

# Introduction to Artificial Intelligence



**COMP307**

**Neural Network 3: Neural  
Engineering**

Yi Mei

*yi.mei@ecs.vuw.ac.nz*

# Outline

- Weight update frequency
- Learning rate
- Overfitting
- Stopping criteria
- Local minima
- ANN architecture
- Momentum

# Weight Update Frequency

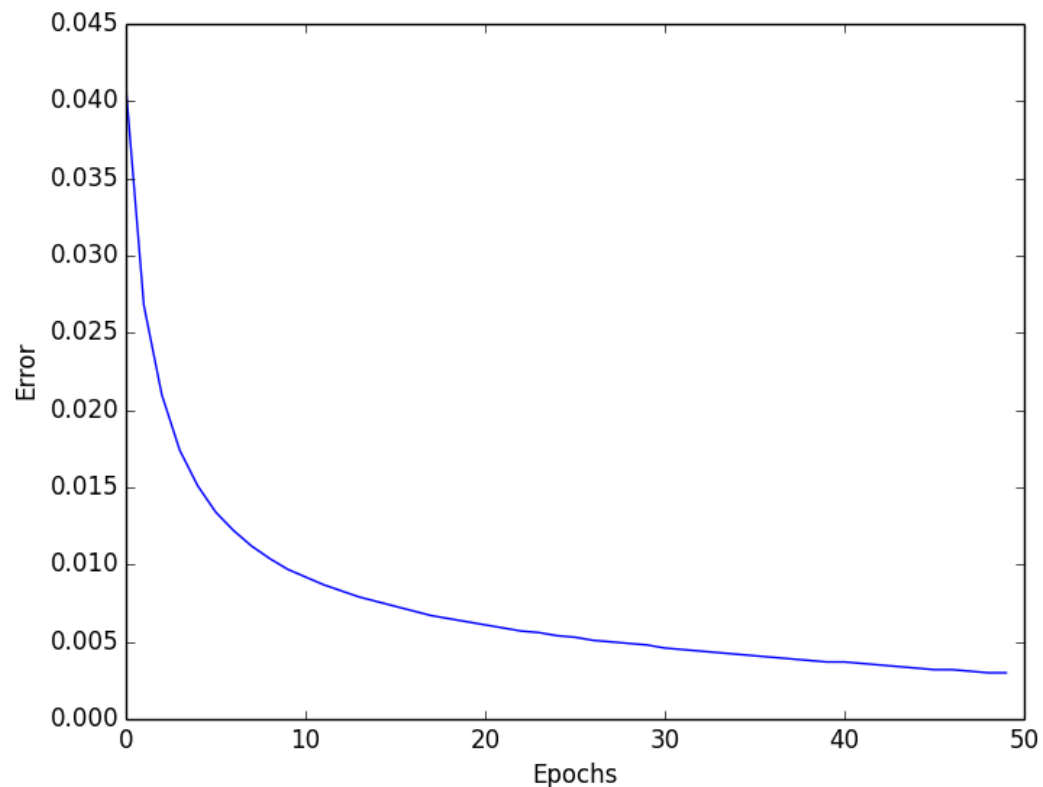
- All the weights are updated after **one feedforward pass** and **one backward propagation/pass**
- Frequency of weight update = Frequency of passes
- **Online learning**: a pass for **each training instance**
- **Batch learning**: a pass for a **batch** (a subset of training instances)
  - weight change is the **sum** of the changes for **all the instances in the batch**
- **Offline learning**: a pass for **all the training instances**
  - Weight change is the **sum** of the changes for **all the training instances**
- **Online** and **batch** learning are **stochastic** gradient descent
- **Offline** learning is **true** gradient descent

# Weight Update Frequency

- Assuming a weight  $w = 0.2$
- 4 training instances
- Online learning
  - Instance 1,  $\Delta w = 0.1$ ,  $w \rightarrow 0.3$
  - Instance 2,  $\Delta w = 0.05$ ,  $w \rightarrow 0.35$
  - Instance 3,  $\Delta w = 0.03$ ,  $w \rightarrow 0.38$
  - Instance 4,  $\Delta w = 0.01$ ,  $w \rightarrow 0.39$
- Offline learning
  - Instance 1,  $\Delta w = 0.1$ ,  $w = 0.2$  unchanged
  - Instance 2,  $\Delta w = 0.08$ ,  $w = 0.2$  unchanged
  - Instance 3,  $\Delta w = -0.03$ ,  $w = 0.2$  unchanged
  - Instance 4,  $\Delta w = 0.05$ ,  $w = 0.2$  unchanged
  - $w \rightarrow 0.2 + 0.1 + 0.08 - 0.03 + 0.05 = 0.4$

# Weight Update Frequency

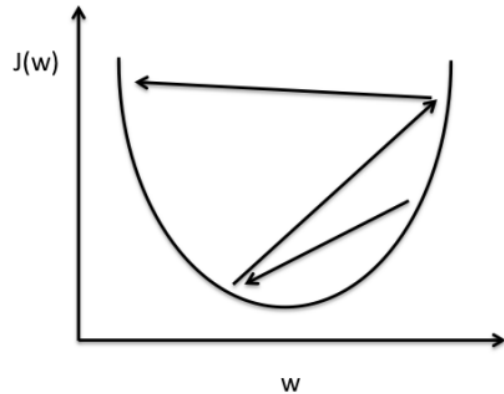
- **Epoch**: period when **all the training instances** are used once
- **#Iterations = #passes**
- 1000 training instances, batch size = 500, then need 2 iterations to complete one epoch



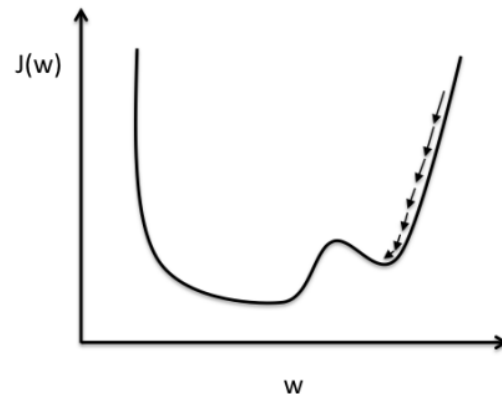
# Learning Rate

- Large learning rate may cause oscillating behaviour
- Small learning rate may cause slow convergence
- 0.2 is a good starting point in practice

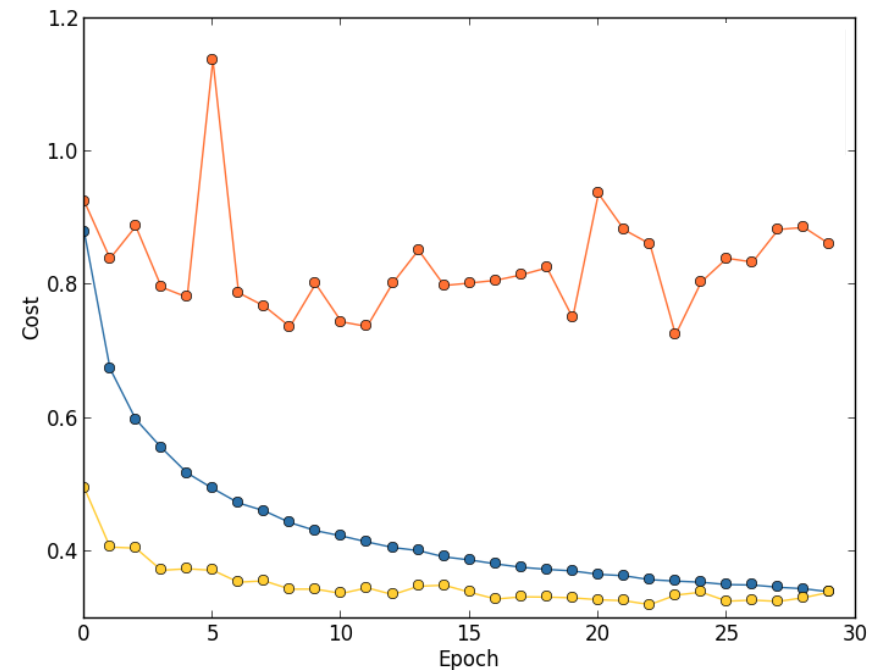
$$\Delta w_{i \rightarrow j} = \eta o_i o_j (1 - o_j) \beta_j$$



Large learning rate: Overshooting.

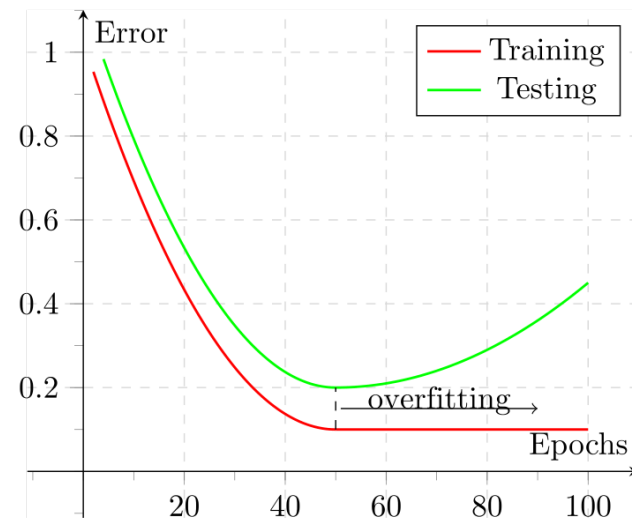
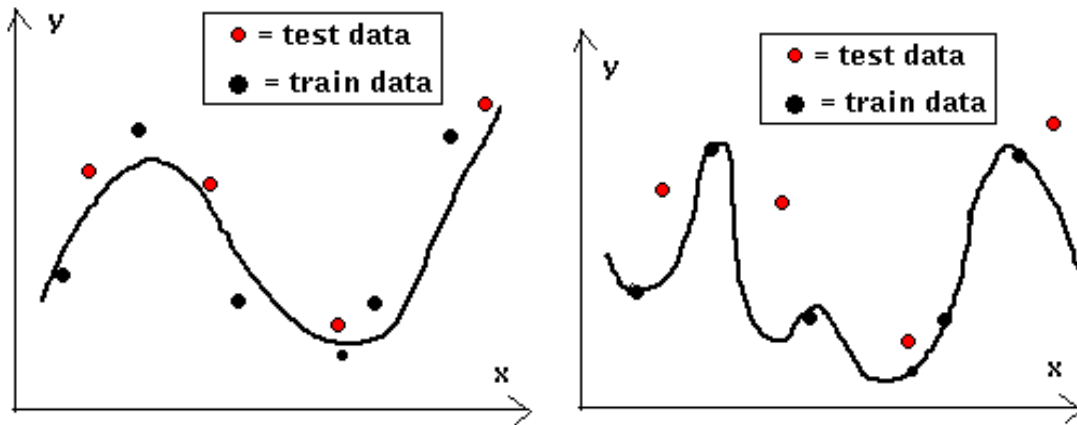


Small learning rate: Many iterations until convergence and trapping in local minima.



# Overfitting

- Has a very **high accuracy on the training set**, but **poor accuracy on the test set**
- Caused by
  - Training for **too long**
  - Too many weights (parameters) to train
  - Too few training instances
- The more parameters to train, the more data (training instances) we need to have an accurate estimation



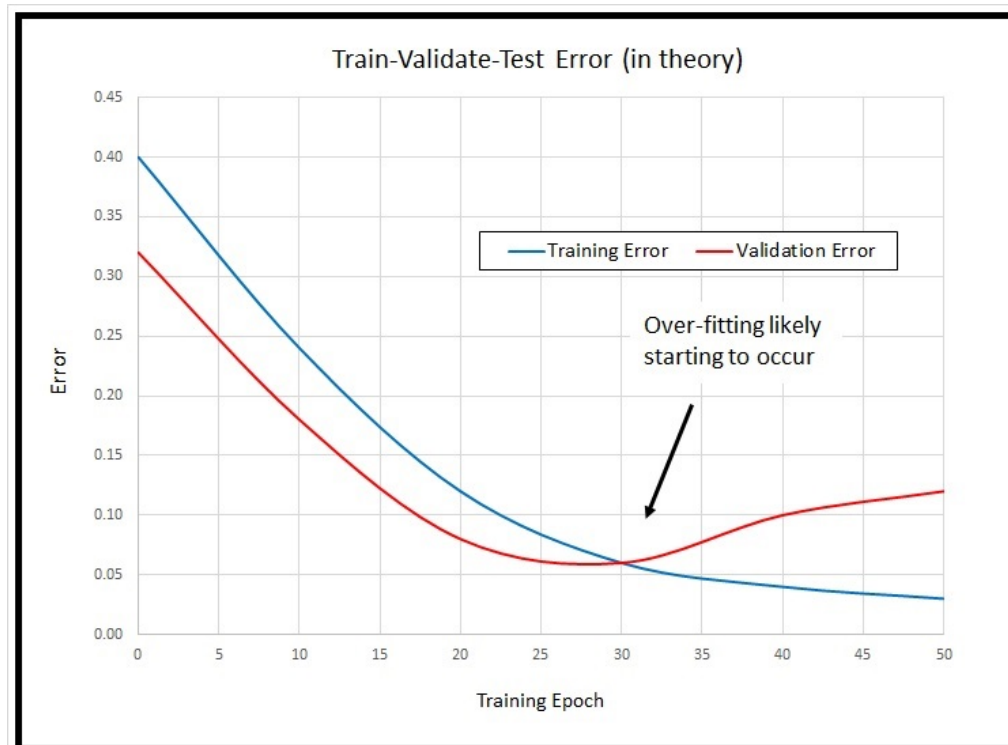
# Stopping Criteria

- When a certain number of **epochs** is reached
- When the **error** (e.g. mean/total squared error) on the training set is smaller than a threshold
- Proportion of correctly classified training instances (i.e. **accuracy**) is larger than a threshold
- **Early stopping** strategy
  - Validation control to **avoid overfitting**



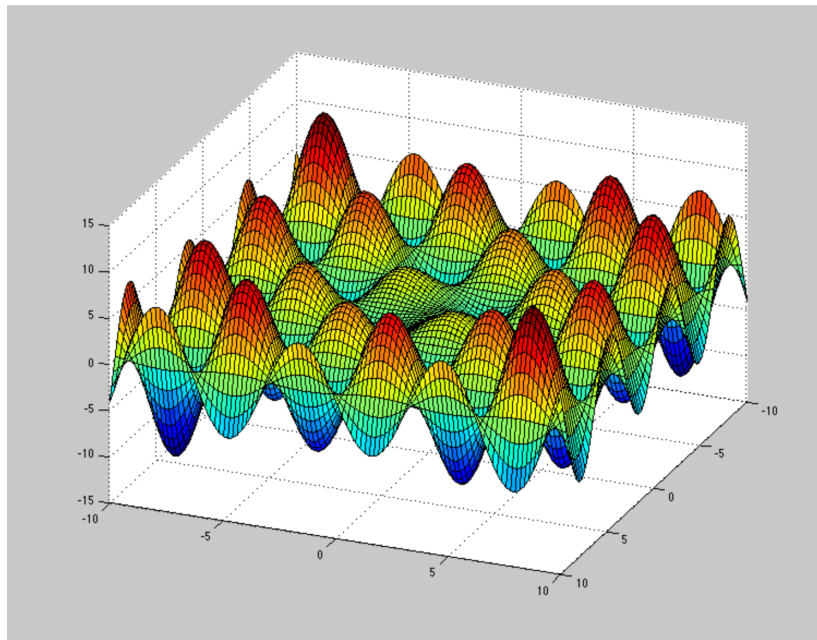
# Validation Control

- Break the training set into 2 parts
- Use part 1 to **compute the weight changes**
- Every  $m$  (e.g. 10, 50, 100) epochs apply the current NN to part 2 (**validation set**) to **calculate the validation error**
- Stop when the **error on the validation set is minimum**

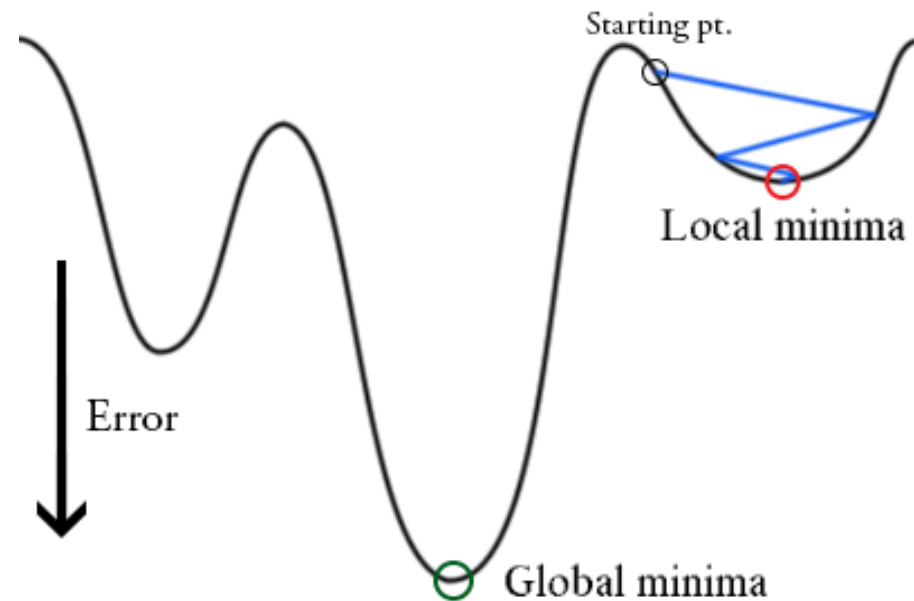


# Local Minima

- For each **weight vector**, we can calculate the **error** of the NN
- The **(weight vector, error) surface/landscape** can be very rugged: **many local minima**
- **Search**: a **trajectory of points** leading to the global minima
- A bad trajectory leads to **poor local minimum**



Surface with 2 weights



Surface with 1 weight

# Local Minima

- How can you tell if a local minimum is reached?
- What to do with a local minimum?

# Local Minima

- How can you tell if a local minimum is reached?
  - No improvement for a certain number of epochs
- What to do with a local minimum?
  - Nothing, if the training is “good enough”
  - Increase the learning rate
  - However, too large learning rate can cause oscillation
  - Start with a large learning rate, then decrease it as training proceeds

# ANN Architecture

- How many input and output nodes?
  - Usually determined by the problem
- Number of input nodes equals the number of features
- Number of outputs
  - 1 output nodes for **binary** classification (true/false)
  - **N** output nodes for **N-class** classification
  - Example: (1,0,0) = class 1; (0, 1, 0) = class 2; (0, 0, 1) = class 3

# ANN Architecture

- How many hidden layers/nodes?
  - Theorem: **one hidden layer is enough** for any problem
  - But training may be faster with several layers
  - Best to have **as few hidden layers/nodes as possible**: better generalisation, fewer weights to optimise (easier to solve)
  - Make the best guess you can
  - If training is unsuccessful try more hidden nodes
  - If training is successful try fewer hidden nodes
  - Observe weights after training: nodes with small weights can probably be eliminated

# Momentum

- Normal to have huge ANNs take days/weeks/months to train
- Speed up is important
- **Momentum** is a widely used approach
  - Use the gradient from last step(s)

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

- Is momentum always working?
- Have you used/seen momentum before?
- How to choose  $\eta$  and  $\alpha$ ?

# Design Questions

- How to properly **arrange the data** for network training and for measuring the results?
- Number of **input/output** nodes?
- How many **hidden layers** are needed and how many **nodes in each hidden layer**?
- Values for the parameters and variables for controlling the training process, for example, **learning rate**, **initial weights**, **momentum** and **number of epochs**?
- Stopping criteria (**validation control**)?
- How often are the weights changed (**batch size**)?



# Summary

- Different frequencies for weight update
  - Online, offline, batch learning
- Learning rate
  - Not too large nor too small, 0.2 is a good starting point
- Overfitting
- Stopping criteria
  - Validation control to avoid overfitting
- Local minima
- ANN architecture: #input/output/hidden nodes
- Momentum to speed up training