Overview of Speech Recognition and Related Machine Learning Techniques[1]

Mengjie Zhang

School of Mathematical and Computing Sciences
Victoria University of Wellington
P.O. Box 600, Wellington, NZ
mengjie@mcs.vuw.ac.nz

Abstract

To provide some background knowledge in speech recognition and machine learning for the NERF-87 group, this report first reviewed the basic concepts, dimensions and commonly used architectures and techniques in speech recognition. After giving a series of concepts, data sets and major paradigms in machine learning, we reviewed two major learning approaches, neural networks and genetic algorithms. In addition, some related work such as how to build a simple speech recognition system, existing HMM software packages and speaker recognition basics are described in the appendices.

Keywords: Voice recognition, pattern recognition, learning, neural networks, genetic algorithms.

1 Introduction

This is the second report for the NERF project (application number 87) — *Novel interactive software tools for teaching English as a second language*. The goal of this report is to provide some background knowledge in speech recognition and machine learning. It is important to note that this is only an overview for speech recognition and machine learning, which include dimensions, aspects, system structures, main techniques and performance measure in speech recognition, and basic concepts, data sets, generalisation ability, learning strategies and major learning paradigms in machine learning. The details of some particular algorithms or methods such as *Hidden Markov Model (HMM)*, *neural network learning* are not presented in this report.

2 Overview of Speech Recognition

Speech is the most natural mode of communication between people. People learn the relevant skills during early childhood, without instruction, and we continue to rely on speech communication through our lives. It comes so naturally to us that we do not realise how complex phenomenon speech is. The human vocal tract and articulators are biological organs with nonlinear properties, whose operation is not just under conscious control but also

[1] NERF-87-TR-02, Janurary 2002.

affected by factors ranging from gender to upbringing to emotional state. As a result, vocalisations can vary widely regarding accent, pronunciation, articulation, roughness, nasality, pitch, volume, and speech; moreover, during transmission, our irregular speech patterns can be further distorted by background noise and echos, as well as electrical characters, if telephones or other electrical equipment are used. All these sources of variability make speech recognition a very complex problem [1, 24].

Because people are so comfortable with speech there is a large desire for being able to interact with machines by speech communication. The ability to automatically transcribe speech signals is interesting in a wide number of contexts, e.g. computer interaction (dictation, command), telephone home-banking, hands-free operation, information retrieval from databases and automatic language translation. The wide range of potential applications has motivated research in *automatic speech recognition* since the 1950s. Great progress has been made so far, especially since the 1970s, using a series of engineered approaches such as template matching, knowledge engineering and statistical modeling. However, computers are still nowhere near the level of human performance at speech recognition, and it appears that further significant advances will require some new insights [1, 24, 30].

2.1 Aspects/Dimensions of Speech Recognition

The complexity of a speech recognition system can vary along the following dimensions [24, 32]:

- **Automatic Speech Recognition and Understanding.** *Automatic speech recognition* is the process by which a computer maps an acoustic speech signal to text. This is quite different from the term *automatic speech understanding*, which refers to the process by which a computer maps an acoustic speech signal to some form of abstract meaning of the speech. In general, the latter is more difficult than the former.

- **Speaker Dependent, Adaptive and Independent System.** *A speaker dependent system* is developed to operate for a single speaker. These systems are usually easier to develop, cheaper to buy and more accurate, but not as flexible as speaker adaptive or speaker independent systems.

  A speaker independent system is developed to operate for any speaker of a particular type (e.g. New Zealander English). These systems are the most difficult to develop, most expensive and accuracy is lower than speaker dependent systems. However, they are more flexible.

  Intermediate to speaker dependent and speaker independent systems are *multi-speaker* systems aimed at a small but fixed group of people. For both multi-speaker and speaker independent recognisers, a large improvement in accuracy can be obtained by *adapting* the recogniser to a specific speaker during operation. Such systems are known as *speaker adaptive systems*, which are developed to adapt its operation to the characteristics of new speakers. Its difficulty lies somewhere between speaker independent and speaker dependent systems. The adaptation can be done in a supervised fashion from a set of enrolment utterances or in an unsupervised fashion by adapting to the user as he/she speakers.

- **Vocabulary Size.** The size of vocabulary of a speech recognition system affects the complexity, processing requirements and the accuracy of the system. Some applications only require a few words (e.g. numbers only), others require very large dictionaries (e.g. dictation machines). There are no established accurate definitions. However, the following idea is generally accepted [32]:

  - small vocabulary - tens of words
  - medium vocabulary - hundreds of words
  - large vocabulary - thousands of words
  - very-large vocabulary - tens of thousands of words.

  As a general rule, it is easy to discriminate among a small set of words, but error rates naturally increase as the vocabulary size grows. For example, the 10 digits "zero" to "nice" can be recognised essentially perfectly, but vocabulary size of 200, 5000, or 100000 may have error rates of 3%, 7%, or 45% [30]. In addition, **even a small vocabulary can be hard to recognise if it contains confusable words**. For example, the 26 letters of English alphabet (treated as 26 "words") are very difficult to discriminate because they contain so many confusable words — the E-set: `B, C, D, E, G, P, T, V, Z` and an 8% error rate is considered good [30].

- **Isolated vs Continuous Speech Recognition.** In *isolated-word recognition* the words must be uttered in isolation and each word is treated independently. *Connected word* recognition requires the words in a sentence to be uttered in isolation separated by artificial periods of silence. However, contrary to isolated word recognition it is here the sequence of words that is of interest and not just the single words in the sentence. Connected speech can be viewed as an idealisation of *continuous speech* in which sentences are uttered in a natural manner without artificial pause between words.

  Isolated and connected speech recognition is relatively easy because word boundaries are detectable and the words tend to be cleanly pronounced. Continuous speech is more difficult to handle because of a variety of effects as follows [24, 30]:

  - First, it is difficult to find the start and end points of words.
  - Another problem is "coarticulation". The production of each phoneme is affected by the production of surrounding phonemes, and similarly the the start and end of words are affected by the preceding and following words.
  - The recognition of continuous speech is also affected by the rate of speech (fast speech tends to be harder).

- **Read and Spontaneous Speech.** Systems can be evaluated on speech that is either read from prepared scripts, or speech that is uttered spontaneously. Until recently, most research in speech recognition has focused on *read* speech [30]. Spontaneous speech is vastly more difficult, since it tends to be peppered with disfluencies like "uh" and "um", false starts, incomplete sentences, stuttering, coughing, and laughter;

Moreover, the vocabulary is essentially unlimited, so the system must be able to deal intelligently with unknown words — this needs to detect and flag their presence, add them to the database, which requires some interaction with the user.

- **Task and Language Constraints.** The recognition performance also varies with the nature of constraints on the word sequences that are allowed during recognition. Some constraints may be task dependent (e.g. an airline querying application may dismiss the hypothesis "the apple is red"); other constraints may be semantic (rejecting "the apple is angry"), or syntactic (rejecting "Red is apply the"). Constraints are often represented by a grammar, which ideally filters out unreasonable sentences so that the speech recogniser evaluates only plausible sentences. Grammar are usually rated by their *perplexity*, a number that indicates the grammar's average branching factor (i.e. the number of words that can follow any given word). The difficulty of a task is more reliably measured by its perplexity than by its vocabulary size.

- **Adverse Conditions.** A systems's performance can also be degraded by a range of adverse conditions. These include environmental noise (e.g. noise in a car or a factory); acoustical distortions (e.g. echoes, room acoustics); different microphones or telephones (e.g. close-speaking, omnidirectional); limited frequency bandwidth (in telephone transmission); and altered speaking manner (shouting, whining, speaking quickly, etc.).

2.2 Speech Recognition Process, Structure and Techniques

2.2.1 General Process

A wide variety of techniques are used to perform speech recognition. There are many types of speech recognition. There also are many levels of speech recognition/analysis/understanding.

Typically speech recognition starts with the digital sampling of speech. The next stage is acoustic signal processing. Most techniques include spectral analysis; e.g. Linear Predictive Coding (LPC) analysis, Mel Frequency Cepstral Coefficients (MFCC), cochlea modelling and many more.

The next stage is recognition of phonemes, groups of phonemes and words. This stage can be achieved by many processes such as Dynamic Time Warping (DTW), hidden Markov modelling (HMM), Neural Networks (NNs), expert systems and combinations of techniques. HMM-based systems are currently the most commonly used and most successful approach.

Most systems utilise some knowledge of the language to aid the recognition process.

Some systems try to "understand" speech. That is, they try to convert the words into a representation of what the speaker intended to mean or achieve by what they said.

2.2.2 A Typical Structure

Speech recognition is basically a pattern recognition task (see section 3 for details). However, unlike conventional "static" pattern classification, speech recognition aims at assigning a sequence of class labels (words) to an observed acoustic signal. If the duration of words were fixed *a priori*, speech recognition could be done in a similar way to "static" pattern

classification by assigning class labels independently to each fixed length speech segment. However, the time-boundaries between words are not known a priori during recognition and it is necessary somehow to align hypothesized word sequences to acoustic signal, that is, to search for those segments of the speech signal that in some sense optimally represent the hypothesized words. This procedure is commonly referred to as *time-alignment and pattern matching*.

A typical speech recognition system structure is shown in Figure 1.

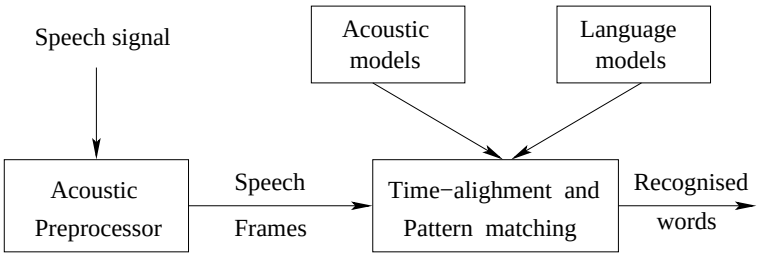

Figure 1: Structure of a speech recognition system

The Preprocessor. The preprocessor transforms the raw acoustic waveform into an intermediate compressed representation that is used for subsequent processing. Typically, the preprocessor can compress the speech data by a factor of 10 by extracting a set of *feature vectors* from the speech signal that preserves information about the uttered message. Although separating from the time alignment and pattern matching module in figure 1, the preprocessor is in principle an integrated part of the overall classifier/recogniser.

In speech recognition the speech signal is assumed piecewise stationary and the preprocessor typically produces a feature vector every 10-20ms calculated from a 20-30ms window of speech. The result of preprocessing is thus a sequence of *speech frames* or feature vectors at 10ms interval with 10-30ms coefficients per frame [24]. The feature vectors are often augmented by their first and sometimes second order derivatives calculated from linear regression on a number of consecutive speech frames. The *delta-features* provide explicit information about speech dynamics. Commonly used techniques for preprocessing are filter-bank analysis, linear prediction analysis, perceptual linear prediction and ceptral analysis.

Time Alignment and Pattern Matching. The time alignment and pattern matching process uses information from both the *acoustic model* and the *language model* to assign a sequence of words to the sequence of speech frames. The acoustic model converts the speech frames into symbolic message units of a language like such as words, syllables or phonemes that can be concatenated under the constraints imposed by the language model to form meaningful sentences.

2.2.3 Overview of the Main Techniques

Depending on the actual form of the acoustic model there are different ways of doing temporal alignment. Past approaches to speech recognition have fallen into four main categories [24, 30]:

- **Template based approaches,** in which unknown speech is compared against a set of prerecorded words (templates), in order to find the best match. This has the advantage of using perfectly accurate word models; but it also has the disadvantage that the prerecorded templates are fixed, so variations in speech can only be modeled by using many templates per word, which eventually becomes impractical.

  Dynamic time warping is such a typical approach. In this approach, the templates usually consists of a representative sequence of feature vectors for corresponding words. The basic idea here is to align the utterance to each of the template words and then select the word or word sequence that contains the "best" alignment. For each utterance, the distance between the template and the observed feature vectors are computed using some distance measure and these local distances are accumulated along each possible alignment path. The lowest scoring path then identifies the optimal alignment for a word and the word template obtaining the lowest overall score depicts the recognised word or word sequence.

  Although dynamic time warping is an elegant solution to the time alignment and patter matching problem, there are three limitations of this methods. Firstly, word templates cannot model acoustic variability between speakers very well, except in a coarse manner by using multiple templates for each word in the training set. Secondly, only templates representing whole words can be used because, in practice, it is almost impossible to record speech segments shorter than a word. In other words, it is not possible to utter phonemes in isolation. Thirdly, there is no automatic way of generating representative templates. The simplest way is to use all the training utterances as templates but this is likely to result in very poor generalisation to unseen data.

- **Knowledge based approaches,** in which "expert" knowledge about variations in speech is hand-coded into a system. This has the advantage of explicit modeling variations in speech; but unfortunately such expert knowledge is difficult to obtain and use successfully. Thus this approach was judged to be impractical, and automatic learning procedure were sought instead.

- **Statistical based approaches,** in which variations in speech are modeled statistically, using automatic, statistical learning procedure, typically the *Hidden Markov Models, or HMMs.* The approach represent the current state of the art. The main disadvantage of statistical models is that they must take a priori modeling assumptions, which are liable to be inaccurate, handicapping the system performance.

  The HHM removes the need for creating reference templates by using a probabilistic acoustic model. Instead of templates for each word (or sub-word), the HMM defines probability distributions for the assumed stationary speech segments within each word class. The advantage of using a probabilistic representation is that the data distributions within each class can be learned automatically from a set of training utterances. In addition, the probabilistic representation is far better capable of representing variations between speakers.

  The HMM is characterised by a set of states connected by transition probabilities. Each state also has assigned a probability density function describing the distribution

of speech frames or *observations*. These state-local distributions are commonly denoted emission or observation distributions. Because of the "forward" nature of speech, the HMM states are usually in a left-to-right manner.

Time alignment and patter matching with an HMM acoustic model is done in much the same way as for the template based approach. However, instead of the local distances between reference and observed feature vectors, it is now the emission probabilities for the input speech frames in combination with the transition probabilities that are used in the search for the optimal alignment. The dynamic programming match is for HMMs called *Viterbi decoding* and can be viewed as a stochastic version of dynamic time warping because the stored reference "template" are stochastic quantities. For an HMM acoustic model the segmentation corresponding to the optimal path identifies an association between HMM states and speech frames — a so-called *state segmentation*.

Ideally, the probabilistic parameters should be estimated such that the overall error rate is minimised. This can be obtained by maximising the *a posteriori* probability of observed word sequence given the observed acoustic signal. However, in many speech recognition systems the HMM for a given word is only trained to maximise the probability of the data corresponding to this word. This method is known as *Maximum Likelyhood (ML)* training and will in practice often lead to HMMs that give large probabilities for other words than the ones that were trained to model. In general, ML estimation will only be optimal if the HMMs are correct models of speech, that is, capable of representing the true within-class data distributions — this is not possible or too difficult in speech recognition.

- **Learning based approaches.** To overcome the disadvantages of the HHMs, machine learning methods could be applied, such as neural networks and genetic algorithm/programming. In these machine learning methods, explicitly rules (or other domain expert knowledge) do not need to be given a priori — they can be learned automatically through emulation or evolutionary process. More details of machine learning can be seen in section 3.

2.3 Performance Evaluation

Performance of speech recognition systems is typically measured by the *word error rate*, E, defined as [32]

$$E = \frac{S + I + D}{N} \times 100\% \tag{1}$$

where N is the total number of words in the test set, and S, I, and D are the total number of substitutions, insertions, and deletions, respectively.

3 Overview of Machine Learning and Pattern Recognition

In this section, we first review some basic concepts and definitions in machine learning and pattern recognition, then summarise the basic issues in neural networks — a major learning paradigm used in speech recognition. Finally, a very brief overview of genetic algorithms, which has also been applied to speech recognition, is presented.

3.1 Basic Concepts

3.1.1 Pattern Recognition

Pattern recognition is the research area that studies the operation and design of systems that recognize patterns in data [25]. It encloses subdisciplines like discriminant analysis, feature extraction, error estimation, cluster analysis (together sometimes called statistical pattern recognition), grammatical inference and parsing (sometimes called syntactical pattern recognition). Important application areas are image analysis, character recognition, speech analysis, human, animal and machine diagnostics, person identification and industrial inspection.

Pattern recognition is also known as *pattern classification*. The related tasks are often processed by statistical methods and machine learning methods. In the rest of this section, we will present an overview of machine learning.

3.1.2 Machine Learning Definitions

The ability to learn is a fundamental trait of intelligence. In general, the main goal of machine learning is to learn or discover some kind of knowledge or features from a data set, and to use them on unseen data set. But what is the definition of machine learning? A precise definition of learning is difficult to formulate. We give some commonly accepted descriptions or definitions of machine learning as follows.

Friedberg [5, page 2] and [6, page 285] framed the central issue of machine learning as follows:

> If we are ever to make a machine that will speak, understand or translate human languages, ... we must reduce these activities to a science so exact that we can tell the machine precisely how to go about doing them or we must develop a machine that can do things without being told precisely how ... . We could teach this machine to perform a task even though we could not describe a precise method for performing it, provided only that we understood the task well enough to be able to ascertain whether or not it had been done successfully. ... In short, although it might learn to perform a task without being told precisely how to perform it, it would still have to be told precisely how to learn.

Michalski, Carbonell and Mitchell [19, pages 1, 28] described the machine learning as follows:

> Learning is a many-faceted phenomenon. Learning processes include the acquisition of new declarative knowledge, the development of motor and cognitive skills through instruction or practice, the organisation of new knowledge into general, effective representations, and the discovery of new facts and theories through observation and experimentation. ... The study and computer modelling of learning processes in their multiple manifestations constitutes the subject matter of machine learning. ... Learning denotes changes in the system that are adaptive in the sense that they enable the system to do the same task or tasks drawn from the same population more efficiently and more effectively the next time.

Mitchell [21, pages xv, 1] further stated that

> Machine learning is the study of computer algorithms that improve automatically through experience.

3.1.3 Data Sets

A collection of input patterns or instances from which the rules or features are induced is usually called a training data set, or a *training set* for short.

A collection of input patterns or instances which were never used or *unseen* when the rules or features were learnt is known as a test data set, or a *test set* for short.

In the usual case, a training set is used for finding or learning rules/features, while a test set is for measuring the performance of these learnt rules or features.

3.1.4 Generalisation

One of the major advantages of a learning system is the ability to learn/extract the useful features from the training data set and to apply these features to the test data. The generalisation ability depends on how well the learning system has modelled the relationships in the training set. If the training set contains all the possible relationships between all the cases, then the learned program, once trained, should give good performance on the test data. There are two important issues here: *overtraining* or *overfitting* and the *training set size*.

3.1.5 Three Learning Strategies

There are three main learning strategies: *supervised, unsupervised*, and *hybrid* [15].

In supervised learning, or learning with a teacher, the learning system is provided with a correct answer (desired output) for each input pattern. The learning process is continued until the system produces answers as close as possible to the known desired correct answers.

Reinforcement learning is a variant of supervised learning in which the learning system is provided with only a critique or a clue about the correct answers, rather than the desired outputs themselves.

Unsupervised learning, or learning without a teacher, does not require a correct answer associated with each input pattern in the training data set. It usually explores the underlying structure in the data, or correlations between patterns in data, and organises patterns into categories from these correlations.

Hybrid learning combines both supervised and unsupervised learning. Part of the solutions (network weights, architecture, or computer programs) are determined through supervised learning, while the others are obtained through unsupervised learning.

3.1.6 Major Learning Paradigms

There is no commonly agreed taxonomy of machine learning paradigms. It is, however, possible to summarise the major paradigms:

- **The Connectionist Paradigm**. Connectionist learning systems are also called *artificial neural networks* (ANNs or NNs) or *parallel distributed processing systems* (PDPs) [28]. Neural networks are algorithms for cognitive tasks, such as learning and optimisation [22], which are based on concepts derived from research of the human brain. A network is generally constructed from nodes, links, weights, biases, and transfer function. The network weights are automatically updated through network training by the learning algorithm, which can be selected or developed based on the architecture of the network.

- **The Genetic Paradigm**. Genetic learning systems originally refer to *genetic algorithms* (GAs) [9], which are search algorithms based on the mechanism of natural selection and natural genetics. They usually use *bit strings*, which are generally called *chromosomes*, to represent candidate solutions. During the evolutionary process, the genetic operators, *selection, crossover* and *mutation* are applied in order to generate fitter solutions.

  Since the 1990s, *genetic programming* (GP) [16] has become another important genetic learning paradigm. Unlike genetic algorithms which use the bit strings as inputs and outputs, genetic programming uses a terminal set and a function set as inputs and evolved programs as outputs of the learning system.

- **The Case Based Learning Paradigm**. Case based learning systems, also known as *instance based learning*, represent knowledge in terms of specific cases or experiences and rely on flexible matching methods to retrieve similar cases and apply them to new situations. One of the common schemes of this paradigm is *nearest neighbour learning*, which finds the stored case nearest to the current situation according to some distance measure and then uses it for classification or prediction.

- **The Induction Learning Paradigm**. Induction learning systems are characterised by deriving or inducing a general rule from a set of examples [23], and usually employ decision trees, condition-action rules, or similar knowledge structures. Nodes in a decision tree involve testing a specific attribute, which usually compares the value of an attribute with a constant. Some (sub) trees, however, compare two or more attributes. Leaf nodes give a classification that applies to all examples which reach the leaf or a probability distribution over all possible classifications. To test the learnt decision tree, unknown examples can be applied by routing the tree according to the values of the attributes tested in successive nodes. When a leaf is reached examples are classified according to the class assigned to the leaf.

- **The Analytic Learning Paradigm**. Analytic learning systems represent knowledge as rules in logic form, but typically employ a performance system that solves multi-step problems using some search process. A common technique is to represent knowledge as Horn clauses, then to phrase problems as theorems and to search for proofs. Learning in this framework uses background knowledge to construct explanations (or proofs), then compiles the explanations into more complex rules that can solve similar problems. Most work on analytic learning has focused on improving the efficiency of search, but some has dealt with improvement of the classification accuracy.

3.2 Neural Networks

Artificial neural network research has experienced three periods of extensive activity [15]. The first peak appeared in the 1940s, when McCulloch and Pitts [17] introduced a binary threshold unit as a computational model for an artificial neuron and described a logical calculus of neural networks. The second occurred in the 1960s due to Rosenblatt's *perceptron* convergence theorem [26] and Minsky and Papert's work which showed the limitations of a simple perceptron [20]. Minsky and Papert's results greatly decreased the enthusiasm of many researchers, especially those in the computer science community [15]. This resulted in a long "pause" in neural network research for almost 20 years. The third period came from the early 1980s and since then neural network research has received considerable renewed interest. The major work includes Hopfield's energy approach [13] in 1982 and the backward error propagation learning algorithm for multilayer perceptrons (multilayer feed forward networks) popularised by Rumelhart et al. [28] in 1986[2]. With the development of various of types and applications, artificial neural network research has gradually become one of the two main strands (*symbolism* versus *connectionism*) in the area of artificial intelligence.

3.2.1 Terminology

A *neuron* (also called a *unit* or *node*) is the basic computational cell in a neural network. A node can have several *links* carrying signals into or out of it. Links are also referred to as *connections*. Each link has an associated *weight*. The weight can be looked upon as a factor which strengthens or weakens a signal which is fed into the link. In addition, a node usually has also an associated *bias*, which can be viewed as a constant input to the node from a "virtual" unit in the network. An artificial neural network in general is a collection of such units linked to each other in some manner. A typical unit or node is presented in figure 2.

In figure 2, the links, input weights, bias and the activation function (or transfer function, f) are presented. The input of the node i, net_i, is computed according to equation 2.

$$net_i = \sum_{k=1}^{n} W_{ik} \cdot O_k + b_i \tag{2}$$

Here n is the number of the nodes connecting into node i, W_{ik} is the weight on the connection to node i from node k, O_k represents the activation (output) of node k, b_i stands

[2]The backward error propagation algorithm was first proposed by Werbos [31] in 1974.

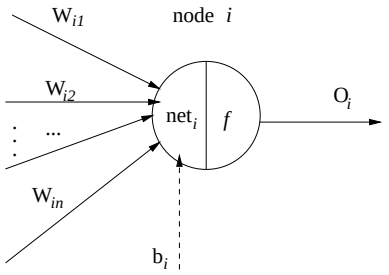

Figure 2: A typical network node or neuron

for the bias of node i. If node i is an input node, then $O_k = 1$ and $b_i = 0$, that is, input nodes usually have zero bias and the net input of an input node is the value of the input pattern at that node. Input nodes have their activations set to a value determined by the input pattern. The activation at non-input nodes is usually determined by applying an activation function to the net input to that node. The most commonly used activation function is the *logistic* or *sigmoid* function. According to this idea, the output of the node i, O_i, is presented in equation 3.

$$O_i = f(net_i) = \frac{1}{1 + e^{-\beta \cdot net_i}} \tag{3}$$

where β is a constant which can be used to change the shape of the curve of function f. The default value of β is 1.0.

Neural networks also distinguish their nodes as being input nodes, hidden nodes and output nodes. The input nodes receive signals from outside the network, for example, features or attributes for a problem domain. The output nodes collectively hold the results of the neural computation, for instance, the labels of the classes. The nodes between the input and output nodes are called intermediate or hidden nodes. There is only one input layer and one output layer in multilayer feed forward networks, however there might be more than one hidden layer in the network architecture. If there is more than one hidden layer, the layers, from the input layer to the output layer, are usually called the first hidden layer, the second hidden layer, etc. In this way, each layer except the input layer has incoming connections from the previous layer, and each layer except the output layer has the connections outgoing to the next layer. Figure 3 shows a typical three layer feed forward neural network.

In figure 3, there is only one hidden layer. The theoretical results provided by Irie and Miyake [14] and Funahashi [8] have proved that any continuous mapping can be approximated by a network with a single hidden layer, which means that one hidden layer is sufficient for any practical purpose and there is no need for more than one hidden layer. The collection of the inputs applied to the input nodes at one time constitutes an *input pattern* and the corresponding outputs produced form an *output pattern*. In supervised learning there is also a *target pattern* for each input pattern. Comparing the network actual outputs and the target patterns can decide whether the network has learnt the tasks.

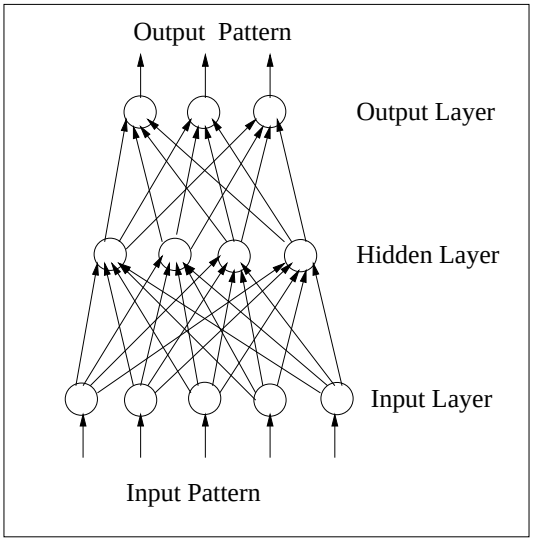

Figure 3: A typical three layer feed forward neural network.

3.2.2 Network Training

The main idea of the network training is presented as follows. Each input pattern in the training set is presented to the network which produces the *actual output*. This is compared with the target output. If the actual output matches (to some desired level of precision) the target output, the neural network is said to have been trained. Otherwise, the internal weights and biases of the network nodes need to be adjusted in such a way as to produce the target outputs. An *epoch* is a *cycle* in training which consists of presenting all the input patterns in the training set, passing the signals through the network and calculating the outputs, and adjusting the weights of the network if the actual outputs and the target outputs do not match. Typically the network weights have to be repeatedly adjusted over a number of epochs until a certain criterion is reached.

There are several algorithms for training multilayer feed forward neural networks, such as the backward error propagation algorithm, the back percolation algorithm, and the quickprop algorithm [33]. Among these, the *backward error propagation* (*BP*) is the most commonly used one. The details of the training algorithm are described by Rumelhart et al. [11, 27].

3.2.3 Tackling a Problem with Neural Networks

Assuming that a generalised tool based on the backward error propagation algorithm for training multilayer feed forward networks is available, the task of using these kinds of neural networks for any given problem becomes one of determining a suitable network architecture and a set of suitable parameters. More specifically, the following questions need to be considered before the experiments can be performed:

- How does one properly arrange the data for network training and for measuring the results?
- What is the number of output nodes?

- How many input nodes are needed?
- How many hidden layers are needed and how many nodes in each hidden layer?
- What values can be given for the parameters and variables for controlling the training process, for example, learning rate, range of initial weights, momentum and number of epochs?
- What termination strategies need to be applied during network training and how many runs do we perform for the problem?
- At what stage are the network weights updated when training patterns are presented?

3.3 Genetic Algorithms

The first genetic algorithm was developed by Holland at the University of Michigan beginning in the early 1960s. In the early 1970s, Holland formulated the "Schema Theorem". This in turn formed the basis of his landmark book *Adaptation in Natural and Artificial Systems* [12]. Since then, the genetic algorithm has gradually become one of the main learning paradigms.

In this subsection, we only review the main idea and genetic operators. The details of genetic algorithms can be found in Goldberg[9] and Holland [12].

3.3.1 The Main Idea

A genetic algorithm is a kind of search and learning method based on Darwinian natural selection theory. The technique involves generating a random initial population with a given number of individuals, each of which represents a potential solution to a given problem. Potential solutions are encoded into *chromosomes*, usually bit strings of some arbitrary length. Each individual's fitness as a solution to the problem is computed and evaluated against some designated criteria. Members of the population are then selected for reproduction based on their fitness and a new generation of potential solutions is generated from the offspring of the most fit individuals. The process of evaluation, selection, recombination is iterated until a certain criterion is reached.

The individuals in a population are evaluated by a fitness function. The fitness function should be able to assess the performance of an individual with reference to the problem for which it is a potential solution. The fitness function plays a very important role in the evolutionary process. If the fitness function is not properly set, the evolutionary learning would most likely fail. The fitness function in general varies with problem domains, since different problems have different goals.

The main goal of selection is to have some part of the individuals' genetic material propagated through to the next generation of potential solutions. A number of selection methods have been developed. Among them the *biased roulette wheel* mechanism [9] is the most commonly used one. In this method each individual in the current population has a slot on a roulette wheel proportional in size to that individual's fitness. The roulette wheel is spun once for each parent required. The winning individuals are paired for reproduction and recombination.

Recombination is achieved by one of the combination operators. There are two main genetic operators: *mutation* and *crossover*, which are used to produce the new offspring.

3.3.2 The Operators

Mutation is the random modification of the selected individuals. The main goal of mutation is to maintain the diversity in the population. An example of a single bit mutation is shown in figure 4.

Figure 4: An example of a single bit mutation.

Crossover operators approximate sexual reproduction in biology and are the combination of the genetic material from two selected individuals to produce offspring. Examples of crossover operators are: single point, two point, and uniform crossover. Single point crossover is achieved by randomly choosing a single point at which to separate, swap, and rejoin the bit strings. In two point crossover, two points are chosen at random and the segments of the bit string between the two points are exchanged and form the two new children. These two methods can suffer from the problems of destroying good schemata or incompletely combining the schemata.

Uniform crossover [2] is implemented by generating a bit mask equal in length to the chromosomes being manipulated, with the value of each bit being determined with some arbitrary probability. For each bit of the mask which has "1", the corresponding bit of the parent chromosomes is swapped before propagation to the offspring; and for each mask bit which has "0", the corresponding parent bits are propagated to the offspring unchanged. An example of uniform crossover is presented in figure 5. Uniform crossover can combine all possible schemata, even if it might be quite disruptive of schemata of any length.

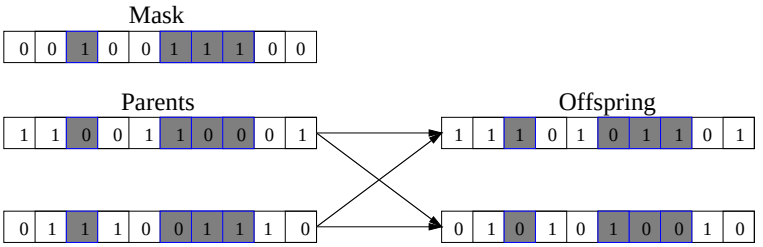

Figure 5: An example of uniform crossover

4 Further Work

In this report, we first reviewed the basic concepts, dimensions and commonly used architecture and techniques of speech recognition. After giving a series of concepts, data sets and major paradigms in machine learning, we reviewed two major learning approaches, neural

networks and genetic algorithms, which have been or have a great potential to be applied to speech recognition. In the appendices, some related work such as how to build a simple speech recognition system, existing HMM software packages and speaker recognition is described.

A list of open issues need to be further investigated:

- Review the state of the art of the HMMs for speech recognition;
- Review the work which has been done in speech recognition using neural networks and genetic algorithm;
- Further collect papers for isolated words recognition;
- Review pronunciation improvement systems;
- Review the first report Nerf-87-TR-01 and get a new version of the proposal.

References

[1] Ronald A. Cole, Joseph Mariani, Hans Uszkoreit, Annie Zaenen, and Victor Zue. *Survey of the State of the Art in Human Language Technology.* Cambridge University Press, 1996. ISBN 0-521-59277-1.

[2] K.A. DeJong and W.M. Spears. On the virtues of parameterised uniform crossover. In Richard K. Belew and Lashon B Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 230–236, San Mateo, July 1991. Morgan Kaufman.

[3] David B. Fogel. *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence.* IEEE Press, IEEE Neural Network Council, Inc., New York, 1995.

[4] Norman Foo. *Advanced Topics in Artificial Inteeligence – Proceedings of Australian Joint Conference on Artificial Intelligence (AI'99).* Springer-verlag, Berlin Heidelberg New York, 1999. Lecture Notes in Artificial Intelligence 1747, ISBN 3-540-66822-5.

[5] R. Friedberg. A learning machine, Part I. *IBM Journal of Research and Development*, 2:2–13, 1958.

[6] R. Friedberg, B. Dunham, and J. North. A learning machine, Part II. *IBM Journal of Research and Development*, 3:282–287, 1959.

[7] Jurgen Fritsch. Modular neural networks for speech recognition. Master's thesis, Interactive Systems Laboratories, Carnegie Mellon University — USA and University of Karlsruhe — Germany, August 1996.

[8] Ken-Ichi Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural Networks*, 2(3):183–192, 1989.

[9] D.E. Goldberg. *Genetic Algorithms in Search, Optimisation and Machine learning.* Addison Wesley, Reading, Ma, 1989.

[10] Jean Hennebert, Martin Hasler, and Herve Dedieu. Neural networks in speech recognition. http://citeseer.nj.nec.com/178766.html.

[11] Geoffrey E. Hinton. Connectionist learning procedures. *Artificial Intelligence*, 40(1-3):185–234, 1989. Reprinted in J. Carbonell, editor, "Machine Learning: Paradigms and Methods", MIT Press, 1990. Also appears as Technical Report CMU-CS-87-115 (version 2), Carnegie Mellon University, Pittsburgh, PA, December 1987.

[12] John Henry Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence.* Ann Arbor : University of Michigan Press; Cambridge, Mass. : MIT Press, 1975.

[13] J. J. Hopfield. Neural networks and physical system with emergent collective computational abilities. In *Proceedings of National academy of Sciences, USA79, vol. 2*, pages 554–558, 1982.

[14] B. Irie and S. Miyake. Capability of three-layered perceptrons. In *Proceedings of the IEEE 1988 International Conference on Neural Networks*, pages I(641–648), New York, 1988. IEEE. Vol. 1.

[15] Anil K. Jain, Jianchang Mao, and K. M. Mohiuddin. Artificial neural networks: A tutorial. *IEEE Computer*, 29(3):31–44, March 1996.

[16] John R. Koza. *Genetic programming : on the programming of computers by means of natural selection.* Cambridge, Mass. : MIT Press, London, England, 1992.

[17] W. McCulloch and W. Pitts. A logical calculus of the ideal immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[18] Wolfgang Menzel, Daniel Herron, Rachel Morton, Datio Pezzotta, Patrizia Bonaventura, and Peter Howarth. Interactive pronunciation training. *ReCALL*, 13(1):67–78, 2001. Cambridge University Press.

[19] Ryszard S. Michalski, Jaime G. Carbonell, and Tom M. Mitchell. *Machine Learning, An Artificial Intelligence Approach.* Tioga Publishing Company, Palo Alto, California, 1983.

[20] M. Minsky and S. Papert. *Perceptrons: An Introduction to Computational Geometry.* MIT Press, Cambridge, Mass., 1969.

[21] T. Mitchell. *Machine Learning.* McGraw-Hill, New York, 1997.

[22] Berndt Muller, Joachim Reinhardt, and Michael T. Strickland. *Neural Networks: An Introduction.* Springer-Verlag, Berlin Heidelberg, Germany, 2nd edition, 1995.

[23] J. Ross Quinlan. *C4.5: Programs for Machine Learning – Constructing Decision Trees*, chapter 2, pages 17–43. Morgan-Kaufmann, San Mateo, California, 1993. ISBN 1-55860-238-0.

[24] Soren Kamaric Riis. *Hidden Markov Models and Neural Networks for Speech Recognition.* PhD thesis, Department of Mathematical Modelling, Technical University of Denmark, Lyngby, 1998. http://citeseer.nj.nec.com/riis98hidden.html, ISSN 0909-3192.

[25] B. D. Ripley. *Pattern Recognition and Neural Networks.* Cambridge University Press, 1996. ISBN 0 521 46086 7.

[26] F. Rosenblatt. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms.* Spartan Books, New York, 1962.

[27] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart, J. L. McClelland, and the PDP research group, editors, *Parallel distributed Processing, Explorations in the Microstructure of Cognition, Volume 1: Foundations*, chapter 8. The MIT Press, Cambridge, Massachusetts, London, England, 1986.

[28] D. E. Rumelhart, J. L. McClelland, and the PDP research group. *Parallel Distributed Processing, Explorations in the Microstructure of Cognition.* The MIT Press, Cambridge, Massachusetts, London, England, 1986. Volume 1: Foundations.

[29] Mike Schuster. Neural networks for speech processing. Web page: citeseer.nj.nec.com/schuster98neural.html.

[30] Joe Tebelskis. *Speech Recognition using Neural Networks.* PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213-3890, May 1995. http://citeseer.nj.nec.com/tebelskis95speech.html.

[31] P. Werbos. *Beyond Regression: New tools for prediction and analysis in the behavioral science.* PhD thesis, Department of Applied Mathematics, Harvard University, Cambridge, Mass., 1974.

[32] Russ Wilcox. Commercial speech recognition. http://www.tiac.net/users/rwilcox/speech.html. This page was updated by Sam Quiring on January 2, 2002.

[33] Andreas Zell, Gunter Mamier, and et al. *SNNS User Manual.* University of Stuttgart, 1995. Version 4.1.

A Building A Simple Speech Recogniser

Doug Danforth[1992, Nov] provides a detailed account in article 253 in the comp.speech archives. This is a simple recognizer that should give you 85%+ recognition accuracy. The accuracy is a function of the words you have in your vocabulary. Long distinct words are easy. Short similar words are hard. You can get 98+% on the digits with this recognizer. An overview is given as follows:

1. Find the beginning and end of the utterance.
2. Filter the raw signal into frequency bands.
3. Cut the utterance into a fixed number of segments.
4. Average data for each band in each segment.
5. Store this pattern with its name.
6. Collect training set of about 3 repetitions of each pattern (word).
7. Recognize unknown by comparing its pattern against all patterns in the training set and returning the name of the pattern closest to the unknown.

This type of recognizer has been used by several companies such as Interstate Electronics. There are many variations on this theme: Use Mel-Ceptral rather than frequency bands, dynamic time warping rather than linear segment rule, Hidden Markov Models with no specific end point determination, etc.

If you use filter bands then you need to know how to construct a filter which has a center frequency and band width. There are many signal processing books that describe how to do this but can get quite technical very fast. In general, a simple "second order state space" filter works very well. Here, each filter is represented by a 2×2 matrix which specifies its center frequency and bandwidth along with a 2×1 vector, its state. The state is modified from sample to sample by first adding the input signal from whatever hardware board you have to one of the components of the state and then multiplying that state by the 2×2 matrix: add and rotate. The output of the filter is just one of the components of the state (it doesn't really matter which, the phase is just shifted slightly).

The 2×2 matrix is constructed as following:

$$R = r \begin{vmatrix} a & -b \\ b & a \end{vmatrix}$$

where $0 < r < 1$, $a = \cos(t), b = \sin(t)$.

The parameter r determines the width of the filter. If r is close to 1 then the width is very narrow and the output can grow very large for inputs with frequency in resonance with the filter. For r small the width is broad and the amplitude grows less strongly.

The parameter t is the frequency of the filter, small t corresponding to low frequency, large (near pi) t to high frequency. The filters usually spread over the range from 200Hz to

4000Hz. The spread should be heavy near the low frequency with fewer filters near the high (critical bands).

The output of a filter will look choppy and irregular just like the input but will be large for resonance input signals. One needs to smooth the output of each band filter by "lowpass" filtering the rectified fullwave (absolute value of) — make all negative values positive. This entails using a second stage with a single 1x1 state scalar that adds a fraction of the rectified bandpass filter output to a fraction of its value:

$$Lowpass := (1 - u) * Lowpass + u * |Bandpass| \tag{4}$$

where $0 < u < 1$.

Resample the Lowpass at about 200 times a second to use for the other parts of the pattern generation.

How many filters? How many segments? Well 16 for both works quite nicely. This gives a pattern of 256 numbers.[3]

How do you find the beginning and end of an utterance? Use a threshold for the total energy (square of the input signal) and remember that just because the signal drops below the threshold does not mean that the word is finished. It may come up again! Consider the word "it". There is a long pause between the "i" and the release of the "t" so you need allow for this. Again, other more sophisticated techniques can avoid having to make these "end point" decisions in this way but take more work to implement.

[3]The amplitude in each segment should be normalized otherwise loud sounds will look different from soft sounds even though the same word is spoken. The Interstate Algorithm actually just uses the amplitude difference (1 bit) from one lowpass filter to the next higher frequency within a segment. For 16 filters there are 15 differences. An increase gets a 1 bit. No change or decrease gets a 0 bit. The result is a 256 bit (pad low frequency with zeros) pattern. You can try other schemes as well.

B Discrete Hidden Markov Model Demonstration Software

Hidden Markov Models (HMMs) are widely used in speech recognition systems. Joe Picone has put together some demonstration software for basic discrete HMMs including Viterbi and Baum-Welch training and evaluation, random sequence generation (generating data from a model), and model updating (useful for incremental training). There is a simple demo program that supports all of these modes from command line arguments. This allows experiments to test the classic coin-toss examples commonly described in textbooks. The code closely parallels the following textbook [32]:

J.R. Deller, Jr., J.G. Proakis, and J.H.L. Hansen, Discrete-Time Processing of Speech Signals, MacMillan, 1993, ISBN: 0-02-328301-7.

The code is written in C++ and is intended to facilitate learning and understanding of the algorithms. The code is available on the ISIP web site:

http://www.isip.msstate.edu/software/

C Speaker Recognition

Speaker recognition is the process of automatically recognizing who is speaking on the basis of individual information included in speech signals. It can be divided into Speaker Identification and Speaker Verification. Speaker identification determines which registered speaker provides a given utterance from amongst a set of known speakers. Speaker verification accepts or rejects the identity claim of a speaker — the person they say they are.

Speaker recognition technology makes it possible for a speaker's voice to control access to restricted services, for example, phone access to banking, database services, shopping or voice mail, and access to secure equipment.

Both technologies require users to "enroll" in the system, that is, to give examples of their speech to a system so that it can characterise (or learn) their voice patterns.