



School of Engineering and Computer Science

COMP 307 — Lecture 09

Machine Learning 6
Neural Networks and Neural Engineering

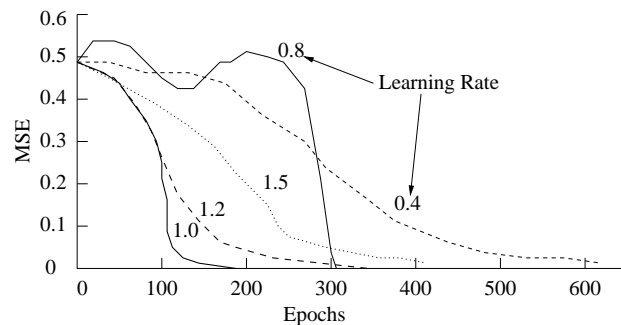
Mengjie Zhang (Meng)

mengjie.zhang@ecs.vuw.ac.nz

COMP307

ML6 (NNs and NE): 3

Learning Rate η



- A learning rate of 1.0 appears best for this problem
- There is usually an optimal learning rate, but it is problem dependent
- In practice $\eta = 0.2$ is good starting point

COMP307

ML6 (NNs and NE): 2

Outline

- Learning rate
- Overfitting
- Stopping criteria
- Local minima/optima
- Network architecture
- Momentum
- Parameters
- Other types of neural networks
- Problems

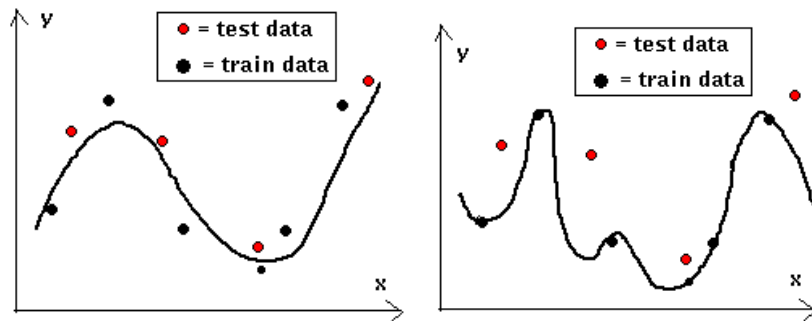
COMP307

ML6 (NNs and NE): 4

Overfitting

- The network has high accuracy on the data from which it was developed (training), but low accuracy on new data (test)
- Caused by
 - Training for too long
 - Having a network with too many nodes and weights
 - * Each weight (and bias) is a parameter that needs to be estimated
 - * The more parameters we have the more data we need for accurate estimates
 - Too few training examples
 - * How many examples do you need to learn properly
 - * Given that we have n examples, how well can we expect to learn

Overfitting (Continued)

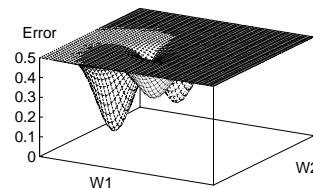


Small networks
Simple curve

Large nets
Complex curve

Local Minima

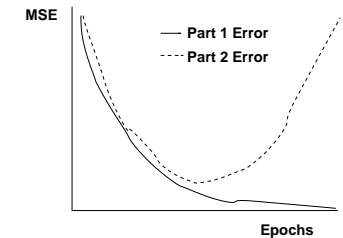
- Suppose we are training a network with two weights w_1, w_2
- For any given values of w_1 and w_2 we can work out the error.
- Plotting error vs weights might give such a surface/landscape



- Note in general we can't do this because there are too many dimensions
- What we desire is a trajectory of points which leads us to the global minimum
- A bad trajectory or starting point will take us to a local minimum
- Oscillating trajectory

When to Stop Training

- The *epoch/cycle control* strategy
- The *error control* strategy: TSS, MSE, RSME can be used.
- The *proportion control* strategy.
- The *user control* strategy.
- The *early stopping* strategy:
Validation control.



- Break the TRAINING set into 2 parts
- Use part 1 to compute the weight changes
- Every m (Typical values 10,50,100) epochs apply the partially trained network to part 2 (the validation set) and save the weights
- Goal: avoid overfitting

Local Minima

- How can you tell if a local minimum has been reached?
 - A number of runs with different starting points end with (very) different TSS errors
- What to do about a local minimum?
 - Nothing, if training is 'good enough'
 - Increase the value of the learning rate η
- Oscillation
 - The error graph is very 'jerky' without an overall downward trend.
 - Decrease the learning rate.
- Sometimes decreasing the learning rate as training proceeds works well.

Network Size/Architecture

- Usually the numbers of inputs and outputs is determined by the problem
- How many hidden layers? Hidden nodes?
- Theorem: One hidden layer is enough for any problem
- But, training might be faster with several layers
- Best is to have as few hidden layers/nodes as possible
 - Forces better generalization
 - Fewer weights to be found
- Determining the number of hidden layers/nodes
 - Make the best guess you can
 - If training is unsuccessful try more hidden nodes
 - If training is successful try fewer hidden nodes
 - Inspect weights after training. Nodes which have small weights can probably be eliminated

Online Learning vs Offline Learning

- Online learning: A variation of BP learning
 - At what stage are the network weights updated when training patterns are presented?
 - In general, two possibilities.
 - Online learning (or called the *stochastic gradient procedure*), where the weight changes are applied to the network after each training pattern.
 - off-line learning (or called the *true gradient procedure*) in which the weight changes are accumulated for all training patterns and the sum of all changes is applied after one full epoch. (The standard BP uses this)

Momentum

- Momentum: When updating a weight, add a contribution from time $t - 1$
- Weight update rule becomes:

$$\Delta w_{i \rightarrow j}(t) = \eta o_i o_j (1 - o_j) \beta_j + \alpha \Delta w_{i \rightarrow j}(t - 1)$$

- Momentum is controversial
 - One view: It's hopeless, don't ever use it
 - Opposite view: It's always better with momentum. You just have to get the right values for η and α

Main Parameters for Network Training

- Random_range(r): The random range ($[-r, r]$) of the initial weights.
- Learning rate(η): The constant in true gradient descent. The bigger the learning rate, the larger the changes in the weights.
- Momentum(α): A constant which determines the effect of past weight changes on the current direction of movement in weight space.
- Critical error (ce): The desired error (TSS, MSE or RMSE) for stopping network training. If the actual error is equal to or less than this error, the training will be terminated.

Tackling a Problem with NNs

Given a neural network package:

- How to 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?

Other Neural Networks

- Multilayer, feed forward, back propagation
 - the “standard”, for learning to classify or predict
- Recurrent/feedback networks
 - Cycles in networks, with or without time delay
 - Sequence prediction — can remember recent inputs
- Network for remembering patterns
 - Hopfield networks: “Content addressable memory”
 - Given part of or distorted pattern, find best matching pattern
- Networks for automatic clustering
 - Kohonen networks
 - Self organising maps (SOMs)

Neural Networks as Nonlinear Regression

- Very powerful, general function fitting mechanisms
 - can represent any function
 - (with enough hidden nodes)
- Output units represent “smoothed” hyperplane
- Hidden nodes transform space arbitrarily
- Useful in huge variety of domains

Summary

- Feed forward networks and neural engineering
 - Back propagation example
 - Learning rate
 - Overfitting
 - Stopping criteria
 - Local minima/optima
 - Network architecture
 - Momentum
 - Parameters
- Other types of neural networks
- Problems and directions
- Next week: Evolutionary Learning/Computing