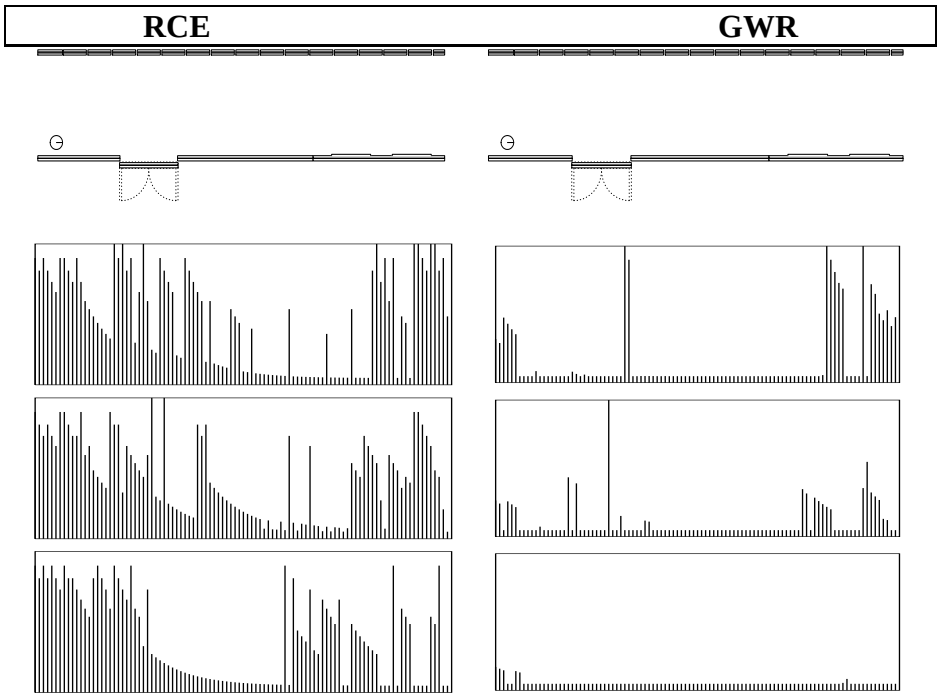


Figure 6.10: The RCE network and Grow When Required network learning about environment C after initial training in environment A. Again, the same perceptions are found to be novel, but the GWR network learns more quickly.

the control program does not constrain the turns very much. However, even by pass 2, very little is found to be novel.

One noticeable feature is that in pass 4 there are several perceptions that are found to be novel. Although no learning is performed in this pass, they are not found to be novel in pass 5. This is because these features are places where the robot was turning a corner, and so they were not seen in pass 5 because the robot took a different trajectory at these points. Furthermore, connected and fully habituated nodes fired during pass 5, mutually habituating the relevant nodes.

This experiment shows clearly that the novelty filter based on the GWR network fulfils the requirement that novelty filter should scale with the size of the environment. Further, the GWR network also fulfils the requirements for the neural network on which the filter is based that were identified in section 1.3. The experiments that are presented in chapters 7 and 8 extend the functionality of the GWR novelty filter.

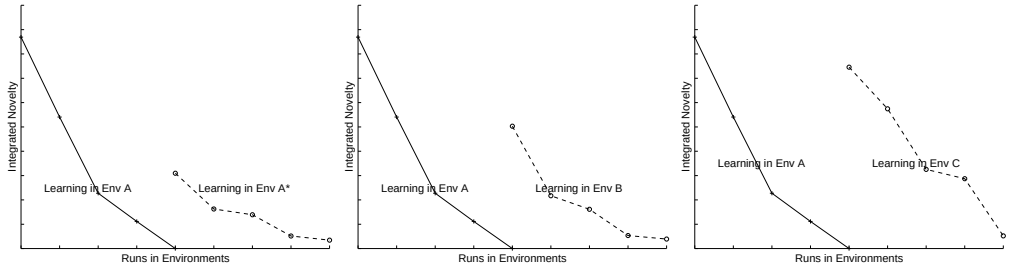


Figure 6.11: Graphs showing how the amount of novelty in an environment decreases as the GWR-based novelty filter learns over three runs in an environment. The graph on the left shows the robot learning first in environment A, and then in environment A*. It can be seen that once the robot has learnt about environment A, very little in A* is novel. Figure 6.7 shows that what is novel is the area around the (now open) door. The middle graph shows how the amount of novelty increases when the robot explores environment B after learning about environment A, and finally the graph on the right shows how the novelty increases when the robot begins to explore environment C.

6.3.4 Discussion

The results shown in this section have demonstrated that the GWR network reliably highlights as novel those features of an environment that have not been seen before. Novelty filters based on other growing networks, the GNG network, which has a similar basis to the GWR network, and the RCE network, were also tested on the same data in order to provide comparisons.

It was shown that these two comparison networks have different difficulties in modelling the data, which the GWR network avoids. The GNG network adds a new node into the network twice during each pass through an environment. This means that the network can only respond to novel stimuli that are detected at those points. Therefore, very little is found to be novel. In fact, nothing will be found to be novel if the network adds nodes to model stimuli that are not seen at the time when the node is actually added, i.e., the network adds a node to recognise a door on the right of the robot after all the doors have been seen.

The RCE network has the opposite problem – features are still found to be novel after two training passes through an environment. There are two reasons for this – first the RCE network uses the input vectors as prototype vectors to define clusters, and does not change the positions of these vectors once they have been set. This means that any input that does not fit into a cluster that has already been made will generate a new cluster. Unfortunately, the noise in the sonar readings means that many new clusters are added for the perceptions at a single point. Secondly, because the RCE

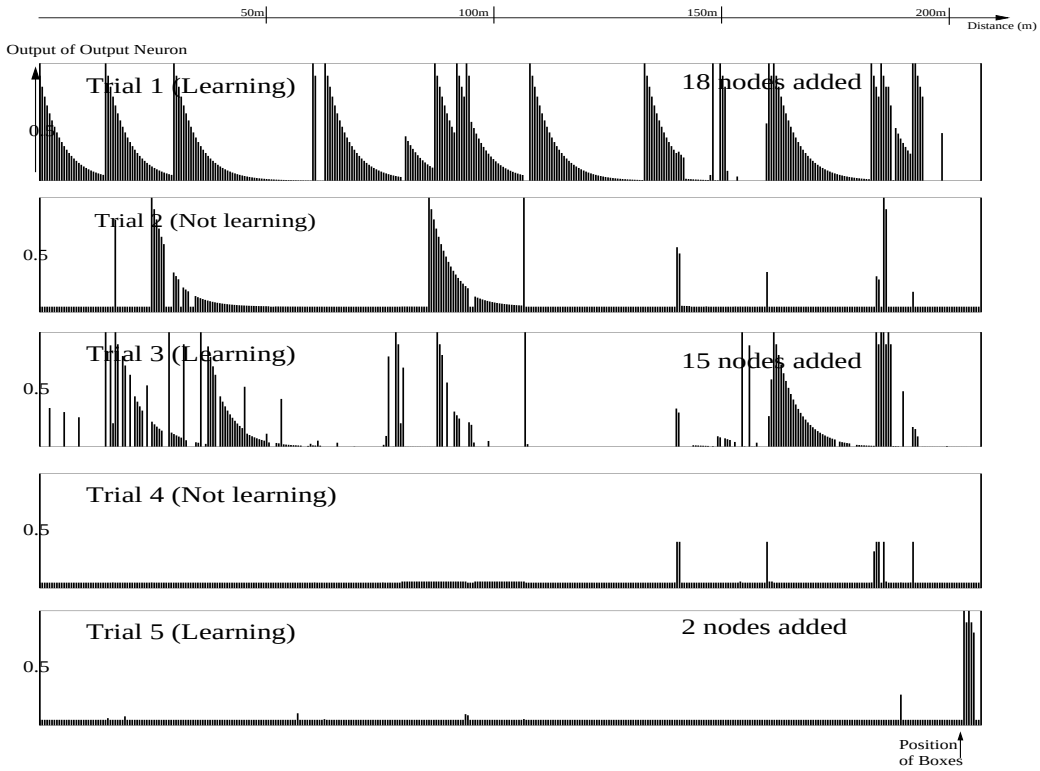


Figure 6.12: Plots of the novelty found when the novelty filter based on the GWR network is trained while the robot explores the entire loop of corridor around the Computer Science Building, a distance of about 200 m. At the end of the final run the environment was changed by making a dead end using cardboard boxes.

network does not use a neighbourhood, the clusters do not affect one another, and so each cluster has to be seen several times before it stops being novel. This takes a very large amount of training.

In effect the GWR network is a compromise between the two networks – nodes are added whenever the current input is not matched to sufficient accuracy by any of the nodes already in the network, but the positions of node centres change as the network learns and neighbourhood connections between nodes are maintained. This means that the network learns to represent the data more accurately, and hence uses fewer nodes.

Comparing the output of the GWR-based novelty filter to that of the HSOM in section 4.3 raises some interesting points. Firstly, because novel inputs generate new nodes, so the output of the novelty filter for novel inputs looks stronger because the spikes are higher (as neighbouring nodes habituate one another, very few nodes are at full strength when they fire in the HSOM). The networks that are generated by the GWR algorithm are also very economical – the network generated in section 6.3.3

had fewer nodes than the smaller of the two HSOMs used in section 4.4. This means that the GWR novelty filter differentiates more clearly between novel and non-novel stimuli.

6.4 Non-Robotic Applications

In this section the GWR network is tested on three datasets that are publicly available via the internet. The first two of these datasets were used in Campbell and Bennett (2000) to test their novelty detection technique that is based on modelling the support of a data distribution using a Support Vector Machine (SVM). This novelty filter was described in section 2.9.2. It uses a soft margin classifier with a Gaussian kernel (equivalent to an RBF network)

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{-|\mathbf{x}_i - \mathbf{x}_j|/2\sigma^2}. \tag{6.1}$$

to generate a hypersphere in feature space that contains most of the data. The novelty detection is based on the idea of Tax and Duin (1999).

The datasets are described below. The first two demonstrate the two areas where novelty detection is typically used – medical diagnosis and fault detection. As was mentioned previously, this is because in both of these cases there are typically many examples of healthy data, patients whose tests do not highlight illnesses and machines working normally, and relatively few of problems. In addition, it is not known for definite that all possible manifestations of unhealthy data are known and the main purpose of the data is to avoid false negatives – not detecting possible problems. These are the circumstances under which novelty detection is the method of choice. The third dataset is similar to the one used by Bishop (1994) to demonstrate the Bayesian approach to neural network validation and novelty detection (see section 2.6).

The following sections describe the datasets and the approach to training and testing that is used. The results achieved by the GWR network are given, as are those of the SVM novelty filter.

6.4.1 Dataset One: Medical Diagnosis

The first dataset is the *biomed* dataset available at `http://lib.stat.cmu.edu/datasets` (Cox et al., 1982). In total, 209 observations are given, of which 15 have one or more of

the four attributes (measurements on blood samples) missing, and were therefore discarded. Of the remaining 194, 127 were normal, i.e., healthy data, and the remaining 67 contained one or more abnormalities, signifying the presence of a genetic disease.

Following the approach of Campbell and Bennett (2000), 100 randomly chosen normal observations were used as a training set and the remaining 27 normal observations together with the 67 inputs that should be detected as novel were used as a test set.

The results reported for the kernel classifier were very good – their best result was that 57 of the 67 novel inputs were correctly labelled and only 2 of the normal data were mislabelled as novel. The GWR network performed similarly. For a value of the insertion threshold of $a_T = 0.92$, 56 of the 67 novel inputs were highlighted as novel, and only 2 of the normal data were misclassified. This is very similar to that of the kernel machine. Unfortunately Campbell and Bennett (2000) do not report which datapoints were misclassified, and therefore it is not possible to see if the same points were found by both filters.

Campbell and Bennett (2000) show how varying the variance σ in the RBF kernel that was used (equation 6.1) affects learning. For small σ, all the test data are labelled as novel, because a small Gaussian is centred on each training point, with each point requiring its own separate Gaussian cluster to represent it. This is what happens in the GWR network when the insertion threshold is high. As the insertion threshold decreases, or equivalently σ increases, generalisation gets better, but then novel inputs that are close to clusters can be missed. Therefore there is a trade off between false positives and false negatives. As was discussed previously, false positives (that is, points incorrectly labelled as novel) are less of a problem, providing there are not too many of them, in which case the classifier is useless. However, the classifier should highlight all the novel inputs. Neither novelty filter manages to detect all the novel inputs on this particular dataset, although they do get fairly close.

6.4.2 Dataset Two: Fault Detection

The second dataset is available at `http://www.brunel.ac.uk/research/cnca/sida/html/data.htm` as part of the EPSRC Structural Integrity and Damage Assessment (SIDA) network. The data consists of several sets of tests on ball-bearing cages. These are vital components of many machines, and therefore it is important to detect faults in them before use.

Every instance of data consisted of 2048 acceleration samples from a Bruel and

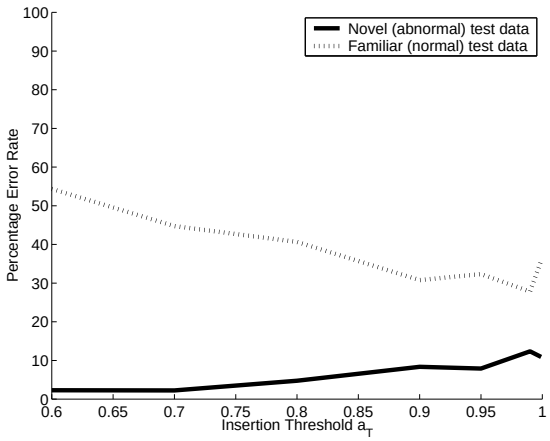


Figure 6.13: The error rate against insertion threshold for normal ball-bearing test data (solid line) again novel test data (dotted line) with the GWR.

Kjaer vibration analyser. A discrete Fast Fourier Transform was used to preprocess the data, resulting in 32 attributes for each input vector. The data was split into five types:

Type	Description	Number of Instances
normal	new ball-bearings	two sets of 913
fault 1	external ring completely broken	two sets of 747
fault 2	basket damaged, 1 degree of freedom	two sets of 996
fault 3	half of basket elements destroyed, 4 degrees of freedom	996
fault 4	no visible damage, but runs loosely	996

The experiments described in Campbell and Bennett (2000) used the first set of data from each of the first three categories (normal, fault 1 and fault 2) to train the classifier, and then used the second set of data in each of these categories for testing. Finally the fault 3 and fault 4 sets were used to test the novelty detection performance. This tests more than just the novelty detection ability of the network, since it also tests the classification performance of data from the three categories that were used in training.

The results reported by Campbell and Bennett (2000) were more concerned with the correct classification of the test data from the first three categories, rather than the novelty detection performance. They report 1.3% error on the data from the normal data, 0% error on fault 1 and 46.7% error on fault 2, but only found 28.3% of the fault 3 data and 25.5% of the fault 4 data to be novel.

The results of using the GWR network are the opposite to those of the SVM approach. Rather than learning a good representation of the trained classes, but misclassifying many of the novel inputs as familiar, the GWR network performs very well at recognising novel inputs, but finds many of the test elements of the familiar classes to be novel too. Figure 6.13 shows how the performance of the GWR network changes for familiar and novel test data as the insertion threshold a_T varies. It can be seen that as the insertion threshold gets closer to 1, so the categorisation performance of the familiar data gets better, but at the cost of missing some of the novel inputs.

When the insertion threshold is $a_T = 0.8$ the GWR network finds 95.4% and 95.1% of fault 3 and fault 4 data to be novel, but misclassifies as novel 37.8% of the good data, 40.3% of fault 1 and 43.8% of fault 2. Only for the fault 2 data are the SVM data similar. However, as was argued in chapter 1, for novelty detection it is more important to find the novel entries – although the number of familiar perceptions misclassified as novel is high, this is less important.

6.4.3 Dataset Three: Three Phase Oil Pipeline Data

The final dataset, available from `http://www.ncrg.aston.ac.uk/GTM/3PhaseData.html`, is similar to that used by Bishop (1994) (described in section 2.6.2). The dataset consists of a set of 12-dimensional synthetic measurements of the flow in a pipeline that contains oil, water and gas. Three different types of flow are represented in the data – a homogeneous mixture of oil, water and gas, an annular configuration and a stratification. These are pictured in Bishop (1994).

In that paper, Bishop created a set of novel data by generating further artificial data. As the creation process is not given in the paper this process was not possible. Instead, one of the classes was missed out of the training set. Then, in the test set, precisely the members of the missing class should be selected as novel. This is exactly what happened.

The original training set comprised 1000 measurements, of which 316 were labelled as being examples of the annular configuration, which was selected as the class to be missed out. In the test set 363 of the 1000 examples belonged to this class, and 352 of these were labelled as novel (97.0%), with no examples from the trained classes being misclassified. On the (admittedly slightly harder) problem considered by Bishop (1994) there were a significant number of misclassifications, although he does not give any details.

6.5 Summary

This chapter has described the application of the GWR network to a number of different novelty detection tasks. In section 6.2 the network was applied to a simple novelty detection task where the training set consisted of data sampled from three Gaussian distributions, and the test set had data added from a further uniform distribution. It was seen that the network perfectly detected all the data from the new distribution, and also outlying data from the Gaussians, to be novel.

In section 6.3 the novelty filter based on the GWR network was used to detect novel stimuli in the sonar data collected by the robot as it explored during the small-scale inspection experiments described in section 4.3. These experiments were used in section 4.3 to show that the novelty filter based on the HSOM could learn about an environment as the robot explored and detect novelty on-line, highlighting perceptions that did not fit into the acquired model.

As a comparison, novelty filters based on two other networks, the GNG and RCE were also used. It was shown that the GNG did not perform novelty detection well because nodes were added only after a fixed number of training iterations, and so the filter could not respond immediately to novel stimuli. The RCE network, however, did perform well on novelty detection, but had many problems with noise, finding many features that it had seen before to be novel.

The GWR network was then applied to a large environment consisting of 200 m of corridor (section 6.3.3). This was the environment used in section 4.4 to demonstrate that the HSOM network could saturate. It was shown that not only did the GWR network learn well and reliably detect as novel the altered part of the environment in the final pass, but in addition, the network that was generated had fewer nodes than even the smallest HSOM used.

Finally, in section 6.4, the GWR network was applied to two datasets that were available in the public domain and that had been used by other novelty detection systems. The two datasets came from the areas where novelty detection is most frequently used, health inspection and fault monitoring. It was shown that the performance of the GWR network on these two datasets was comparable to that of the test network, a kernel-based learning machine, which is a very satisfying result.

The next chapter demonstrates the effects of using the GWR novelty filter when the robot explores the same environments, but generating inputs to the novelty filter from images captured by a monochrome CCD camera. This is a much more complicated sensor and requires substantial preprocessing before the image can be used as input to

the network.

Chapter 7

Detecting Novelty in Visual Input

The question of how to detect novelty in images taken from a camera is considered in this chapter. A number of ways of generating input vectors for presentation to the novelty filter are discussed, and the results demonstrated.

7.1 Introduction

The experiments that have been concerned with inspection by a robot described in previous chapters have used the sonar sensors to sample the environment. This is quite limiting. Although ultra-sonics are important sensors for a mobile robot, they are not the only type of sensors that are used in modern research robots. The Manchester Nomad 200 robot *FortyTwo*, shown again in figure 7.1, is also equipped with infra-red sensors and a monochrome CCD camera. The infra-red sensors, which are passive range-finding sensors like the sonars, were used to control the wall-following behaviour, as was described in section 4.3. Therefore they can be dealt with in the same way as the sonars.

The camera is a very different sensor. Images taken from the camera as the robot explores an environment provide completely different perceptions than the sonar sensors. This means that there is no guarantee that the same perceptions will be found to be novel when camera images are used instead of sonar data.

This chapter discusses a few ways in which images from a camera can be used as input to a novelty filter. The novelty filter based on the GWR network is used for all the experiments described in this chapter.

Figure 7.2 shows the steps that are taken for an image before it is used as input to the

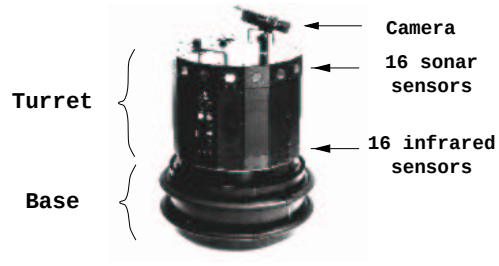


Figure 7.1: The Nomad 200 mobile robot. Note the position of the camera mounted on the top of the turret.

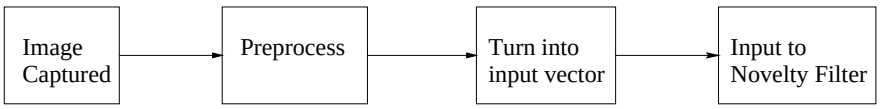


Figure 7.2: The sequence of operations used to process the raw images.

novelty filter. Sections 7.3 and 7.4 describe the techniques by which the images were preprocessed and transformed into input vectors for the novelty filter. Then section 7.5 compares the results of some of the different preprocessing techniques and input vector generation methods using experiments that parallel those that were used in previous sections.

7.2 Image Capture

The camera that is connected to *FortyTwo* is a monochrome CCD camera with a resolution of 480×200 pixels. The camera is mounted on top of the turret and has fixed angle and orientation relative to the turret. Therefore, the direction in which the camera faces can only be altered by rotating the turret. The focusing is also manual, so once the experiment was begun, the focus could not be changed. This meant that not every image was exactly in focus.

One question is, in which direction should the camera point to observe the scene? For example, the camera may face down the length of the corridor in the direction that the robot is travelling, or towards the wall that the robot was following. An alternative that has been used by other researchers such as Franz et al. (1998) is to use a parabolic mirror mounted above the camera with the camera facing the mirror, thus generating a full 360° image. As a parabolic mirror was not available, it was not possible to use this option.

If the camera faced forward then complex images with many features in them were produced. The same feature would be seen over several shots as the robot moved towards it, and a lot of complex preprocessing would be required to parse the image. This is still a very large area of research (Lacey et al., 1995; Sonka et al., 1993). The aim of this thesis is to demonstrate that the on-line novelty filter can be used to highlight novel features of an environment. This should be shown without relying on a large amount of complex preprocessing of the images. For that reason the camera was positioned so that images showed the appearance of the nearest wall, the one that the robot was following. The robot travelled between 20 cm and 60 cm from the wall. The camera was focused when the robot was approximately 40 cm from the wall. Some examples of typical images are shown in figures 7.3 and 7.4.

7.2.1 Seeing the Same Image Twice – The Visual 'Magnet'

If the robot stops in the same position twice, different images will be seen if the orientation of the camera is even slightly different the two times. In order to avoid this problem, an algorithm was developed that attempts to ensure that the camera always sees the same image from the same location, or at least quantises the number of possible images that can be seen at any location. It is called the visual magnet because of the fact that the robot is 'drawn' in a particular direction.

The algorithm takes the current image, uses an edge detection algorithm (see the discussion of different edge detection techniques in section 7.3.2) to find vertical edges in the image and then computes a histogram of these edge points by counting the number of pixels in each column of the image that are part of an edge. The algorithm then directs the robot to turn its turret so that the centre of the image is positioned at the point where the maximum of the histogram is. At this point a new image is taken and used as the basis for the input to the novelty filter, after the processing steps described in the next sections. The wall-follower is independent of the direction of the turret, so that the movement of the camera does not affect it.

As the camera can only be moved in the horizontal plane, the image is only centred for this histogram. With a pan and tilt camera it would also be possible to centre the image in the vertical plane. This algorithm could also be applied to any other problem where the number of directions that the camera should point in needs to be quantised.

7.3 Preprocessing of the Images

Although it was suggested above that as little preprocessing as possible should be performed on the images, it was discovered that it is necessary for some image enhancement to be performed owing to the poor quality of the images. This section describes three techniques that were applied successfully.

7.3.1 Histogram Equalisation

The images captured by the camera were typically of low contrast. This made it hard to recognise particular features of interest. One of the most common ways to deal with this is to use histogram equalisation (Gonzalez and Wintz, 1987), which flattens the peaks and enlarges the minima of the histogram. The technique works by computing a transformation function that attempts to make the histogram uniform over all values of image intensity by rescaling the inputs.

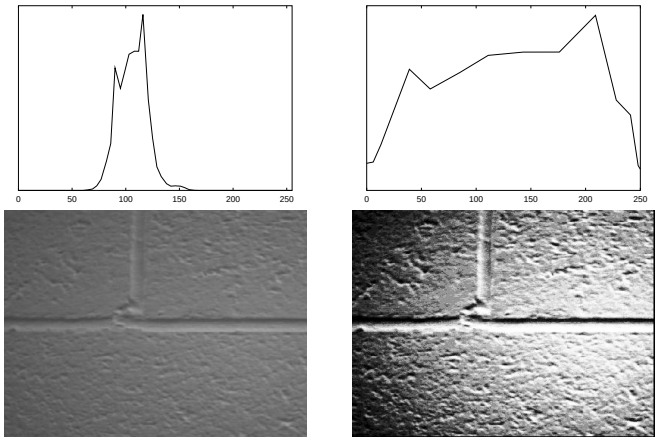


Figure 7.3: *Top: Left:* A typical histogram of image intensity against greyscale values before histogram equalisation. *Right:* The same histogram after histogram equalisation (scales are the same). *Bottom:* The images that produced the histograms.

Figure 7.3 shows the effects of applying the algorithm. On the left of the figure a typical image captured by the camera is shown, together with the histogram of image intensity. It can be seen that the histogram is sharply peaked and does not cover the whole range of intensities. On the right the histogram and corresponding image are shown after histogram equalisation. The image is clearer and the histogram is spread over the whole range of image intensities.

7.3.2 Edge Detection

Detecting edges within an image is a very useful form of local preprocessing. An edge is shown in an image by a discontinuity in image intensity across the pixels. Thus, a pixel is part of an edge if the intensity changes abruptly from adjacent pixels.

The most common way to detect an edge is to approximate the first and second derivatives of the intensity function by computing the difference between successive pixels. The first derivative is zero when the image intensity is constant (i.e., has the same grey level over several pixels) and takes a non-zero value when a change occurs. The second derivative is therefore zero at all points except where the grey level first begins to change, and where it stops changing. Thus, edges in an image can be detected by examining the magnitude of the first derivative, and the sign of the second derivative describes whether the edge pixel is the one on the light side of the border or the dark side.

Two different techniques were tried in the work described here. The first, and simplest, is the Laplacian operator

$$\nabla^2(x,y) = \frac{\partial^2 g(x,y)}{\partial x^2} + \frac{\partial^2 g(x,y)}{\partial y^2} \tag{7.1}$$

where $g(x,y)$ is the grey level of the pixel at coordinates (i, j) in the image. This filter can be computed using a single mask $\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$, which is run over each point in the image, providing an approximate value for the strength of edge at that point. Larger filters (5×5 and 7×7) can also be used, but the 3×3 filter was found to be sufficient for the purposes of this work. This filter is invariant to rotations and thus provides a measure for the strength of an edge without any indication of the direction. Unfortunately, it is also very sensitive to noise.

A more complicated system was therefore used. This system measures the first derivatives of the image intensity instead, using a set of masks, one for each direction of line. One of the simplest sets, the Sobel operators, look for horizontal and vertical lines, computing $\frac{\partial g}{\partial x}$ and $\frac{\partial g}{\partial y}$. Any other line will have horizontal and vertical components, and the direction of the line can be computed using $\tan^{-1}\left(\frac{\partial g}{\partial x} / \frac{\partial g}{\partial y}\right)$. The 3×3 Sobel operators are:

$$\begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} \text{ and } \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}.$$

The filters are run over the image and the strength of the edge at each point computed using

$$\sqrt{\left(\frac{\partial g}{\partial x}\right)^2 + \left(\frac{\partial g}{\partial y}\right)^2}. \tag{7.2}$$

If the value of this is greater than some threshold then the point is declared to be an edge, and hence the pixel is set to be white, otherwise it is set to be black. In this way the image is considerably simplified, as each element is simply a binary value, edge or not-edge. Figure 7.4 shows the effects of three different thresholds on the edge classification using the Sobel filter described above. It can be seen that setting the threshold too high means that some features are lost, whereas setting it too low means that many pixels that are not edges are included.

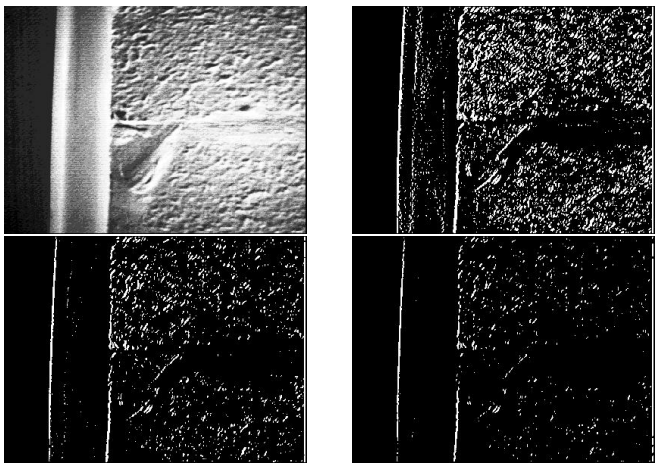


Figure 7.4: The effect of using different thresholds for edge detection. *Top: Left:* The original image. *Right:* Using a threshold of 100. *Bottom:* Thresholds of 150 (*left*) and 200 (*right*).

There are other techniques for finding edges, such as the Canny edge detector (Sonka et al., 1993) and the Hough Transform (Hough, 1962; Haralick and Shapiro, 1992). During investigations it was found that the additional complexity introduced through using these methods was too great to justify the better edge detection that resulted.

7.3.3 Reducing the Area of the Image

The size of the image, and hence the amount of storage space that is required for each image and the size of the input vector to the novelty filter, can be reduced in two ways. The first is to coarse-code the image, for example by making each pixel in the image be the median average of those around it. This results in a smaller image, but can mean that many features of the images are destroyed, for example, edges are smeared. It is this approach that is taken here, when the image size is reduced it is by coarse-coding.

The other way is to only store some part of the image, for example by having a focus of attention (fovea) that could be centred on some 'interesting' part of the image, such as the place where the histograms of horizontal and vertical edges intersect. Something similar is used in the visual systems of animals (Leow and Miikkulainen, 1991; Sengpiel and Hübener, 1999), where only some parts of the visual field are processed extensively, with many parts receiving only cursory attention.

7.4 Producing an Input Vector from an Image

After preprocessing, the image needs to be input to the novelty filter. For this, the two-dimensional image has to be turned into a one-dimensional array. The simplest approach is to concatenate each row of the image, one after the other. However, by doing this structure is lost, since pixels that are neighbours in a column are now far apart.

One way of avoiding this problem is to abstract away from the image completely. For example, histograms of the image can be used in place of actual elements of the image. This is described in section 7.4.1. Another alternative is to use only some pixels from the image, and to choose them in such a way that some structure from the image is kept. One possible way of doing this is described in section 7.4.2.

Finally, sections 7.4.3 and 7.4.4 describe two further approaches. Both attempt to generate input vectors that contain information about higher-order features of the image. The first generates a number of filters based on the principal components of blocks of the image, while the second looks for areas that are made up of similar textures.

7.4.1 Histograms of Edges

The edge detection techniques described in section 7.3.2 produce a binary coded version of the image. This image could then be used to produce an input vector for the network. However, this is little better than just using the image itself. Instead, histograms are generated by counting the number of edge pixels in each row and column of the image, and it is these histograms that are used as input.

The complete process is that the image is captured, the edge detection filters are applied and then the image is scaled using coarse-coding to reduce the image by a factor of 5. The histograms in the $x-$ (96 elements) and $y-$ (40 elements) directions are then computed, and the two concatenated to make a 136 element input vector. This is normalised and then used as the input vector.

7.4.2 An Image 'Fingerprint'

The most simple technique used is to attempt to maintain some of the structure of the image and reduce the size of the input vector by only using some pixels from the image and ordering them so that pixels that are close together in the image are close together in the input vector. The choice of which pixels is an important one. To try and exploit the local properties of the image, and assuming that the visual 'magnet' had made the centre of the image the most useful part, a spiral (shown in figure 7.5) was used. In the spiral most elements of the input vector represent pixels at the centre of the image (which is where the histogram of edges is maximum because of the visual magnet), but some elements from the edge are also included.

A number of different shapes were investigated, together with different lengths of input vector. The spiral performed the best, and the number of elements of the input vector that were included does not appear to be crucial. In section 7.5 the results reported are for an input vector of size 100.

7.4.3 Using Principal Component Features

Principal Component Analysis (PCA) (see section 2.8.5) offers one way of reducing the complexity of the input while retaining the important features. Sanger (1989) proposed using PCA as a method of image compression. The image is partitioned into non-overlapping 8×8 blocks and the algorithm operates on these partitions. Each of the 8×8 blocks of the image is used as input (after concatenating the successive rows of the block) to a single layer neural network that learns using the Generalised

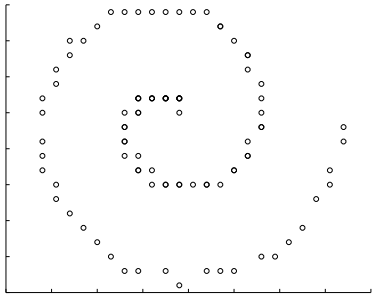


Figure 7.5: The spiral pattern that was used to generate the 'fingerprint' input vector to the novelty filter.

Hebbian Algorithm (GHA) and converges to the first M eigenvectors of the input correlation matrix, where M is the number of output units (Sanger, 1989). The learning rule is shown in vector form in equation 7.3:

$$\Delta \mathbf{W}(t) = \gamma(t) \left(\mathbf{y}(t)\mathbf{x}^T(t) - LT \left[\mathbf{y}(t)\mathbf{y}^T(t) \right] \mathbf{W}(t) \right), \tag{7.3}$$

for N-dimensional input $\mathbf{x}$, $M \times N$ weight matrix $\mathbf{W}$ and M-dimensional output $\mathbf{y} = \mathbf{W}\mathbf{x}$. $LT[\cdot]$ denotes the lower triangular matrix where all elements above the diagonal are set to 0.

In the image coding application the GHA algorithm is used to learn eight principal component filters (comprising 8×8 blocks) that best represent the image, by using all the blocks from the image as inputs to the network repeatedly until the network converges (in the experiments reported here, the algorithm ran over the entire image 200 times with a learning rate of $\gamma = 0.002$). The 8×64 weight vector then represents eight 8×8 principal component filters. An image is coded by partitioning it into 8×8 blocks and presenting each block to the trained network to generate eight coefficients. These coefficients encode how closely the current block resembles each of the filters generated by the GHA, and can be used to recreate the picture.

Sanger shows that the system is capable of some generalisation, as filters extracted from one image are used to encode a very different image, although this feature depends upon the statistical properties of the two images.

It is the eight coefficients generated by the trained network that are used as the components of the input vector for the novelty filter. In use the following steps are taken:

- When the first image is seen

  - Partition the image into 8×8 blocks
  - Generate 1×64 inputs from the 8×8 blocks
  - Train the network on these blocks using the GHA
  - Store the weights

- For each successive image
  - Partition the image into 8×8 blocks
  - Generate 1×64 inputs from the 8×8 blocks
  - Present each input to the trained network to generate coefficients
  - Use the coefficients of each block as elements of the input vector

Using this method assumes that the first image is representative of the images, and so is suitable for generating the principal component filters. In many cases this will be true as most parts of an environment look similar. In cases where it is not so, many perceptions will be found to be novel as the filters will not be a good approximation to the input space. This could be dealt with by producing filters taken from several images and either picking the most representative, or taking some form of average of them. However, it would not be possible to perform this task on-line and so it is not used here.

7.4.4 Textural Energy

The local textural energy filters of Laws (Haralick and Shapiro, 1992) are another way in which an image can be turned into an input vector. For each element of the image a number of texture features are computed using a set of 5×5 masks that are run over the image. This technique was used with some success by von Wichert (1996) before images were quantised by means of a hierarchically structured set of SOMs.

This technique was not found to work well for novelty detection, and so the results of using it are not shown below.

7.5 Experimental Results

Experimental Technique

As far as possible, the experiments reported in this section parallel those described in previous sections, such as 4.3 and 6.3. As there, the robot used a wall-following

behaviour to maintain a trajectory close to the wall (between 20 cm and 60 cm), which was followed for 10 m, with the robot stopping regularly to record a perception and present it to the novelty filter. However, rather than presenting an average of the sonar readings over the last 10 cm of travel, the robot stopped every 20 cm and an input vector generated from an image taken at the current place was presented. A distance of 20 cm was used so that the images overlapped partially, but were not too similar. In line with the previous experiments, several runs in each environment were used to train the novelty filter, and the output of the filter was recorded for each input along the route.

A large number of different combinations of preprocessing and input vector generation techniques were tried, together with different values for the insertion threshold in the novelty filter. The results that are reported are the results that appeared by eye to be the best for the three input vector generation techniques. That is, the filter learned a representation of the environment over several runs and then reliably detected certain features to be novel.

In all cases the novelty filter takes significantly longer to learn an accurate model of an environment than it did with sonar inputs. For example, only on the fifth learning pass is the whole environment found normal using inputs taken from the camera, while for the sonar data it was the third run. There are four principal reasons for this. The first is that the size of the input vector is considerably larger, so that there are more weights to train, but additionally the perceptions are more complex and also noisier. Finally, the robot makes fewer perceptions of each environment during a run with the camera, so that the filter received fewer inputs during each training run.

Histogram of Edges

As in section 4.3, the first experiments consisted of the novelty filter learning a representation of environment A over a number of training runs and then being tested in environment A* (the same environment, but with a door in the corridor opened – the environments are the same as were used previously, and can be seen in figure 4.5 and at the top of each of the result figures). Figure 7.6 shows the output of the novelty filter when the input vectors were made using the histogram of edges described in section 7.3.2. For this method, an edge threshold of 150 was used to generate a binary image, which was coarse-coded to be 96×40 pixels (from 480×200, a five times reduction). From this, horizontal and vertical histograms were generated and the two histograms were concatenated to form a 136 $(96+40)$ element input vector. This input

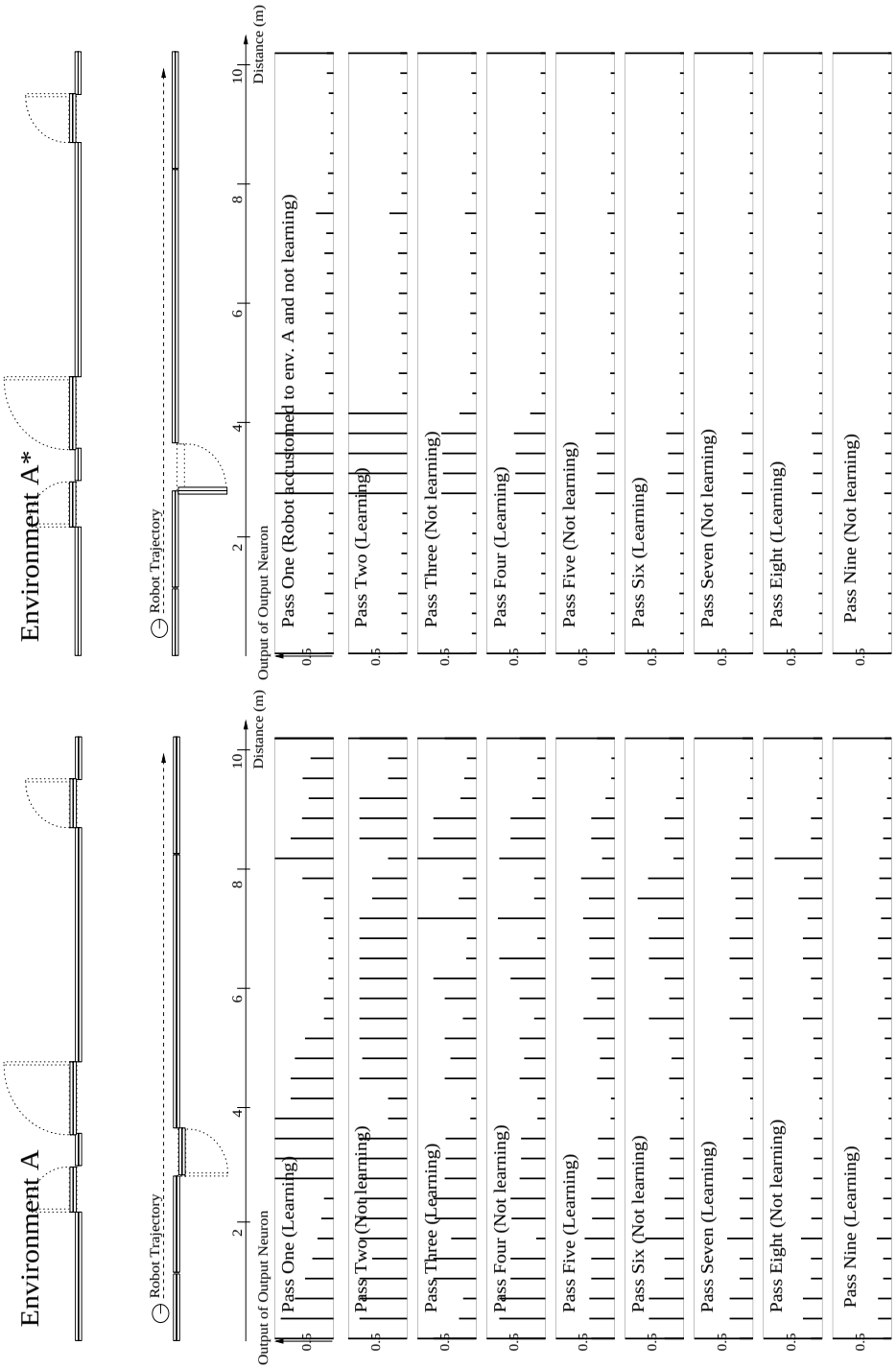


Figure 7.6: The results of the first experiment with the camera using the input vector generated from the histogram of edges. *Left:* The untrained novelty filter learns about environment A. Several learning runs are required before the network settles down. *Right:* When the trained network explores in environment A* the only feature that is found to be novel is the open doorway. These perceptions are then learnt.

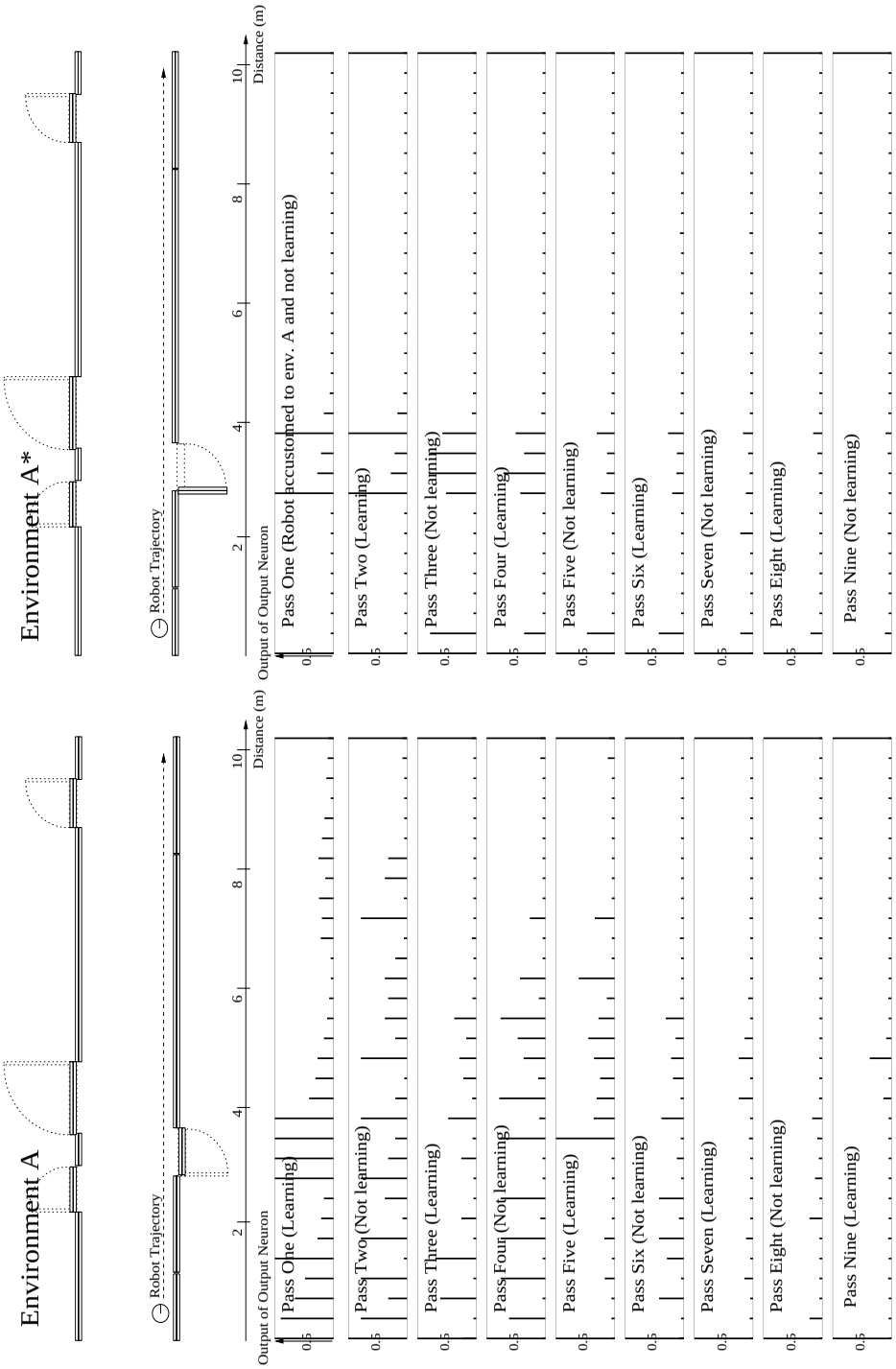


Figure 7.7: The results of the first experiment with the camera using the input vector generated using the fingerprint technique. *Left:* The novelty filter learns about the new environment faster with this technique than with the histogram of edges. *Right:* Again, only the perceptions of the doorway are found to be novel, and they are learnt rapidly.

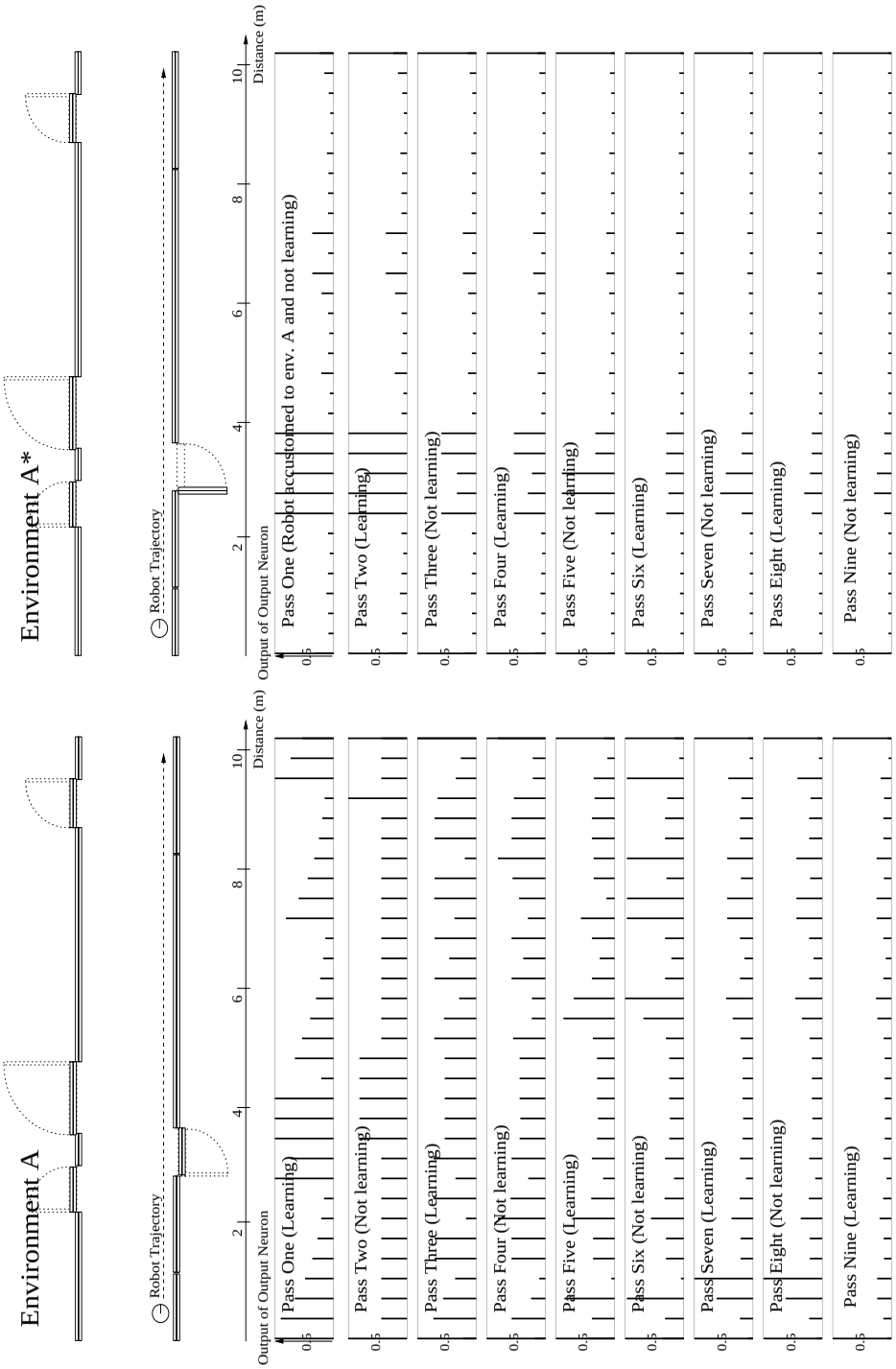


Figure 7.8: The results of the first experiment with the camera using the input vector generated using principal component filters. *Left:* The novelty filter takes a long time to learn about the new environment using this technique, and finds many perceptions very novel in pass six. *Right:* The perceptions of the doorway are again found to be novel, and the filter learns to recognise these perceptions.

vector was then normalised so that all inputs lay on the 136-dimensional unit hypersphere. An insertion threshold for the GWR network of 0.7 was used. These were found to be the optimal parameters through off-line testing.

The left of figure 7.6 shows that the network learns to recognise images of the wall from the outset, and finds the appearance of the door to be novel. Then, only the second crack in the wall is found to be highly novel during the first run. However, in pass two, where the network is not learning, a lot of novelty is found. This is probably because each of the images in this run is liable to be slightly different to those seen in the first run owing to a slight change in starting position. During the next few passes the filter learns, and in pass nine (after the fifth learning pass) it can be seen that no perceptions are found to be novel.

On the right of the figure the responses of the network when the robot explores environment A* after the filter was trained in environment A. It can be seen that only the perceptions of the doorway are novel, which is what should be found. Only in pass seven does the filter stop finding these perceptions to be novel, which means that three learning runs are required to habituate to the image of an open door.

The Spiral Fingerprint

Figure 7.7 demonstrates the results of the same experiment when the spiral fingerprint shown in figure 7.5 was used, and figure 7.8 when Sanger's principal component filters are used to encode the image. Again, an insertion threshold of 0.7 was used for the GWR network.

It can be seen in figure 7.7 that the fingerprint technique allows the novelty filter to learn a good representation of the environment relatively quickly, very little is found to be novel even by pass three. The right of the figure also shows that the doorway is found to be novel, and that this is learnt fairly quickly, after two training runs the novelty filter stops responding to these perceptions.

Principal Component Filters

The principal component filter technique shown in figure 7.8 takes longer than the other to learn, and finds more features to be novel. Most of this is probably due to the fact that the input vector is considerably longer – 1600 elements as opposed to 100 for the fingerprint. In addition, there are several perceptions that are found to be novel for no apparent reason. For example, in passes five and six, several perceptions of blank

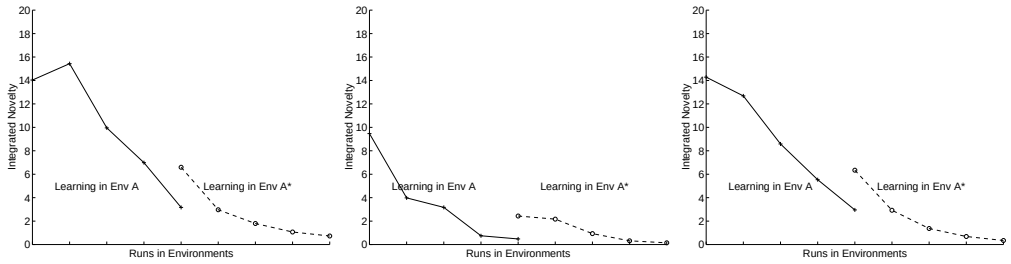


Figure 7.9: Graphs of how the novelty changes when the novelty filter is trained in environment A and then environment A* using the three different input vector generation techniques. *Left:* Histogram of edges, *centre:* fingerprint, *right:* principal component filters.

wall halfway through the environment are found to be novel. However, again only the features of the doorway are highlighted when the robot explores environment A*, and this is learnt satisfactorily.

Comparing the Different Input Vector Generation Techniques

In figure 7.9 the total amount of novelty found during each learning run, first in environment A and then in environment A* is plotted for each of the three input vector generation methods. It can be seen that the characteristics of the three curves are similar – the amount of novelty decreases over the learning runs in environment A, increases a little bit when the robot moves into environment A*, and then decreases as the filter learns about that environment. The most interesting observation is that for the histogram of edges, the second run finds more novelty than the first one. The reason for this is unclear, it does not happen for the other runs.

It is clear that the fingerprint technique learns fastest. This is probably because the input vector is considerably smaller for this method. The amount of novelty found after training in environment A is much less than using the other techniques, and less novelty is found in environment A* than when using the other methods. For this reason it is this technique that is used in preference to the others.

Further Experiments Using the Spiral Fingerprint

Figure 7.10 shows the results of the novelty filter with the fingerprint input vector when it is tested in further environments. In figure 7.10 the network that was trained in environment A (as shown on the left of figure 7.7) is tested in two further environments, B and C. As in the experiments where the environment was perceived through sonar, environment B is a similar part of the environment to environment A, and environment

C is very different. Again, see figure 4.5 for pictures of the environments. In particular, environment C will look very different using visual input, as the walls are made of considerably smaller bricks than environments A and B.

It can be seen that the filter recognises all the areas of environment B that are wall, but finds the perceptions of doorways to be novel. There are two reasons for this. One is that the doorways are deeper inset, which also affected the behaviour of the novelty filter with sonar, and the other is that the doorposts are painted black, whereas the doorposts in environment A were painted white. For environment C it can be seen that almost every perception is found to be novel, but that the filter learns a good representation over two learning passes through the environment.

More Comparisons

The integrated novelty in the visual inspection experiments in all four environments are shown in figure 7.11 for the edge detection method, figure 7.12 for the fingerprint and figure 7.13 for the principal component filters. In each figure the graph on the left shows the robot learning first in environment A, and then in environment A*. It can be seen that once the robot has learnt about environment A, very little in A* is novel. The middle graph shows how the amount of novelty increases when the robot explores environment B after learning about environment A, and finally the graph on the right shows how the novelty increases when the robot begins to explore environment C.

In each case, about the same amount of novelty is found in environment B as in environment A, and more novelty is found in environment C. This is because learning about environment A does not affect the learning in environment C, because to the camera these environments are very different. It can be seen that the novelty decreases fastest for the fingerprint technique, and also decreases to the lowest levels for this technique.

7.6 Summary

This chapter has described methods by which the novelty filter can receive data captured by a camera as input. There are two stages of processing required to transform a raw monochrome image into an input vector. First, the image is enhanced to make features clearer and then an input vector is generated from the enhanced image.

Four different techniques for generating input vectors were described in this chapter. The first two used a histogram of the number of elements in each row and column

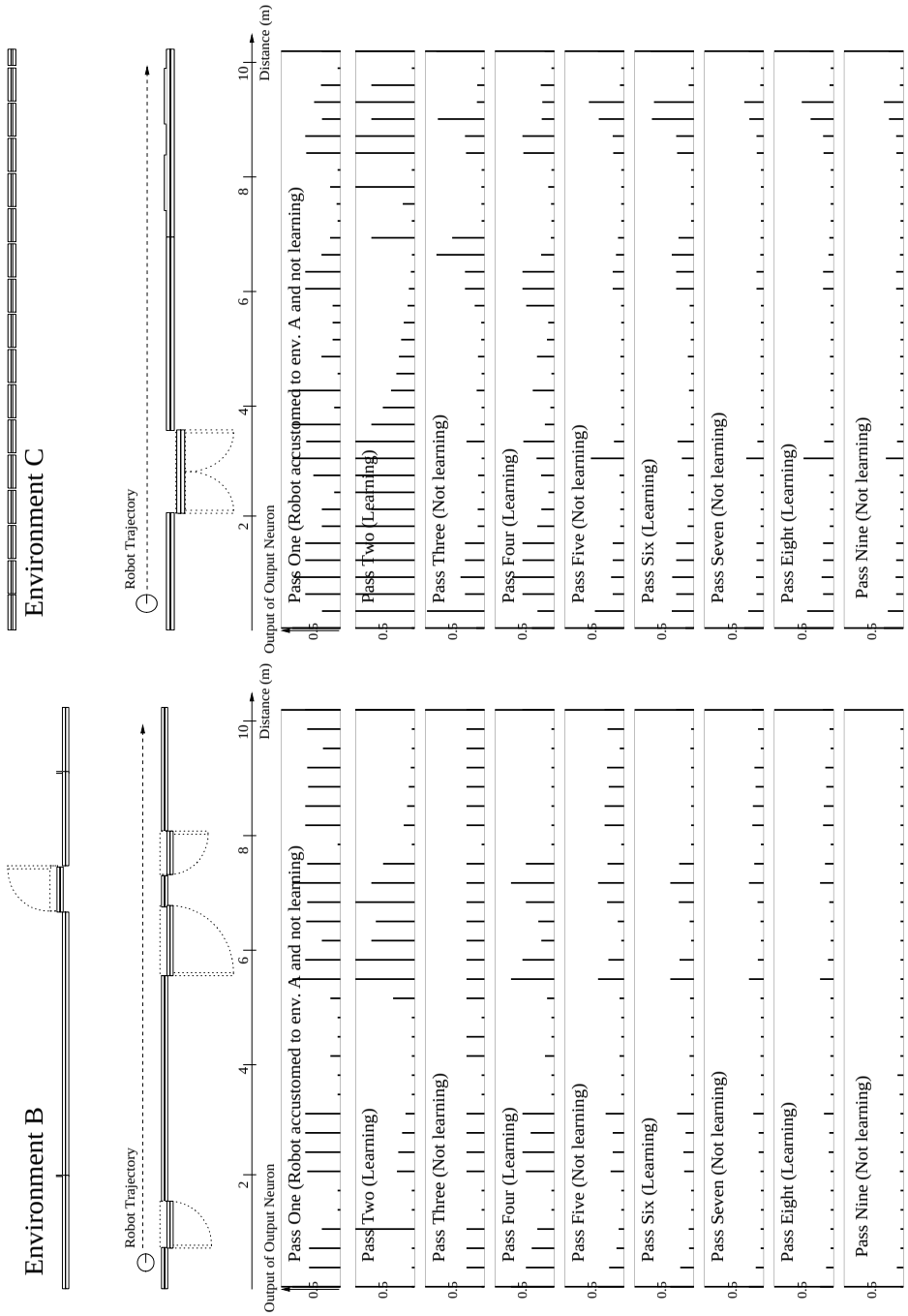


Figure 7.10: The second and third experiments using visual input, and generating the input vector using the fingerprint technique. The novelty filter trained in environment A was used to evaluate two further environments, as in the experiments using sonar perceptions. *Left:* Mostly it is the doorways that are found to be novel, and the wall is recognised. *Right:* Here the wall, which has a very different appearance as it is made out of smaller bricks, is found to be novel initially, although it is learnt after two training runs.

that were considered to be part of an edge and a 'fingerprint' of pixel intensities chosen from the image (the pixels were taken from a spiral with the focus at the centre of the image). The other two methods aimed to maintain structure within the image. The first of these was to generate a set of filters found by training a network that extracted the principal components of small blocks from an image. The combination of these filters required to recreate each block in the image was then used to generate an input vector to the novelty filter. Finally, a set of local textural energy filters were described, where a set of filters are passed over the image looking for areas of constant texture within the image. This method was unsuccessful for the problem investigated here and so the results were not reported.

These techniques were then tested on images collected by the robot as it explored a number of different environments. The environments were the same as were used for the experiments with the novelty filter and sonar data in sections 4.3 and 6.3. It was found that the first three of these techniques performed reasonably well, as the novelty filter learned a reliable representation of environment A over the course of five training runs and then detected only the newly open doorway of environment A* to be novel. As the fingerprint technique learnt fastest, the results for this technique are shown in detail for explorations of environments B and C, too.

The techniques described above are simple ways of generating input vectors from images. This is necessary as the techniques have to operate on-line while the robot explores and so they need to be fast. It is possible that extracting features from the image and learning about them would be a fruitful line of research, but that is beyond the scope of this work.

It has been shown that the novelty filter based on the GWR network is capable of dealing with complex input vectors and learning a representation of the inputs through this approach. The next chapter gives some further experiments designed to demonstrate how the novelty filter can be used in practise for an inspection task.

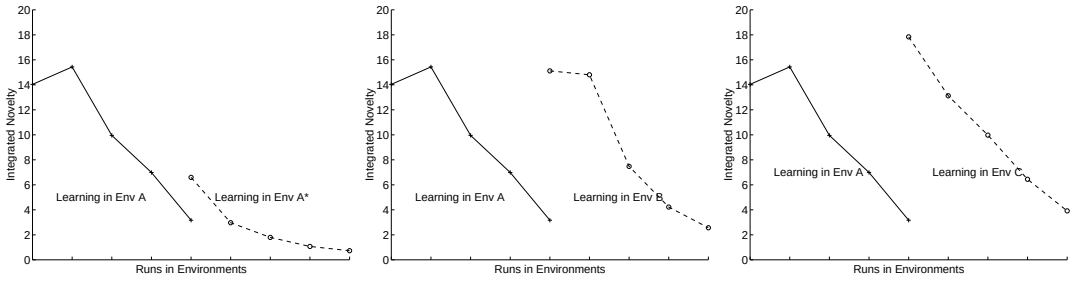


Figure 7.11: Integrated novelty for the histogram of edges across the different environments.

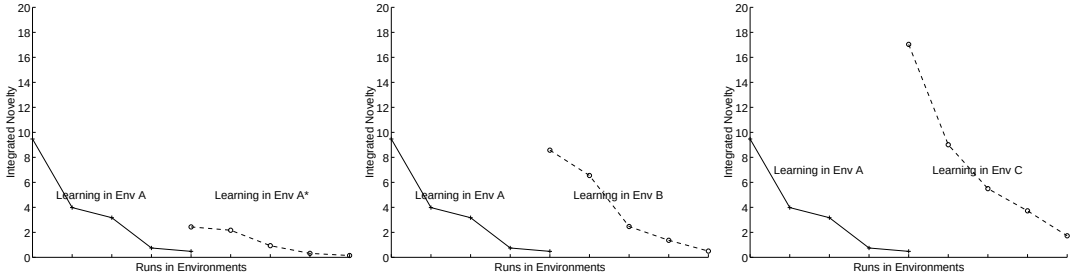


Figure 7.12: Integrated novelty for the fingerprint across the different environments.

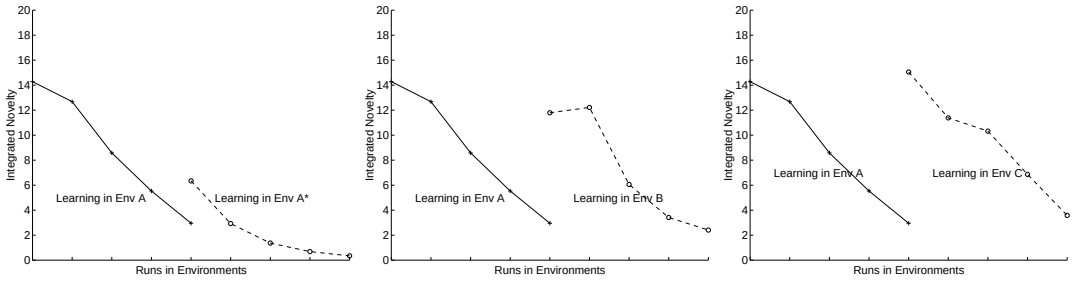


Figure 7.13: Integrated novelty for the principal component filters across the different environments.

Chapter 8

Further Robotic Experiments with the Novelty Filter

An extension to the novelty filter algorithm is described that allows the novelty filter that best matches the current environment to be selected from a battery of pre-trained novelty filters. This means that the filters can be used autonomously in a variety of environments.

8.1 Introduction

This chapter concludes the description of experimental investigations into the GWR novelty filter based on habituation. It details two further sets of experiments with the filter, using perceptions from the robot's sonar sensors as it explores different environments.

The first set of experiments shows that a bank of novelty filters trained in different environments can be used in tandem to identify an environment and therefore to select which novelty filter should be used to evaluate that particular environment. The principal motivation for this investigation is that the robot may have to inspect an area that contains several different environment types. For example, an office building contains corridors and rooms amongst other environments and a sewer could well have pipes of different thicknesses made from a variety of materials. As a feature that is common in one environment could be novel in another, it is necessary to use different novelty filters and select which to use at the current time.

The second set of experiments demonstrates that the novelty filter will select as novel interesting features of an environment. In this experiment the novelty filter was

trained on a set of environments that contained every feature except for doors and doorposts. Once the filter had learnt a representation of these environments it was allowed to explore two environments that contained doorways, and it is demonstrated that only the doorways were found to be novel.

8.2 Selecting a Suitable Trained Novelty Filter

8.2.1 Motivation

The technique described in this section demonstrates a way in which the novelty filter can be used in a number of different environments. This is potentially useful for inspection tasks, as when the robot explores it could experience many different environments. An example would be an office-type environment, where there are corridors, rooms and open spaces. What is normal in one environment could be novel in another, and so it is not sufficient to train one novelty filter to recognise every perception. For example, seeing chairs in an office is very common, but seeing them in a corridor is a novel event in most buildings.

This problem is dealt with by using a number of different novelty filters, with one trained in each environment, and describing a mechanism by which the correct novelty filter can be selected while the robot explores. For such a mechanism to be useful the system also needs to be capable of recognising that the particular environment explored is not one for which there is currently a model and therefore a new novelty filter should be added and trained for this environment. This allows the whole system to grow as the robot gains experience of other environments.

In addition, the system enables a further investigation of the properties of the novelty filter, in particular enabling the quantification of the performance of the filter.

8.2.2 A Description of the Algorithm

A vector of 'familiarity indices' is used to keep a record of how familiar each of the different trained novelty filters finds the current perceptions of the robot. This familiarity vector (with one element for each of the m trained novelty filters) is updated after each perception has been presented to all of the novelty filters, each of which has produced a novelty value n.

All of the elements of the familiarity vector are initialised to be $1/m$. For each input to the novelty filters the following steps are taken:

- compute the novelty value n_i (i.e., the output of the novelty filter) for the current filter, i
- update the element of the familiarity vector $\mathbf{f}$ for that network:

$$\mathbf{f}_i = \mathbf{f}_i - c \times n_i, \tag{8.1}$$

  where c is a scaling constant
- update all the other elements so that the sum of the elements remains normalised:

$$\mathbf{f}_j = \mathbf{f}_j + \frac{c \times n_i}{n_i - 1}, \; \forall j \neq i \tag{8.2}$$

- repeat for all the other novelty filters

In the experiments reported in the next section a value of $c = 0.1$ was used. Investigations showed that the value was not critical, although obviously it does affect how quickly the familiarity vector responds to inputs that the filters find to be novel.

Once the robot had travelled through the environment a decision was made about which environment it was. If one element of the familiarity index was significantly larger than the others (i.e., one familiarity index was above 0.7), then the corresponding environment was taken as the one being explored. However, the algorithm also stores the accumulated novelty of each novelty filter as the robot explores it. If the best-matching novelty filter has very high accumulated novelty then it could be that it is not actually a good match to the inputs, merely a better match than the others. In this case a new novelty filter should be made and trained.

In the case where no environment is obviously more familiar than any of the others it is assumed that this is because the robot has just explored a novel environment. The accumulated novelty should also be high in this case. If the environment was novel then it would be suitable to generate a new filter and further explore the environment, training the filter. In this way the algorithm could extend itself as required.

8.2.3 A Description of the Experiments

Four separate GWR-based novelty filters were trained, one in each of environments A, A*, B and C (as described in section 4.3 and shown in figure 4.5), so the elements of the familiarity vector were initialised to $1/4$. An insertion threshold of $a_T = 0.9$ was

used, as in section 6.3. The robot made three training runs in each of the environments, sampling the environment with its sonar sensors (as in previous experiments, presenting an average of the last 10 cm of travel every 10 cm). Each network was initially completely untrained, and after the three training runs the filter had stopped finding any features in the environment novel.

After training each of these filters the robot was exposed to an unknown environment. Five different environments were used for this purpose, each of the four used for training (A, A*, B and C) and a control environment that was completely different. This novel environment was part of the robot laboratory, which is wider than the corridors and has obstacles placed in the path of the robot, so that the perceptions were very different to those in the corridor environments.

Once the robot was placed in an environment it explored that environment using the wall-following behaviour to follow the wall to the right of the robot. As the robot travelled in this test environment the algorithm had to decide which environment the robot was exploring – one of the known environments or the control environment.

8.2.4 Experimental Results

Once the training in each environment had been performed the robot made five runs in each of the five environments A, A*, B, C and the control environment. During these experiments the novelty filter computed the familiarity index for each of the four trained environments and therefore decided which of the environments it was currently exploring. No prior information about which environment the robot was exploring was given. If none of the trained environments matched the inputs then it was assumed that the environment was novel.

Table 8.1 shows the familiarity vectors for each test environment averaged over the five runs. It can be seen that in each case the correct environment is selected (shown in bold). The way that the familiarity vector changes with accumulated evidence from the environment for one run in each of the five test environments is shown in figure 8.1. All the graphs in an environment were similar.

From table 8.1 it can be seen that the familiarity index for environment A is very high when the robot explores environment A. The top left image of figure 8.1 shows how the familiarity indices vary as the robot explores the environment. It can be seen that initially environments A and A* are equally likely, but that once the perceptions of the doorway are seen, A* becomes less likely. It is unclear why the index jumps suddenly after about 75 perceptions. It is possible that the second crack in the wall

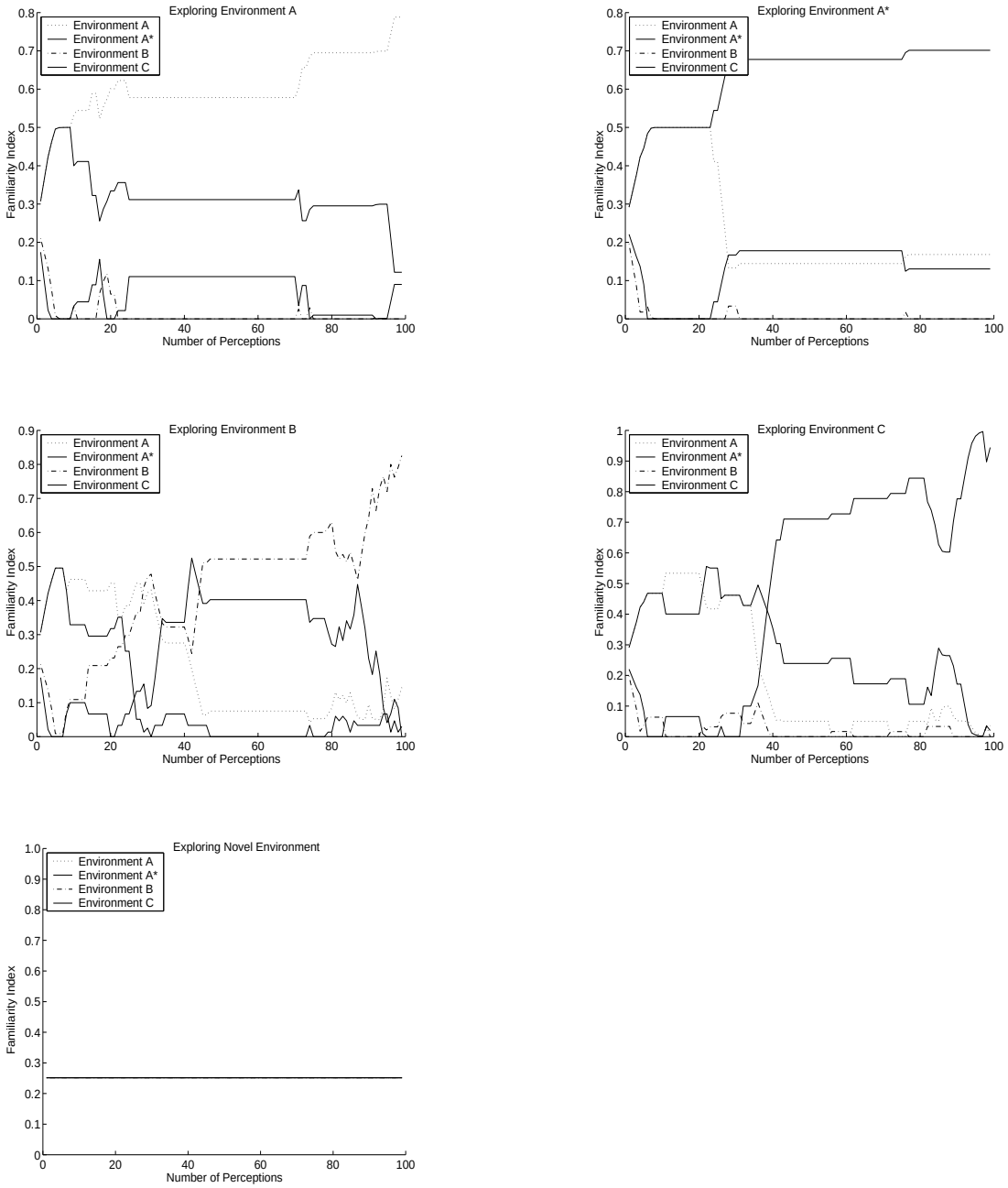


Figure 8.1: Plots of the familiarity indices for the trained novelty filters during sample runs in each of the five test environments. The $x-$axis shows the number of perceptions of the environment. As in other experiments, a perception was presented to the novelty filter for every 10 cm of travel. *Top left:* Environment A. *Top right:* Environment A*. *Middle left:* Environment B. *Middle right:* Environment C. *Bottom:* Novel (control) environment.

Train Env	Test Environment				
	A	A*	B	C	Novel
A	**0.809** (0.055)	0.042 (0.031)	0.115 (0.038)	0.026 (0.029)	0.255 (0.067)
A*	0.157 (0.150)	**0.723** (0.191)	0.014 (0.0128)	0.106 (0.103)	0.242 (0.073)
B	0.012 (0.009)	0.073 (0.078)	**0.899** (0.093)	0.016 (0.032)	0.245 (0.045)
C	0.068 (0.060)	0.102 (0.082)	0.034 (0.024)	**0.796** (0.099)	0.263 (0.102)

Table 8.1: The familiarity index for each of the environments for the different trained networks, averaged over five runs. Each table entry gives mean (standard deviation).

(see figure 4.5) was detected, and that this is only in the model of environment A.

Similarly, for exploring environment A* (shown on the top right of figure 8.1), environments A and A* are initially equally likely, but at the perceptions of the doorway the familiarity index for A* dominates and remains stable from then on.

For testing in environment B, table 8.1 shows that at the end of the run the algorithm is very confident in identifying the environment. However, the graph in figure 8.1 shows that it is only relatively late in the exploration that environment B becomes likely. This seems reasonable, since environments A and B are so similar. Initially environments A and A* are favourites and environment C is very unlikely. However, where the perceptions of the doorway are seen (which is more deeply inset than that in environment A) both B and C become more likely. Only when the second doorway and the boxes on the wall are seen at the end of the run does the familiarity index for B become the clear winner.

Similar effects are seen for environment C. Initially, all environments are about equal, but once the perceptions of the particularly deeply inset doorway are seen, the familiarity index for C quickly overtakes the others.

The bottom figure of figure 8.1 shows how the familiarity indices evolved when the robot explored the control environment, which did not have a trained novelty filter. In this case none of the environments became more likely, which is good, and this is borne out by the table. The fact that the control environment is very different to the trained environments may have helped this.

To rectify this last problem, experiments were performed using only three of the novelty filters trained for the last experiment, and then the robot was tested in all four environments. The results of two of these experiments are shown in figure 8.2. On the left, filters trained in A, A* and C are used while the robot explores environment B, and on the right filters trained in A, A* and B are tested in environment C. It can be seen that initially environment A is found to be very likely, but that around the perceptions

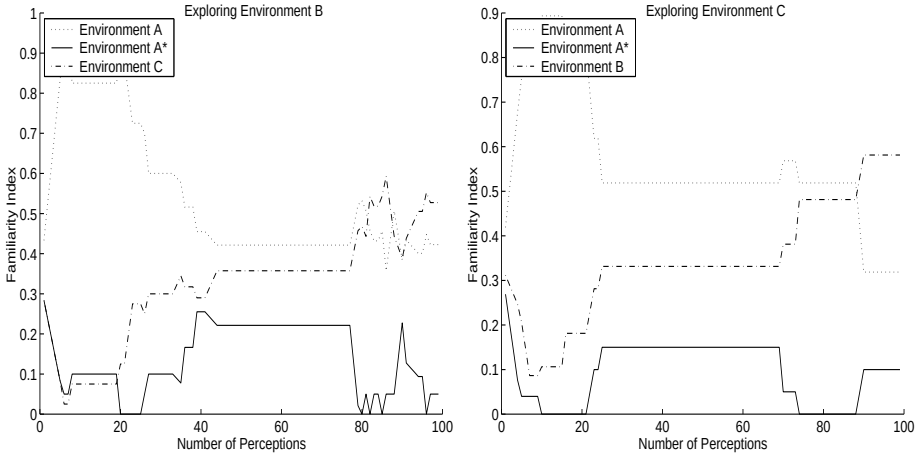


Figure 8.2: *Left:* Familiarity indices for the novelty filters trained in A, A* and C, for tests in environment B. *Right:* Testing on environment C after training on A, A* and B.

of the doorway this decreases, and C becomes equally familiar. From that point both of these environments remain level. A*, which contains the open doorway, is found to be very different. Similar results are found when testing in environment C.

8.2.5 Conclusions

The results demonstrated in this section have shown that the novelty filter based on the GWR network can reliably detect which of a set of previously unexplored environments it is in. This fact that the correct filter was selected 100% of the time suggests that the environments were sufficiently different for the filter to be useful. However, several of the environments (A, A* and B) are segments of the same corridor, and so the resolution of the filters with the sonar sensors is obviously high.

8.3 Highlighting Novel Features

The second experiment was concerned with demonstrating that the features that the novelty filter highlighted were interesting ones. For this experiment a novelty filter was trained on several sections of corridor made up solely of walls, with no doors in them, although every other feature of environments A, B and C was present, as far as could be ascertained. Again, a GWR network with insertion threshold $a_T = 0.9$ was used. Once the novelty filter had stopped responding to any of the perceptions found in these environments, the robot explored environments A and B.

Figure 8.3 shows the output of the trained novelty filter for each of these environments. It can be seen that the only features that are found to be novel are those that were not seen in the training environment, i.e., the doorways.

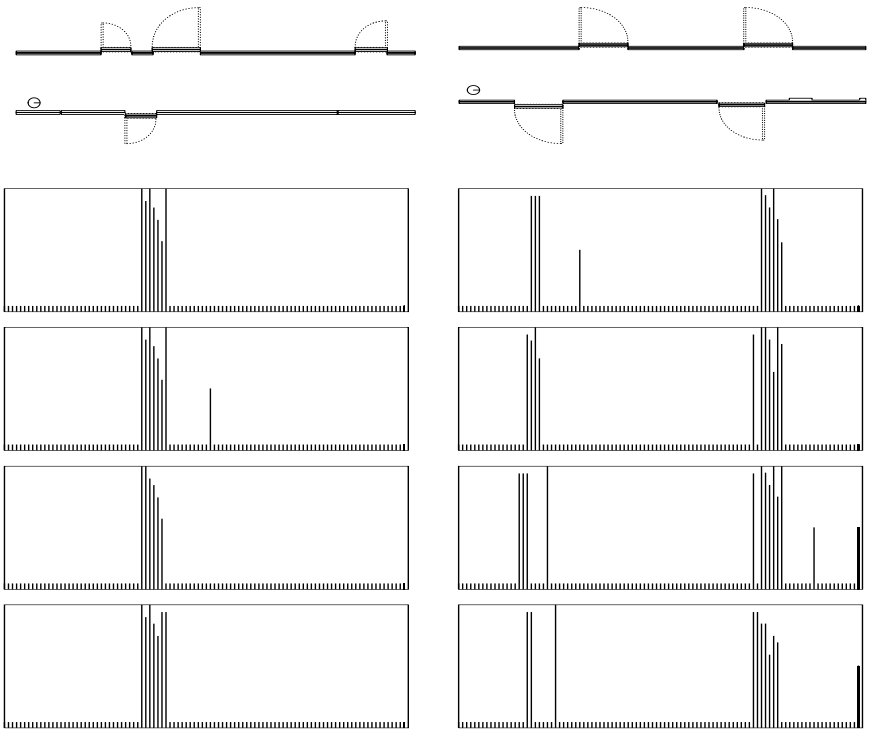


Figure 8.3: The output of a novelty filter trained to recognise sections of corridors that contain only wall when tested on environments A (left) and B (right) that contains doors. Only the perceptions of the doorways are highlighted as novel.

8.4 Summary

This chapter has provided the final experimental demonstrations of the novelty filter. The GWR-based novelty filter received the perceptions of the robot in a number of environments through the sonar sensors and performed two different tasks.

The first task (described in section 8.2) was to attempt to classify which environment the robot was exploring – one of several that had been seen previously and modelled, or a completely new one. This enabled the novelty filter to operate in a sequence of different environments and to select the correct model with which to evaluate the current environment without human intervention. It also demonstrated that the model

that the novelty filter learns can be made environment specific through sufficient exploration. Such a system could also be used for non-robotic applications where inputs from a variety of different tests could be presented.

Section 8.3 described the second experiment. This involved training a novelty filter across several environments that were deliberately chosen to exclude one common feature, in this case doors, during corridor exploration. Once the filter had learnt a representation of these training environments the robot travelled along two environments that contained doorways as well as many of the trained features and it was shown that only the doorways were found to be novel.

These two sets of experiments, together with the experiments described in previous chapters, have shown that the novelty filter based on habituation can be used to acquire a model of an environment on-line and simultaneously detect (and quantify) novel features with respect to the model learnt so far. It has been demonstrated that the novelty filter can be used in large environments through the use of the GWR network, and in section 8.2 it was shown that the system can also be used to receive inputs from several different environments and to select which environment, if any, best represents the current inputs.

Chapter 9

Summary and Conclusions

This concluding chapter provides an overview of the research and suggests areas that the work has opened up for further investigation. The objectives derived in chapter 1 are used to evaluate the research. Finally, a discussion of the research is given.

9.1 An Overview of the Research

This thesis describes a novelty filter that is based on habituation, a reversible decrement in response to a stimulus that is seen repeatedly without adverse effect. In contrast to previous research, as was discussed in chapter 2, this has meant that the novelty filter is suitable for on-line use, learning a model as data is presented and simultaneously testing each input with respect to the model learnt so far. This means that the filter can be used for on-line inspection tasks, for example, using a mobile robot to explore environments and detect deviations from the acquired model.

Chapter 1 began the thesis by discussing what novelty detection actually is and suggesting a number of areas where the ability to detect novel inputs was a useful activity for an artificial learning system. In particular, the usefulness of a robotic inspection agent was discussed, and this motivated the research described in this thesis. The aims of the research were introduced, together with a set of objectives that a useful novelty filter should be able to fulfill. These objectives are used to develop a set of assessment criteria by which the research can be evaluated. This will be considered further in section 9.2.

The desired objectives of the novelty filter were used in chapter 2 to evaluate novelty filters that have been proposed in the literature. It was found that none of the

novelty filters described were capable of fulfilling all of the criteria, in particular the requirements that a filter should operate on-line, quantify the amount of novelty in a perception and forget stimuli that are seen only rarely.

In chapter 3 a new novelty filter was introduced in abstract as an unsupervised neural network equipped with a set of habituating synapses, one for each node in the network, linking the nodes to an output neuron. An implementation based on the Self-Organising Map, the Habituating Self-Organising Map (HSOM), was introduced in section 3.3 and other networks that could be used as the basis for the novelty filter were discussed.

Using habituation to evaluate the amount of novelty in a perception has a number of advantages. The fact that the dynamics of the habitation equation follow an exponential decay means that the amount of novelty in a perception can be computed easily by looking at the strength of the synapse for the node that best matches the input. Other benefits of using habituation, such as the forgetting caused by dishabituation, where a previously habituated signal becomes stronger again, were also described. Finally, a possible problem with the HSOM was described; the fact that the SOM is not designed for continuous learning, and so when used for on-line robotic inspection the HSOM could be subject to saturation so that every feature is seen as normal.

Experiments with the HSOM were reported in chapter 4. The first experiment demonstrated how the novelty filter can be used to focus the attention of a robot equipped with passive light sensors onto the most novel features in its environment. This attention was demonstrated by the robot travelling towards the newest feature, a behaviour that was termed ‘neotaxis’. It was demonstrated that the HSOM could be used to satisfactorily categorise the different input signals used in these experiments and direct the attention of the robot towards the newest stimulus.

Further experiments in section 4.3 described how the novelty filter can be used for a simple inspection task. For these experiments a Nomad 200 robot equipped with 16 sonar sensors and using a wall-following behaviour to control its motion travelled through a number of different environments presenting perceptions of the environment from the sonar sensors to the HSOM-based novelty filter. The novelty filter learnt a representation of a training environment by travelling through it several times. Once the representation was accurate, so that no perceptions were found to be novel, the robot was moved to a different environment and evaluated the novelty of each perception in the new environment with respect to the model acquired previously.

The effects of using dishabituation to forget stimuli that have not been seen recently

was demonstrated in section 4.3.3. It was shown that the novelty filter would forget a perception that was not seen for a few hundred inputs, no matter how familiar it was beforehand, so that it was again found to be novel. Then, in section 4.4, the existence of the saturation problem for the HSOM was demonstrated by using data from a considerably larger environment than those used previously, a 200 m loop of corridor.

Chapter 5 introduced a solution to the scalability problem caused by the continuous learning requirements of robot tasks such as inspection. After an investigation of the literature on growing neural networks it was found that there was not currently an algorithm suitable for use as the basis for a novelty filter that can solve the problem of continuous learning. For this reason an alternative was proposed, where nodes were added into the map field whenever a suitable node was not currently present in the map. The network was termed the 'Grow When Required' (GWR) network. The GWR network also maintains neighbourhood connections between nodes that represent similar inputs by using competitive Hebbian learning. This enables the filter to generalise between similar inputs.

The behaviour of the network was analysed in section 5.5, and a number of demonstrations of the network on well-known (non-novelty detection) datasets were given, together with comparison results from other growing networks. It was shown that the GWR network performs at least as well as the other algorithms on all the tests, and usually finds the solution considerably more quickly. The perfectly topology-preserving nature of the algorithm was also demonstrated.

The GWR network was applied to the task of novelty detection in chapter 6. First, a simple toy dataset was used to demonstrate that the algorithm detected novelty successfully, and then the inspection experiments used to demonstrate the HSOM network were reproduced with the GWR network, as well as with novelty filters based on the Growing Neural Gas (GNG) and unsupervised RCE networks. The GWR network reliably detected those areas of the environment that were novel, and did not have a problem with noise, which caused a lot of problems for the RCE network. Furthermore, the GWR network produced efficient and accurate representations of the input space, generalising well and responding rapidly and accurately to novel inputs. The ability of the GWR network to scale with the complexity of the input set was demonstrated clearly using the 200 m loop of corridor that caused problems for the HSOM.

The end of chapter 6 demonstrated the use of the GWR network on three non-robotic novelty detection tasks that have been reported in the literature. The first was

a medical diagnosis task, identifying unspecified abnormalities in blood test results. The second problem was a machine fault detection task for ball-bearings. For both tasks the results were compared to the results reported by other researchers who used a novelty filter based on the Support Vector Machine. The performance of the two techniques was shown to be comparable, a very pleasing result, although there were differences in the responses of the networks. The third problem was similar to one used to demonstrate neural network validation on simulated data from oil pipelines.

Inputs from a very different sensor to the sonar sensors used previously were considered in chapter 7, with the new sensor being a monochrome CCD camera. Before the output from the camera could be given to the novelty filter an input vector needed to be created. Several methods of doing this were described in section 7.4, and results from the best three methods were presented. It was found that using a simple input vector generation method using a spiral 'fingerprint' worked well, and the results of using this system to investigate the environments described for the sonar inspection tasks were detailed, demonstrating that the novelty filter based on the GWR network can be successfully applied to a variety of inputs.

Final experiments were described in chapter 8. In the first of these an algorithm was described that enabled the robot to be sent out with a bank of novelty filters trained in different environments and to decide autonomously which novelty filter should be used for the current environment. This means that an inspection agent could explore a variety of different environments and detect novelty with respect to the expected perceptions of that environment. The experiment also demonstrated the reliability of the trained novelty filters.

Another experiment in chapter 8 trained the novelty filter on perceptions of corridor environments that were chosen carefully to ensure that no doors were seen by the robot, but that every other feature was seen. Once the novelty filter had acquired a good representation of these training environments the robot was allowed to explore two environments that contained these features, but also contained doors. It was demonstrated that only the doors were found to be novel.

These experiments were all designed to demonstrate the learning abilities and operation of the novelty filter based on habituation. It has been shown that the filter can operate on inputs from different robot sensors and learn models of several environments as the robot explores, and that the novelty filter can be used on-line to evaluate perceptions with respect to the model acquired so far. The novelty filter can also quantify how novel each perception is with respect to the model acquired so far. It has

been demonstrated that the filter performs well on datasets from other domains where novelty detection is a useful tool.

9.2 Aims and Objectives (Again)

In section 1.3 a list of aims and objectives for this research was given. These are reproduced here, and a set of assessment criteria are created, to allow the success of the thesis to be evaluated easily.

9.2.1 Aims

The overall aim of this research can be given in the following two sentences:

> To develop, and investigate the behaviour of, an adaptive novelty filter, suitable for on-line use, that quantifies the amount of novelty in a perception with respect to an acquired model, and then adds the knowledge of the perception to the model. The novelty filter will be applied to the perceptions of a mobile robot as it performs simple inspection tasks, first exploring a number of environments, learning a model of these environments and then highlighting deviations from the model.

9.2.2 Objectives

In chapter 1 a set of objectives for the novelty filter and the neural network on which it was based were given. These are reproduced below, and a selection of them are evaluated in section 9.2.3.

- The novelty filter should:
  1. successfully detect and learn about novelty in the inputs
  2. generalise between similar inputs
  3. operate on-line
  4. quantify novelty
  5. always find perceptions novel that are seen very infrequently
  6. forget about perceptions that are not seen for a long time
  7. scale with the size of the input set

8. deal with a wide variety of inputs
9. handle multiple models of different environments

- The experiments should demonstrate that the novelty filter can:

10. enable a robot to focus its attention on novel features
11. enable a robot to highlight novel features of an environment
12. highlight novelty in other datasets

- The neural network basis for the novelty filter should:

13. be capable of continuous learning
14. learn economical and efficient representations of the input space
15. preserve the topology of the input space

9.2.3 Assessment Criteria

The table on page 192 lists a set of assessment criteria generated from the objectives, by which the research can be evaluated. The final column of the table gives section references for the place where that criteria was most clearly demonstrated. In addition, this section discusses some of the criteria in more detail. The objectives chosen for this discussion are those that may not be obvious, or that are particularly important.

Overall it can be seen that the research has fulfilled the objectives that were given in section 1.3. Extensions to the work are discussed in section 9.3.

Objective	Criteria	Section	Page
1	learns a model	5	112
	highlights novel features	6	138
	highlights only novel features	8.3	183
2	similar stimuli habituate each other	5.5	121
	the amount of generalisation is tunable	5.6	127
3	operates on-line	6	138
4	very novel $>$ a bit novel	4.3	95
	a bit novel $>$ normal	4.3	95
5	forgets about rare perceptions	4.3.3	104
6	forgets about perceptions that are not seen for some time	4.3.3	104
7	detects novelty in small environments	6.3	141
	detects novelty in larger environments	6.3.3	147
	generates parsimonious and efficient representations of the inputs	6.3.3	147
8	detects novelty in light stimuli	4.2	89
	detects novelty in sonar input	6.3	141
	detects novelty in images	7	157
	detects novelty in non-robotic datasets	6.4	151
9	can select one of several filters that best models environment	8.2	178
	can recognise that the current environment is not in filter set	8.2	178
10	selects most novel feature	4.2	89
11	highlights novel features	8.2	178
	does not highlight familiar features	6.3	141
12	successful on toy dataset	6.2	139
	successful on the biomed dataset	6.4.1	151
	operates on the ball-bearing dataset	6.4.2	152
13	network learns on-line without problems	5.6	127
	network adds new nodes as required	5.5	121
14	the representation is efficient	5.5.3	126
	the representation is parsimonious	5.5.3	126
15	investigate topology preservation	5.5.2	122
	network performs well on topology measures	5.6	127

Objective 4: Quantifying Novelty

Objective 4 was concerned with quantifying novelty. The two assessment criteria that deal with this objective require that very novel inputs are found to be more novel than partly novel inputs and that familiar inputs are found to be less novel than partly novel inputs.

The quantification of the amount of novelty comes from the strength of the synapse connecting the winning unit to the output neuron in the novelty filter, which is controlled by the amount of habituation that the synapse has received. Providing that the network guarantees that the input that caused the habituation is the same as, or at least similar to, the current input, these conditions will be met by the fact that the habituation dynamics exhibit exponential decay.

The proviso that a perception is represented by the same node consistently is met by the GWR network, since otherwise a new node will be added to the map. However, for the HSOM there was no such guarantee since the way that the weights evolve could mean that the position of a node could move about in the map space as the weights evolve.

Objective 7: Scalability

In essence, objective 7 was the main spur for the development of the GWR network – that the novelty filter should scale with the size of the environment. There were a number of other benefits to the introduction of the network, however, such as the one mentioned under objective 4 above.

The requirement that the size of the network should not be preset comes from the desire to allow on-line learning, which means that the size of the dataset may well not be known in advance. In addition, the behaviour of different sized networks on the same dataset was shown to vary when a variety of HSOM networks were tested on the data from the 200 m loop of corridor described in section 4.4. This was because a network that was too small would saturate, but a network that was too large would not generalise well as the inputs would be very well separated.

The final assessment criterion listed under objective 7 is that the network should generate efficient and parsimonious representations of the inputs, so that the computational costs are kept down. This is also listed as two criteria for objective 14, where the focus is on the neural network rather than the whole novelty filter. For objective 7 the

intention is to investigate the performance of the network with respect to the inspection task with sonar inputs. This was demonstrated by the discussion of the number of nodes that the GWR network used to represent the data of the 200 m environment (section 6.3.3) compared to the sizes of the various HSOM networks used.

Objective 11: Suitability for Inspection Tasks

This objective is concerned with the inspection task that has formed the principal focus of the experimental sections. The analyses of the outputs when the HSOM and GWR networks were used for the inspection tasks using the sonar perceptions of the robot demonstrated that the features that were detected were novel and that only those features were novel. The same result was demonstrated using the images from the camera for the GWR network.

However, it could be argued that these results were entirely subjective – the researcher matched the perceptions of the robot at each place to the amount of novelty and provided explanations of the findings, but that does not mean that those features were the novel ones – predicting exactly what the robot sees and does not see with its sonar sensors is at best an inexact science. This was one of the benefits of the investigation in section 8.2 where an algorithm that picked the correct novelty filter from a battery of available trained filters was described. The fact that the correct novelty filter was selected each time demonstrates that the output of the novelty filter does correspond to the perceptions of the environment.

Objective 15: Topology Preservation

The three objectives regarding the neural network implementation of the novelty filter (13, 14 and 15) are only satisfied by the GWR network, not the HSOM. This point has been made in the discussion of the other objectives above, but is made especially clear by the criteria here. The measures that were introduced in sections 5.5.2 and 5.5.3 provide the data that is required to demonstrate that the performance of the GWR network is good and it is shown there that the network is perfectly topology-preserving and produces mappings at least as good as those of comparable growing networks.

9.2.4 What is Novel in this Thesis?

Chapter 2 showed clearly that the novelty filter produced in this research is certainly not the first novelty filter. So what contributions has this thesis made to the area? The

following list gives some of the more important ones:

- The novelty filter based on habituation was the first to do all of the following:
  - quantify novelty
  - operate on-line
  - be robust to *some* novel inputs in a training set
  - be able to operate in an unsupervised fashion
  - be able to be based on whatever neural network is suitable for the current task.
- In addition, the experiments used in this thesis have:
  - been the first to explicitly apply a novelty filter to robotic tasks
  - shown that a novelty filter can be used as the basis of an inspection agent
  - shown that the performance of the filter is comparable to that of other novelty filters in the literature
- and the Grow When Required network developed as part of the research
  - is capable of continuous learning
  - grows to match the input space
  - is perfectly topology-preserving
  - learns efficient and parsimonious representations of the inputs.
  - performs at least as well as other growing networks on a variety of datasets and better on novelty detection tasks

The next section considers some areas of potential future work leading on from the research that has been described.

9.3 Open Questions and Future Work

As shown in section 9.1, this thesis has proposed and investigated the behaviour of a novelty filter suitable for on-line use on a mobile robot. However, the research has opened up areas of future research as it has progressed. This section describes in more

detail some of these areas that are relevant to the novelty filter that has been proposed. The list is not intended to be exhaustive.

The first two suggestions propose extensions to the inputs to the filter. While these extensions are of interest, they should not affect the way that the novelty filter would be implemented or the effects of using the novelty filter. The third suggestion looks at whether or not temporal information could be usefully analysed by the novelty filter. This was partially investigated during the research, but further work could be done.

A novelty filter is a tool that could be applied to datasets to preprocess them in order to simplify the processing requirements of other tasks. The later suggestions in this section reflect this by suggesting ways in which the novelty filter could now be used in conjunction with other algorithms to provide further useful behaviours for robots and other artificial learning systems.

Another area where further investigation would be beneficial is a more detailed analysis of the properties of the GWR network and related self-organising algorithms. As was detailed in chapter 5, this type of learning has been under investigation for many years already, but there remains a lot of work to be done.

9.3.1 Better Vision Processing

The methods of processing images that were performed in chapter 7 were simple, but effective. Image processing has been an area of research for many years and it would be interesting to see whether more effective methods of image processing would produce significant benefits.

One thing that would certainly make a difference would be using image segmentation (Ruiz-del-Solar and Köppen, 1995; Li, 1999) to process the image, extracting features of interest. These features could then be evaluated individually, although there would have to be a lot of work on scaling and colouration issues (Gonzalez and Wintz, 1987). This would enable a form of situational novelty to be investigated – the chair is not novel, but the chair in the corridor is novel, the third type of novelty that was mentioned in section 1.1.

One other extension, which was mentioned in section 7.2 is the use of an omnidirectional camera using a parabolic lens. This would mean that each image captured information from all around the robot, as the sonar sensors do, which would mean that the preprocessing would have to be more complicated, but would also make the camera images more useful.

Using the novelty filter in conjunction with a pan and tilt camera would provide

one possible way to choose how to redirect the camera, turning towards the most novel feature in range. This can be compared to the visual 'magnet' that was described in section 7.2.1. Again, this would require image segmentation so that the most novel part of the image could be selected.

9.3.2 Sensor Fusion

Two very different robot sensors were used as inputs to the novelty filter in the inspection task, the sonar sensors – which provide approximate distances to the nearest object in the path of the sonar pulse – and the considerably more complex output of a monochrome CCD camera. These were the two types of sensor that were available on the robot (as well as the infra-red sensors that were used to control the wall-follower).

In the experiments reported in this thesis the two different sensor modalities were used separately, and the fusion of the two was not considered. The principal reason for this is that it is not clear why such a functionality would be useful for the inspection task, since the main effect would be reducing the generality of the learning – with sensor fusion, seeing a doorway will be novel if the doorway is near the corner of a corridor even if many doorways in the middle of corridors have been seen previously. This is the trade-off that is involved in sensor fusion, because more information is provided, more training is also needed. That said, there may very well be a number of applications where sensor fusion is very useful for novelty detection, and it is a popular research topic in mobile robotics (van Dam et al., 1996).

9.3.3 Temporal Processing

Although the perceptions of the robot are captured sequentially (due to the nature of the task), apart from in the neotaxis experiments there has been no attempt to look at how the temporal information could be used. The main reason for this is that the sequence of perceptions door-wall-door should be neither more nor less novel than door-wall-wall.

However, linking perceptions in this way can also be useful for certain applications. For example, one early indication of a machine fault could be that a high reading on one input is not immediately followed by some remedial action. Being able to detect such things should not be difficult to add to the system and a number of unsupervised learning networks suitable for the purpose were described in section 3.4.4.

9.3.4 Novelty-Based Exploration

In section 8.2 a way of choosing one novelty filter from a battery of trained filters was described. Included in this approach was a method of realising that the environment that the robot was currently exploring was not one that it had a trained filter for. In these circumstances it was suggested that the robot could train a new filter in this environment.

This would not be particularly difficult to implement and would then allow the robot to explore a series of environments choosing to head towards the most novel perceptions. The robot could start with no knowledge of any environments and simultaneously explore and build maps of the environments that it explored, starting new novelty filters whenever the current filter did not represent the perceptions of the robot sufficiently well. In this way environments that the robot travelled through could be automatically categorised.

9.3.5 Using Novelty Detection for Preprocessing and Attention

As was mentioned in section 1.2, one application of the novelty filter is as a preprocessing filter in order to reduce the amount of processing required by other elements of a system. This type of application was demonstrated in section 4.2 for the light sensors on the Fischertechnik robot. Further work in this area would suggest how useful such a method of directing attention towards novel stimuli actually is. This work could look at both the psychological data and the implementation of the novelty filter, together with suitable preprocessing techniques, on a mobile robot.

9.3.6 Inspection Tasks in Real World Domains

The principal application that has been used to investigate the behaviour of the novelty filters designed in this research has been the robotic inspection task. While it could be argued that using corridor environments for this task is at best artificial, the experiments have demonstrated the concept for this type of environment and there is no reason why this could not be extended.

One area where robotic inspection would be particular useful would be in sewer inspection and other types of pipeline detection, e.g., oil pipelines. This focus of the work was highlighted in an article in the New Scientist, 17th June 2000 (Mullins,

2000). A number of robots designed for this task have been developed (Kirchner and Hertzberg, 1997), as have sensors for the task (Bin Hussen, 2000), as was discussed in section 1.2. Applying the novelty detection algorithms produced in this thesis to the real task of sewer inspection or some other related application would be a question of testing the sensors and devising suitable preprocessing techniques.

It would not be possible to take the exact algorithms that were derived in this work and apply them to robotic inspection tasks without any further research. In particular, the sensors that would be used to perceive the environments would require investigation, as would the preprocessing that would need to be performed. Furthermore, for sewer inspection in particular, the features that should be detected could be tree roots and cracks, which may well be very small in comparison to the whole scene. This means that the sensory perceptions must be high resolution, and yet there is a trade-off between high resolution perceptions and useful generalisation. This would need to be investigated. Additionally, the wall-following behaviour that was used in the experiments would obviously need to be improved before the system could be used to examine real environments properly. One possible way to do this would be by using the novelty-attention link to direct the motion of the robot towards the more novel objects.

That said, the actual novelty detection algorithms should be perfectly suitable for these tasks without any modifications. Once the sensory input and preprocessing methods were chosen, some experimentation would be required to find suitable values for the parameters, especially the insertion threshold, and training environments devised that demonstrate all (or at least most) of the normal features of the target environment. The robot can then be allowed to explore these to acquire the model of normality and then explore the wider environment. If the robot were required to explore areas and then report back on the position of problems, then obviously a reliable navigation system would also be required.

9.4 Discussion and Conclusions

The primary aim of this work has been to develop, and investigate the behaviour of, an adaptive novelty filter that was capable of operating on-line, so that inputs were received, evaluated as to their novelty and then, if required, assimilated into the model. An investigation of the different types of novelty and a number of thoughts about what behaviour constitutes a novelty filter suggested that the filter would learn to recognise

and ignore inputs that were seen frequently, thereby highlighting novel inputs. One way that animals do this is habituation. This very simple example of plasticity in the brain was therefore used as the basis for the novelty filter proposed in this research.

Using habituation, together with an unsupervised self-organising neural network, as a novelty filter enables inputs to be recognised by the filter and therefore the novelty of the perception with respect to the model acquired so far to be evaluated. As the strength of each synapse used varies continuously between completely novel and familiar, so the novelty of a perception can be quantified, so that even in the presence of several novel stimuli, the most novel one can be selected.

Habituation also means that if a stimulus is not seen for some time then the relevant synapse can dishabituate, meaning that the next time that the stimulus is seen it will be found to be novel, or at least more novel than previously. This enables the novelty filter to deal with novel perceptions that are seen occasionally in the training set – a node of the unsupervised network will be trained to recognise that perception, but the novelty filter will continue to find it novel providing that it is seen only occasionally. This affords the novelty filter some robustness for training sets that it is not possible to control perfectly.

The unsupervised network that the novelty filter is based on is a very important part of the novelty filter. It is the network that recognises perceptions and identifies whether or not similar perceptions have been seen. By using a network that has neighbourhood connections between nodes that represent similar perceptions and provides a feature mapping between input nodes and the map representation, the positioning of nodes in the map field can be exploited, so that neighbouring nodes mutually habituate one another and so perceptions similar to those already seen will not be found to be novel.

Much of the work described in this thesis is the development of a self-organising network with neighbourhood connections that is suitable for continuous learning. The importance of this for novelty detection is clear – the network must be able to add new perceptions to the model on-line as they occur so that it does not saturate. Another vital point is that the node that is used to represent a particular input must stay the same, so that the habituation synapse represents the novelty of the correct perception consistently. The ‘Grow When Required’ (GWR) network that was developed to satisfy these criteria has also been shown to be perfectly topology-preserving. The network has applications in many other areas as well as novelty detection, as it is suitable for use wherever unsupervised, self-organising networks are the method of choice.

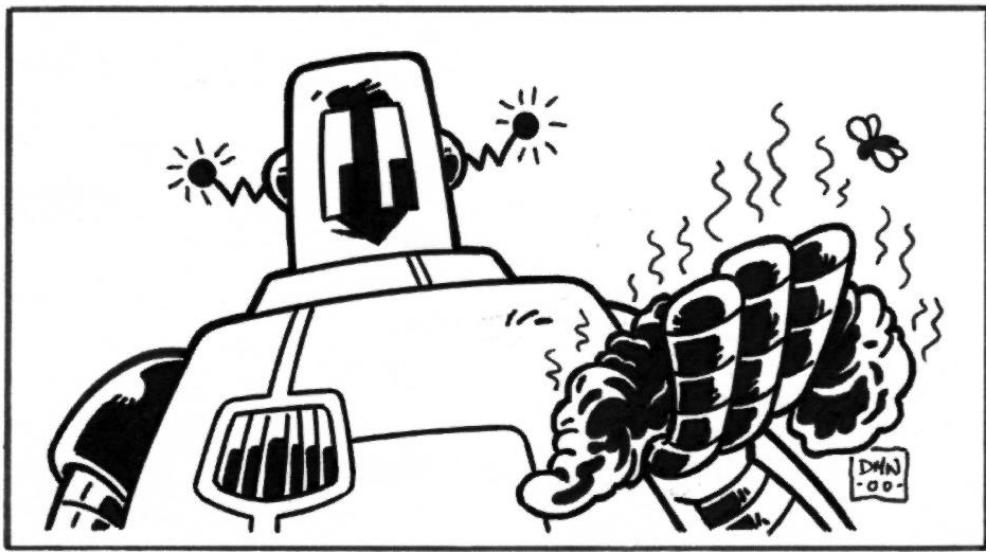
Some criticisms of the work are that the model of habituation used is simplistic, that

the experiments are not necessarily very realistic as the environments that were used were all corridors, and that it is virtually impossible to quantify the results, because the concept of what is novel is so subjective.

The first of these criticisms is not particularly valid since more complex models of habituation were investigated before Stanley's model was chosen and that all that is required is a curve that shows decay as the number of perceptions increases. By using the model of Wang and Hsu (equations 2.6 and 2.7) the faster rehabituation of dishabituated stimuli can be built in.

The first criticism of the experimental techniques was dealt with in section 9.3.6 of the future work section, while the second is unanswerable, since it is part of the nature of novelty detection. Some attempt to work around the problem was given in section 8.2, where the novelty filter was extended so that the correct model from a set of pre-trained filters was used. The novelty filter chose the correct filter 100% of the time. It was also demonstrated that the novelty filter selected only the features that were novel, and detected them reliably.

This final chapter has suggested a number of extensions to the work described in this thesis. Most of these have been applications of the novelty filter developed to tasks in both mobile robotics and other areas of machine learning. In addition, the aims and objectives introduced in chapter 1 have been evaluated with regard to the work described in the thesis. It has been shown that the novelty filter successfully operates on-line, quantifies novelty and evaluates each perception with respect to the model acquired so far before adding the new perception to the model, if required. The performance of the novelty filter has been demonstrated on a number of experiments on robots and on other real world datasets, with positive results.



My neural network is in place
My onboard sensors interface
My bits and bytes have all been counted
Upon the board my RAM is mounted
My algorithms all are tested
As an AI I can’t be bested
A robot paragon of wit
Sent underground to scour for shit.

Picture by Dave Windett, poem by Keith Marsland

Bibliography

Dirk Aeyels. On the dynamic behaviour of the novelty detector and the novelty filter. In B. Bonnard, B. Bride, J.P. Gauthier, and I. Kupka, editors, *Analysis of Controlled Dynamical Systems*, pages 1 – 10, 1990.

S. Albrecht, J. Busch, M. Kloppenburg, F. Metze, and P. Tavan. Generalised radial basis function networks for classification and novelty detection: Self-organisation of optimal Bayesian decision. *Neural Networks*, 13:1075 – 1093, 2000.

E. Ardizzone, A. Chella, S. Gaglio, D. Morreale, and S. Sorbello. The novelty filter approach to detection of motion. In E.R. Caianiello, editor, *Third Italian Workshop on Parallel Architectures and Neural Networks*, pages 301–308, 1990.

Craig Bailey and M.C. Chen. Morphological basis of long-term habituation and sensitization in *aplysia*. *Science*, 220:91–93, 1983.

Christian Balkenius. Attention, habituation and conditioning: Towards a computational model. *Cognitive Science Quarterly*, 1(2), 2000.

V. Barnett and T. Lewis. *Outliers in Statistical Data*. Wiley Series in Probability and Mathematical Statistics. John Wiley and Sons, New York, USA, 3rd edition, 1994.

H.-U. Bauer and K. Pawelzik. Quantifying the neighbourhood preservation of self-organising maps. *IEEE Transactions on Neural Networks*, 3(4):570 – 579, 1992.

H.-U. Bauer and Th. Villmann. Growing a hypercubical output space in a self-organizing feature map. *IEEE Transactions on Neural Networks*, 8(2):218–226, 1997.

Claude Berge. *Topological Spaces: Including a Treatment of Multi-Valued Functions, Vector Spaces and Convexity*. Dover, New York, 1997.

Mohamad P. Bin Hussen. Automated inspection of sewers using ultrasonic sensors. In *Proceedings of the EUREL Conference on Advanced Robotics Systems*, 2000.

Christopher M. Bishop. Novelty detection and neural network validation. *IEEE Proceedings on Vision, Image and Signal Processing*, 141(4):217–222, 1994.

Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford, England, 1995. ISBN 0-19-853864-2.

Rafal Bogacz, Malcolm W. Brown, and Christophe Giraud-Carrier. High capacity neural networks for familiarity discrimination. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN'99)*, pages 773 – 778, 1999.

Rafal Bogacz, Malcolm W. Brown, and Christophe Giraud-Carrier. Model of familiarity discrimination in the perirhinal cortex. *Journal of Computational Neuroscience*, 2000.

Rodney Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2:14 – 23, 1986.

M. W. Brown and J.-Z. Xiang. Recognition memory: Neuronal substrates of the judgement of prior occurrence. *Progress in Neurobiology*, 55:149 – 189, 1998.

M.W. Brown. Neuronal responses and recognition memory. *Seminars in the Neurosciences*, 8:23 – 32, 1996.

Jörg Bruske and Gerald Sommer. Dynamic cell structure learns perfectly topology preserving maps. *Neural Computation*, 7:845 – 865, 1995.

W. Burgard, A.B., Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun. The interactive museum tour-guide robot. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI'98)*, Madison, Wisconsin, 1998.

Christopher J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121 – 167, 1998.

Vanco Burzevski and Chilukuri K. Mohan. Hierarchical growing cell structures. In *Proceedings of the IEEE International Conference on Neural Networks (ICNN'96)*, volume 3, pages 1658 – 1663, 1996.

John H. Byrne and Kevin J. Gingrich. Mathematical model of cellular and molecular processes contributing to associative and nonassociatiave learning in *aplysia*. In John H. Byrne and William O. Berry, editors, *Neural Models of Plasticity*, chapter 4, pages 58–72. Academic Press, New York, 1989.

Colin Campbell and Kristin P. Bennett. A linear programming approach to novelty detection. In T.K. Leen, T.G. Diettrich, and V. Tresp, editors, *Proceedings of Advances in Neural Information Processing Systems 13 (NIPS'00)*, Cambridge, MA, 2000. MIT Press.

Gail A. Carpenter and Stephen Grossberg. The ART of adaptive pattern recognition by a self–organising neural network. *IEEE Computer*, 21:77 – 88, 1988.

V.F. Castellucci, T.J. Carew, and E.R. Kandel. Cellular analysis of long-term habituation of the gill-withdrawel reflex in *aplysia*. *Science*, 202:1306–1308, 1978.

Thomas P. Caudell and David S. Newman. An adaptive resonance architecture to define normality and detect novelties in time series and databases. In *IEEE World Congress on Neural Networks*, pages 166–176, 1993.

Chambers. *Chambers English Dictionary*. Chambers, Edinburgh, 7th edition, 1990.

G.J. Chappell and J.G. Taylor. The temporal Kohonen map. *Neural Networks*, 6: 441–445, 1993.

Haibo Chen, Roger D. Boyle, Howard R. Kirby, and Frank O. Montgomery. Indentifying motorway incidents by novelty detection. In *8th World Conference on Transport Research*, 1998.

Guojian Cheng and Andreas Zell. Externally growing cell structures for pattern classification. In *Proceedings of the 2nd International Symposium on Neural Computation (NC'2000)*, pages 233 – 239, 2000.

M. Cottrell, J.C. Fort, and G. Pages. Theoretic aspects of the SOM algorithm. In *Proceedings of Workshop on Self-Organising Maps (WSOM'97)*, pages 246 – 267, 1997.

L. H. Cox, M.M. Johnson, and K. Kafadar. Exposition of statistical graphics technology. In *ASA Proceedings of the Statistical Computation Section*, pages 55–56, 1982.

Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines (and other kernel-based learning methods)*. Cambridge University Press, Cambridge, 2000.

Paul Crook and Gillian Hayes. A robot implementation of a biologically inspired method for novelty detection. In *Proceedings of Towards Intelligent Mobile Robots (TIMR'01)*, 2001.

Dipanka Dasgupta and Stephanie Forrest. Novelty detection in times series data using ideas from immunology. In *Proceedings of the Fifth International Conference on Intelligent Systems*, 1996.

Wolfgang J. Daunicht. Autoassociation and novelty detection by neuromechanics. *Science*, 253:1289 – 1291, 13 September 1991.

Luc Devroye and Gary L. Wise. Detection of abnormal behaviour via nonparametric estimation of the support. *SIAM Journal of Applied Mathematics*, 38(3):480 – 488, 1980.

Patrik D'Haeseleer, Stephanie Forrest, and Paul Helman. An immunological approach to change detection: Algorithms, analysis and implications. In *IEEE Symposium on Security and Privacy*, 1996.

Tom Duckett and Ulrich Nehmzow. Mobile robot self-localisation and measurement of performance in middle scale environments. *Robotics and Autonomous Systems*, 24(1–2):57–69, 1998.

R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis*. John Wiley and Sons, New York, USA, 1973.

Hamed Elsimary. Implementation of neural network and genetic algorithms for novelty filters for fault detection. In *Proceedings of the 39th Midwest Symposium on Circuits and Systems*, pages 1432–1435, 1996.

Neil Euliano and Jose C. Principe. Spatio-temporal self-organizing feature maps. In *Proceedings of the IEEE International Conference on Neural Networks (ICNN'96)*, volume 4, pages 1900 – 1905, 1996.

Neil R. Euliano and Jose C. Principe. A spatio-temporal memory based on SOMs with activity diffusion. In *Proceedings of the Workshop on Self-Organising Maps (WSOM'99)*, pages 253 – 266, 1999.

J.-P. Ewert and W. Kehl. Configurational prey-selection by individual experience in the toad *bufo bufo*. *Journal of Comparative Physiology A*, 126:105–114, 1978.

S. E. Fahlman and C. Lebiere. The cascade-correlation learning architecture. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems 2 (NIPS'90)*, pages 524–532. Morgan Kaufmann, San Mateo, 1990.

Craig L. Fancourt and Jose C. Principe. On the use of neural networks in the generalised likelihood ratio test for detecting abrupt changes in signals. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN'00)*, volume II, pages 243 – 248, 2000.

M.O. Franz, B. Schölkopf, H.A. Mallot, and H.H. Bülthoff. Where did I take this snapshot? Scene-based homing by image matching. *Biological Cybernetics*, 79(3): 191 – 202, 1998.

Bernd Fritzke. Supervised learning with growing cell structures. In J. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems 6 (NIPS'93)*, pages 255–262. Morgan Kaufmann Publishers, San Mateo, CA, 1994.

Bernd Fritzke. A growing neural gas network learns topologies. In G. Tesauro, D. S. Touretzky, and T. K. Leen, editors, *Advances in Neural Information Processing Systems 7 (NIPS'94)*, pages 625–632, Cambridge, MA, 1995. MIT Press.

Bernd Fritzke. A self-organizing network that can follow non-stationary distributions. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN'97)*, pages 613–618. Springer, Berlin, 1997.

Rafael C. Gonzalez and Paul Wintz. *Digital Image Processing*. Addison-Wesley, Reading, Massachusetts, 2nd edition, 1987.

Geoffrey J. Goodhill and Terrence J. Sejnowski. A unifying objective function for topographic mappings. *Neural Computation*, 9:1291–1304, 1997.

S. Greenberg, V. Castellucci, H. Bayley, and J. Schwartz. A molecular mechanism for long-term sensitisation in *aplysia*. *Nature*, 329:62–65, 1987.

Stephen Grossberg. A neural theory of punishment and avoidance. I. Qualitative theory. *Mathematical Biosciences*, 15:39–67, 1972a.

Stephen Grossberg. A neural theory of punishment and avoidance. II. Quantitative theory. *Mathematical Biosciences*, 15:253–285, 1972b.

P.M. Groves and R.F. Thompson. Habituation: A dual-process theory. *Psychological Review*, 77(5):419–450, 1970.

E.J. Gumbel. *Statistics of Extremes*. Columbia University Press, New York, USA, 1958.

J. Hájek and Z. Šidák. *Theory of Rank Tests*. Academic Press, New York, USA, 1967.

Robert M. Haralick and Linda G. Shapiro. *Computer and Robot Vision*, volume 1. Addison-Wesley, Reading, Massachusetts, 1992.

Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice-Hall, New Jersey, USA, 2nd edition, 1999.

D.O. Hebb. *The Organisation of Behaviour*. Wiley, New York, 1949.

Joachim Hertzberg, Thomas Christaller, Frank Kirchner, Ulrich Licht, and Erich Rome. Sewer robotics. In *From Animals to Animats 5: Proceedings of SAB'98*, pages 427 – 436, 1998.

Simon J. Hickinbotham and James Austin. Neural networks for novelty detection in airframe strain data. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN'00)*, volume VI, pages 375 – 380, 2000.

Tuong Vinh Ho and Jean Rouat. A novelty detector using a network of integrate and fire neurons. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN'97)*, pages 103 – 108, 1997.

Tuong Vinh Ho and Jean Rouat. Novelty detection based on relaxation time of a network of integrate–and–fire neurons. In *Proceedings of the 2nd IEEE World Congress on Computational Intelligence (WCCI'98)*, pages 1524–1529, 1998.

Albert J. Höglund, Kimmo Hätönen, and Antti Sorvari. A computer host-based user anomaly detection system using the self-organising map. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN'00)*, volume V, pages 411 – 416, 2000.

J.J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. In *Proceedings of the National Academy of Sciences*, volume 79, pages 2554–2558, USA, 1982.

P.V.C. Hough. A method and means for recognising complex patterns, 1962. U.S. Patent Number 3,069,654.

Peter J. Huber. *Robust Statistics*. Wiley Series in Probability and Mathematical Statistics. John Wiley and Sons, New York, USA, 1981.

Arun Jagota. Novelty detection on a very large number of memories stored in a Hopfield-style network. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN'91)*, 1991.

Natalie Japkowicz, Catherine Myers, and Mark Gluck. A novelty detection approach to classification. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI'95)*, pages 518 – 523, 1995.

I.T. Jolliffe. *Principal Component Analysis*. Springer Series in Statistics. Springer-Verlag, Berlin, Germany, 1986.

S. Kaski, J. Kangas, and T. Kohonen. Bibliography of self-organising map (SOM) papers: 1981 – 1997. *Neural Computing Surveys*, 1:102 – 350, 1998.

Frank Kirchner and Joachim Hertzberg. A prototype study of an autonomous robot platform for sewerage system maintenance. *Autonomous Robots*, 4(4):319 – 333, 1997.

R. T. Knight. Contribution of human hippocampal region to novelty detection. *Nature*, 383:256 – 259, 1996.

Hanseok Ko, Robert Baran, and Mohammed Arozullah. Neural network based novelty filtering for signal detection enhancement. In *Proceedings of the 35th Midwest Symposium on Circuits and Systems*, pages 252–255, 1992.

Hanseok Ko and Garry M. Jacyna. Dynamical behaviour of autoassociative memory performing novelty filtering for signal enhancement. *IEEE Transactions on Neural Networks*, 11(5):1152 – 1161, 2000.

Teuvo Kohonen. Self-organised formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69, 1982.

Teuvo Kohonen. *Self-Organization and Associative Memory, 3rd ed*. Springer, Berlin, 1993.

Teuvo Kohonen and E. Oja. Fast adaptive formation of orthogonalizing filters and associative memory in recurrent networks of neuron-like elements. *Biological Cybernetics*, 25:85–95, 1976.

Andreas Kurz. Constructing maps for mobile robot navigation based on ultrasonic range data. *IEEE Transactions on Systems, Man and Cybernetics — Part B: Cybernetics*, 26(2):233–242, 1996.

A.J. Lacey, N.A. Thacker, and N.L. Seed. Smart feature detection using an invariance network architecture. In *Proceedings of the British Machine Vision Conference*, pages 327 – 336, 1995.

Dimitrios Lambrinos, Christian Scheier, and Rolf Pfeifer. Unsupervised classification of sensory-motor states in a real world artifact using a temporal Kohonen map. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN'95)*, pages 467 – 472, 1995.

Rolando Lara and M.A. Arbib. A model of the neural mechanisms responsible for pattern recognition and stimulus specific habituation in toads. *Biological Cybernetics*, 51:223–237, 1985.

Wee Kheng Leow and Risto Miikkulainen. A neural network for attentional spotlight. In *Proceedings of the International Conference on Neural Networks (ICANN'91)*, pages 436–441, 1991.

Daniel S. Levine and Paul S. Prueitt. Modelling some effects of frontal lobe damage – novelty and perseveration. *Neural Networks*, 2:103 – 116, 1989.

Daniel S. Levine and Paul S. Prueitt. Simulations of conditioned perseveration and novelty preference from frontal lobe damage. In Michael L. Commons, Stephen Grossberg, and John E.R. Staddon, editors, *Neural Network Models of Conditioning and Action*, chapter 5, pages 123 – 147. Lawrence Erlbaum Associates, Hillsdale, NJ, 1992.

Zhaoping Li. Visual segmentation by contextual influences via intra-cortical interactions in the primary visual cortex. *Network: Computational Neural Systems*, 10: 187–212, 1999.

Fredrik Linaker and Lars Niklasson. Times series segmentation using an adaptive resource allocating vector quantisation network based on change detection. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN'00)*, volume VI, pages 323 – 328, 2000.

Georges Linares, Pascal Nocera, and Henri Meloni. Model breaking detection using independent component classifier. In *Proceedings of Fifth International Conference on Artificial Neural Networks (ICANN'97)*, pages 559 – 563, 1997.

Peter Lozo. Neural circuit for match/mismatch, familiarity/novelty and synchronization detection in SAART networks. In *International Symposium on Signal Processing and its Applications*, pages 549–552, 1996.

R.R. MacDonald. On statistical testing in psychology. *British Journal of Psychology*, 88:333 – 347, 1997.

Michael C. Mackey and Leon Glass. Oscillation and chaos in physiological control system. *Science*, 197:287 – 289, 1977.

Yuval Marom and Gillian Hayes. Preliminary approaches to attention for social learning. In *ICAI Workshop on Biologically Inspired Machine Learning*, pages 8 – 18, 1999.

Yuval Marom and Gillian Hayes. Towards a model of temporal attention for on-line learning in a mobile robot. In *Proceedings of the International Conference on Anticipatory Systems (CASYS'2000)*, 2000.

Stephen Marsland, Ulrich Nehmzow, and Tom Duckett. Learning to select distinctive landmarks for mobile robot navigation. *Robotics and Autonomous Systems*, 37:241 – 260, 2001.

Thomas Martinetz. Competitive Hebbian learning rule forms perfectly topology preserving maps. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN'93)*, pages 426 – 438, 1993.

Thomas M. Martinetz, Stanislav G. Berkovich, and Klaus J. Schulten. "Neural-gas" network for vector quantization and its application to time-series prediction. *IEEE Transactions on Neural Networks*, 4(4):558– 569, 1993.

Thomas M. Martinetz and Klaus J. Schulten. A “neural-gas” network learns topologies. In T. Kohonen, K. Mäkisara, O. Simula, and J. Kangas, editors, *Artificial Neural Networks*, pages 397–402, Amsterdam, 1991. Elsevier.

Thomas M. Martinetz and Klaus J. Schulten. Topology representing networks. *Neural Networks*, 7(3):505 – 522, 1994.

Dominique Martinez. Neural tree density estimation for novelty detection. *IEEE Transactions on Neural Networks*, 9(2):330 – 338, 1998.

Kiyotoshi Matsuoka and Mitsuri Kawamoto. A self-organising neural network for principal component analysis, orthogonal projection and novelty filtering. In *World Congress on Neural Networks (WCNN’93)*, volume II, pages 501 – 504, 1993.

James L. McClelland, David E. Rumelhart, and the PDP Research Group. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 2: Psychological and Biological Models*. MIT Press, Cambridge, MA, 1986.

Merriam-Webster. *WWWebster Dictionary*. Merriam-Webster, `http://www.m-w.com/netdict.htm`, 1999.

Janet Metcalfe. Novelty monitoring, metacognition, and control in a composite holographic associative recall model: Implications for Korsakoff amnesia. *Psychological Review*, 100(1):3 – 22, 1993.

John Moody and Christian J. Darken. Fast learning in networks of locally-tuned processing units. *Neural Computation*, 1:281 – 294, 1989.

Alberto Muñoz and Jorge Muruzábal. Self-organising maps for outlier detection. *Neurocomputing*, 18:33 – 60, 1998.

Justin Mullins. Sewer robots learn to do our dirty work. *New Scientist*, 166(2243):14, 17 June 2000.

Alexandre Nairac, Timothy Corbett-Clark, Ruth Ripley, Neil Townsend, and Lionel Tarassenko. Choosing an appropriate model for novelty detection. In *Proceedings of the Fifth Conference on Artificial Neural Networks (ICANN’97)*, pages 442 – 447, 1997.

Alexandre Nairac, Neil Townsend, Roy Carr, Steve King, Peter Cowley, and Lionel Tarassenko. A system for the analysis of jet system vibration data. *Integrated Computer-Aided Engineering*, 6(1):53 – 65, 1999.

H. Öğmen, R.V. Prakash, and M. Moussa. Some neural correlates of sensorial and cognitive control of behaviour. In *Proceedings of the SPIE, Science of Neural Networks*, volume 1710, pages 177 – 188, 1992.

Haluk Öğmen and Ramkrishna Prakash. A developmental perspective to neural models of intelligence and learning. In Daniel S. Levine and Wesley R. Elsberry, editors, *Optimality in Biological and Artificial Networks?*, chapter 18, pages 363 – 395. Lawrence Erlbaum Associates, Hillsdale, NJ, 1997.

Erkki Oja. S-orthogonal projection operators as asymptotic solutions of a class of matrix differential equations. *SIAM Journal of Mathematical Analysis*, 9(5):848 – 854, October 1978.

John O'Keefe and Lynn Nadel. *The Hippocampus as a Cognitive Map*. Oxford University Press, Oxford, England, 1978.

Sageev Oore, Geoffrey E. Hinton, and Gregory Dudek. A mobile robot that learns its place. *Neural Computation*, 9:683–699, 1997.

L. Paletta, E. Rome, and A. Pinz. Visual object detection for autonomous sewer robots. In *Proceedings of the 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems (ICRA'00)*, pages 1087 – 1093, 1999.

Amanda Parker, Edward Wilding, and Colin Akerman. The von Restorff effect in visual object recognition in humans and monkeys: The role of frontal/perirhinal interaction. *Journal of Cognitive Neuroscience*, 10(6):691 – 703, 1998.

Lucas Parra, Gustavo Deco, and Stefan Miesbach. Statistical independence and novelty detection with information preserving non-linear maps. *Neural Computation*, 8(2): 260 – 269, 1996.

Harman V.S. Peeke and Michael J. Herz, editors. *Habituation*, volume 1: Behavioural Studies. Academic Press, New York, 1973a.

Harman V.S. Peeke and Michael J. Herz, editors. *Habituation*, volume 2: Physiological Substrates. Academic Press, New York, 1973b.

William Penny and Stephen Roberts. Neural network predictions with error bars. Technical Report TR-97-1, Imperial College, London, 1997.

Roger Penrose. A generalized inverse for matrices. In *Proceedings of the Cambridge Philosophical Society*, volume 52, pages 406–413, 1955.

Michael P. Perrone and Leon N. Cooper. When networks disagree: Ensemble methods for hybrid neural networks. In R.J. Mammone, editor, *Neural Networks for Speech and Image Processing*, chapter 10. Chapman–Hall, New York, USA, 1993.

Dean A. Pomerleau. Input reconstruction reliability esimation. In Stephen Josè Hanson, Jack D. Cowan, and C. Lee Giles, editors, *Advances in Neural Information Processing Systems 5 (NIPS'92)*, pages 279 – 286, 1992.

Karl Popper. *The logic of scientific discovery*. Hutchinson, London, 1959.

Ramkrishna Prakash and Haluk Öğmen. Self-organisation via active exploration: Hardware implementation of a neural robot. *Robotica*, 16:127 – 141, 1998.

Karl H. Pribram. A further experimental analysis of the behavioural deficit that follows injury to the primate frontal cortex. *Experimental Neurology*, 3:432–466, 1961.

Karl H. Pribram. Familiarity and novelty: The contributions of the limbric forebrain to valuation and the processing of relevance. In Daniel S. Levine and Samuel J. Leven, editors, *Motivation, Emotion and Goal Direction in Neural Networks*, chapter 10, pages 337 – 365. Lawrence Erlbaum Associates, Hillsdale, NJ, 1992.

Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257 – 285, 1989.

Lawrence R. Rabiner and B.H. Juang. An introduction to Hidden Markov Models. *IEEE ASSP Magazine*, pages 4–16, January 1986.

Douglas L. Reilly, Leon N. Cooper, and Charles Erlbaum. A neural model for category learning. *Biological Cybernetics*, 45:35 – 41, 1982.

Stephen Roberts. Novelty detection using extreme value statistics. *IEE Proceedings on Vision, Image and Signal Processing*, 146(3):124 – 129, 1998.

Stephen Roberts, William Penny, and David Pillot. Novelty, confidence and errors in connectionist systems. *Proceedings of the IEE Colloquium on Intelligent Systems and Fault Detection*, 261(10):1 – 10, 1996.

Stephen Roberts and Lionel Tarassenko. A probabilistic resource allocating network for novelty detection. *Neural Computation*, 6:270 – 284, 1994.

Stephen Roberts, Lionel Tarassenko, James Pardey, and David Siegwart. A validation index for artificial neural networks. In *Proceedings of the 1st International Conference and Expert Systems in Medicine and Healthcare (NNESMED'94)*, pages 23 – 30, 1994.

Stephen J. Roberts. Extreme value statistics for novelty detection in biomedical data processing. In *Proceedings of the International Conference on Advances in Medical Signal and Information Processing (MEDSIP'00)*, 2000.

F. Rosenblatt. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Spartan, Washington, DC, 1962.

P.J. Rousseeuw and A.M. Leroy. *Robust Regression and Outlier Detection*. Wiley Series in Probability and Mathematical Statistics. John Wiley and Sons, New York, USA, 1987.

Javier Ruiz-del-Solar and Mario Köppen. A neural architecture for preattentive segmentation of sewage pipes video images. In Jose Mira and Francisco Sandoval, editors, *From Natural to Articial Neural Computation, Proceedings of the International Workshop on Artificial Neural Networks*, number 930 in Lecture Notes in Computer Science, pages 875 – 881, 1995.

J. W. Sammon. A non-linear mapping for data structure analysis. *IEEE Transactions on Computers*, 18(5):401 – 409, 1969.

T.D. Sanger. Optimal unsupervised learning in a single layer feedforward network. *Neural Networks*, 2:459 – 473, 1989.

Christian Scheier and Stefan Egner. Visual attention in a mobile robot. In *Proceedings of the IEEE International Symposium on Industrial Economics*, 1997.

Bernhard Schölkopf, John C. Platt, John Shawe–Taylor, Alex J. Smola, and Robert C. Williamson. Estimating the support of a high–dimensional distribution. Technical Report MSR–TR–99–87, Microsoft Research, 1999.

Frank Sengpiel and Mark Hübener. Visual attention: Spotlight on the primary visual cortex. *Current Biology*, 9(9):R318–R321, 1999.

B.W. Silverman. *Density Estimation for Statistics and Data Analysis*. Monographs on Statistics and Applied Probability. Chapman and Hall, London, 1986.

Padhraic Smyth. Hidden Markov models for fault detection in dynamic systems. *Pattern Recognition*, 27(1):149 – 164, 1994a.

Padhraic Smyth. Markov monitoring with unknown states. *IEEE Journal on Selected Areas in Communications*, 12(9):1600 – 1612, 1994b.

Milan Sonka, Vaclav Hlavac, and Roger Boyle. *Image Processing, Analysis and Machine Vision*. Chapman and Hall Computing, London, UK, 1993.

J.E.R. Staddon. A note on rate-sensitive habituation. In *From Animals to Animats, Proceedings of the Second International Conference on Adaptive Behaviour (SAB'92)*, pages 203 – 207, 1992.

James C. Stanley. Computer simulation of a model of habituation. *Nature*, 261:146–148, 1976.

Bryan Stiles and Joydeep Ghosh. A habituation based neural network for spatio-temporal classification. In F. Girosi, J. Makhoul, El Manolakos, and E. Wilson, editors, *Proceedings of the Fifth IEEE Workshop on Neural Networks for Signal Processing*, pages 135 – 144, 1995.

Bryan Stiles and Joydeep Ghosh. A habituation based neural network for spatio-temporal classification. *Neurocomputing*, 15(3):273 – 307, 1997.

Robert J. Streifel, R.J. Marks II, M.A. El-Sharkawi, and I. Kerszenbaum. Detection of shorted-turns in the field winding of turbine-generator rotors using novelty detectors – development and field test. *IEEE Transactions on Energy Conversion*, 11(2):312 – 317, 1996.

J. Tani and S. Nolfi. Learning to perceive the world as articulated: an approach for heirarchical learning in sensory-motor systems. *Neural Networks*, 12:1131–1141, 1999.

L. Tarassenko, P. Hayton, N. Cerneaz, and M. Brady. Novelty detection for the identification of masses in mammograms. In *Proceedings of the 4th IEE International Conference on Artificial Neural Networks (ICANN'95)*, pages 442 – 447, 1995.

David Tax and R. Duin. Data domain description by support vectors. In M. Verleysen, editor, *Proceedings of European Symposium on Artificial Neural Networks (ESANN'99)*, pages 251 – 256, 1999.

David M.J. Tax and Robert P.W. Duin. Outlier detection using classifier instability. In A. Amin, D. Dori, P. Pudil, and H. Freeman, editors, *Advances in Pattern Recognition*, volume 1451 of *Lecture Notes in Computer Science*, pages 593–601, Berlin, 1998. Springer.

David M.J. Tax, Alexander Ypma, and Robert P.W. Duin. Support vector data description applied to machine vibration analysis. In M. Boasson, J.A. Kaandorp, J.F.M. Tonino, and M.G. Vosselman, editors, *Annual Conference of the Advanced School for Computing and Imaging (ASCI'99)*, pages 398 – 405, 1999.

John G. Taylor. Self-organisation in the time domain. In M.A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*, pages 843–846. MIT Press, Cambridge, MA, 1995.

Odin Taylor and John MacIntyre. Adaptive local fusion systems for novelty detection and diagnostics in condition monitoring. In *SPIE International Symposium on Aerospace/Defense Sensing*, 1998.

Neil A. Thacker and John E.W. Mayhew. Designing a layered network for context sensitive pattern classification. *Neural Networks*, 3(3):291 – 299, 1990.

R.F. Thompson. The neurobiology of learning and memory. *Science*, 233:941–947, 1986.

R.F. Thompson and W.A. Spencer. Habituation: A model phenomenon for the study of neuronal substrates of behaviour. *Psychological Review*, 73(1):16–43, 1966.

Thomas J. Tighe and Robert N. Leaton, editors. *Habituation: Perspectives from Child Development, Animal Behaviour, and Neurophysiology*. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1976.

Hans G.C. Tråvén. A neural network approach to statistical pattern classification by "semiparametric" estimation of probability density functions. *IEEE Transactions on Neural Networks*, 2(3):366 – 377, 1991.

M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3:71 – 86, 1991.

J. W. M. van Dam, B. J. A. Kröse, and F. C. A. Groen. Neural network applications in sensor fusion for an autonomous mobile robot. In L. Dorst, M. van Lambalgen, and F. Voorbraak, editors, *Reasoning with Uncertainty in Robotics*, pages 263–277. Springer, 1996.

Vladimir Vapnik. *The Nature of Statistical Learning*. Springer-Verlag, Berlin, 1995.

Th. Villmann and H.-U. Bauer. Applications of the growing self-organising map. *Neurocomputing*, 21:91 – 100, 1998.

Thomas Villmann, Ralf Der, Michael Herrmann, and Thomas M. Martinetz. Topology preservation in self-organising feature maps: Exact definition and measurement. *IEEE Transactions on Neural Networks*, 8(2):256 – 266, March 1997.

Nikos A. Vlassis, Apostolos Dimopoulos, and George Papakonstantinou. The probabilistic growing cell structures algorithm. In *Proceedings of the 7th International Conference on Artificial Neural Networks (ICANN'97)*, pages 645 – 654, 1997.

Hedwig von Restorff. Analyse von vorgangen in spurenfeld (an analysis of the processes in the trace field). *Psychologice Forschung*, 18:299 – 342, 1933.

Georg von Wichert. Self–organising visual perception for mobile robot navigation. In *Proceedings of Eurobot '96, the 1st European Workshop on Advanced Mobile Robots*, pages 194 – 200. IEEE, 1996.

DeLiang Wang and Michael A. Arbib. Hierarchical dishabituation of visual discrimination in toads. In *From Animals to Animats 1: Proceedings of First International Conference on the Simulation of Adaptive Behaviour (SAB'91)*, pages 77–88, 1991a.

DeLiang Wang and Michael A. Arbib. How does the toad's visual system discriminate different worm-like stimuli? *Biological Cybernetics*, 64:251–261, 1991b.

DeLiang Wang and Michael A. Arbib. Modelling the dishabituation hierarchy: The role of the primordial hippocampus. *Biological Cybernetics*, 76:535–544, 1992.

DeLiang Wang and Jorg-Peter Ewert. Configuration pattern discrimination responsible for dishabituation in common toads *bufo bufo (l.)*: Behavioural tests of the

predictions of a neural model. *Journal of Comparative Physiology A*, 170:317–325, 1992.

DeLiang Wang and Chochun Hsu. SLONN: A simulation language for modelling of neural networks. *Simulation*, 55:69–83, 1990.

Andreas S. Weigend and David A. Nix. Predictions with confidence intervals (local error bars). In *Proceedings of the International Conference on Neural Information Processing (ICONIP'94)*, pages 847 – 852, 1994.

K. Worden. Structural fault detection using a novelty measure. *Journal of Sound and Vibration*, 201(1):85 – 101, 1997.

K. Worden, S.G. Pierce, G. Manson, W.R. Philp, W.J. Staszewski, and B. Culshaw. Detection of defects in composite plates using lamp waves and novelty detection. *International Journal of Systems Science*, 31(11):1397 – 1409, 2000.

Brian Yamauchi and Randall Beer. Spatial learning for navigation in dynamic environments. *IEEE Transactions on Systems, Man and Cybernetics Section B*, 26(3): 496 – 505, 1996.

Alexander Ypma and Robert P. Duin. Novelty detection using self-organizing maps. In *Proceedings of International Conference on Neural Information Processing and Intelligent Information Systems (ICONIP'97)*, pages 1322 – 1325, 1997.

David Zeaman. The ubiquity of novelty – familiarity (habituation?) effects. In Thomas J. Tighe and Robert N. Leaton, editors, *Habituation: Perspectives from Child Development, Animal Behaviour, and Neurophysiology*, chapter 9. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1976.

Appendix A

Transfer Function Derivation

This appendix supplements the discussion in section 2.4 by providing further detail for the derivations in equations 2.21 and 2.22.

Equation 2.21 can be derived in the following way:

$$\begin{aligned}
\Phi^{-1}\tilde{\mathbf{x}} &= \mathbf{x} && \text{(A.1)}\\
\Rightarrow \Phi^{-1}(I-M)^{-1}\mathbf{x} &= \mathbf{x} && \text{(A.2)}\\
\Rightarrow \frac{d\Phi^{-1}}{dt}(I-M)^{-1}\mathbf{x}+\Phi^{-1}\frac{d}{dt}(I-M)^{-1}\mathbf{x} &= 0 && \text{(A.3)}\\
\Rightarrow \frac{d\Phi^{-1}}{dt}\Phi\mathbf{x}+\Phi^{-1}\frac{d}{dt}(\Phi\mathbf{x}) &= 0 && \text{(A.4)}\\
\Rightarrow \frac{d\Phi^{-1}}{dt} &= -\Phi^{-1}\frac{d\Phi}{dt}\mathbf{x}\mathbf{x}^{-1}\Phi^{-1} && \text{(A.5)}\\
&= -(I-M)\frac{d}{dt}(I-M)^{-1}(I-M)\\
&= -\frac{dM}{dt}. && \text{(A.6)}
\end{aligned}$$

Equation 2.22 comes from the following:

$$\begin{aligned}
\frac{d\Phi}{dt} &= \Phi\frac{dM}{dt}\Phi && \text{(A.7)}\\
&= -\alpha\Phi\tilde{\mathbf{x}}\tilde{\mathbf{x}}^{T}\Phi && \text{(A.8)}\\
&= -\alpha\Phi^{2}\mathbf{x}(\Phi\mathbf{x})^{T}\Phi && \text{(A.9)}\\
&= -\alpha\Phi^{2}\mathbf{x}\mathbf{x}^{T}\Phi^{2}, && \text{(A.10)}
\end{aligned}$$

as Φ is symmetric.

Index