

# 31YB Lecture 9:

Mon, 27 Oct

## ANN / MLP Applications & Generalization Issues

- More MLP/BP issues
- Classification & Regression Problems
- More on Generalization Issues
- Introduction to Unsupervised Learning

---

---

---

---

---

---

---

---

### Review: 'Back-prop' algorithm summary

- ◆ Initialise weights at random, choose a learning rate  $\eta$
- ◆ Until network is trained:
  - ◆ For each training example (inputs and target outputs):
    - Do **forward pass** through net (with fixed initial/current weights) to produce **network outputs** (e.g. for J hidden layer nodes & N inputs in 2-layer MLP):  

$$y_k = f\left(\sum_{j=0}^J w_{jk} o_j\right)$$
 where  $o_j$  is output from each hidden node  $j$ :  $o_j = f\left(\sum_{i=0}^I w_{ij} x_i\right)$
    - For each **output unit k**, compute **deltas**:  $d_k = (y_{target_k} - y_k) y_k (1 - y_k)$
    - For hidden units  $j$  (from last to first hidden layer, for the case of more than 1 hidden layer) **compute deltas by back propagation**:  

$$d_j = o_j (1 - o_j) \sum_k w_{jk} d_k$$
    - For **all weights**, **change** each weight by **gradient descent**:  $\Delta w_{ij} = \eta d_j x_i$   
 Specifically, for a 2-layer MLP, for weight from input layer unit  $i$  to hidden layer unit  $j$  the weight changes by:  $\Delta w_{ij} = \eta d_j x_i$   
 And, for weight from hidden layer unit  $j$  to output layer unit  $k$ , the weight changes by:  $\Delta w_{jk} = \eta d_k o_j$

---

---

---

---

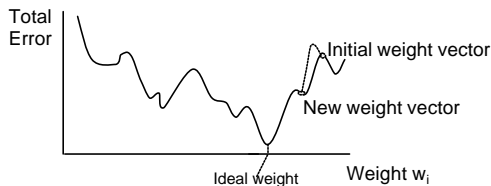
---

---

---

---

### Recap: Generalised Delta Rule or BP



- BP uses gradient descent technique (i.e. following opposite direction to that of the gradient of error with respect to weights) for changing weights
- Total error minimisation using BP is an example of a non-linear optimisation problem: many local minima exist in addition to the single desired global (error) minimum (represented by the ideal weight)
- There are many weights in general: resulting in a complex high-dimensional weight space

---

---

---

---

---

---

---

---

## Recap: Some MLP/BP Issues - Choice of Experimental Parameters

- What values should be used for  $\eta$  and  $\alpha$  ?
- How many layers should be used?
- How many units in each hidden layer?
  - Generally a value for  $\eta$  between 0.03 & 0.1, and for  $\alpha$  between 0.7 and 0.95, found useful but...? (*answer*: choice dependent on problem, and trial & error approach usually easier in practice!)
  - Use one hidden layer, at least until you are convinced you need more than one (how does one know?) (*answer*: problem dependent, trial & error)
  - it is usually best to use a smaller number of hidden units than the number of inputs, in order to allow a compact representation to form. But there can be other issues as well, and there is no good answer to this problem, yet!

---

---

---

---

---

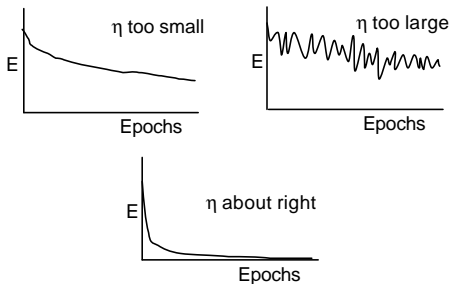
---

---

---

## Recap: Some MLP/BP Issues - Choice of Experimental Parameters - contd....

- What value for  $\eta$  ?




---

---

---

---

---

---

---

---

## Recap: Some MLP/BP Issues - contd..

- And, importantly, when to stop training?
  - How does one decide, either network has converged or, it will not converge (question also in DR)
    - One can stop training, when: (I) number of epochs exceeds some pre-specified value, or (II) the error is less than some pre-specified value
- ... but, which error should we try to minimize?
  - If one chooses the usual mean/sum squared error, one runs the danger of having almost correct output for almost all of the output units, but having a relatively large error for one unit, or one pattern.
  - To avoid this, one can use different error function: however, one can also use usual error measure for training, but alter the criterion for training cessation
  - One can continue until:  $E_{\text{maxerror}} = \max_p \max_k |Y_{\text{target}_k^p} - Y_k^p|$  (where  $p$  indexes the patterns, and  $k$  the output units) is less than some prespecified amount.  $E_{\text{maxerror}}$  gives largest error on a single output unit for any pattern
- ... and what value of error do we stop at?
  - For classification networks with discrete desired outputs,  $0 < E_{\text{maxerror}} < 0.5$ , otherwise? (*answer*: problem dependent, trial & error)

---

---

---

---

---

---

---

---

## ANN/MLP Applications: Classification and Regression

- ✦ *Classification* problems assign each input vector to a discrete class or category  
e.g. optical character recognition.
- ✦ In *regression* problems, the output variables are continuous  
e.g. exchange rate prediction.
- ✦ Both are cases of *function approximation* - see *Hand-out 2 + Tutorial 2 examples*

---

---

---

---

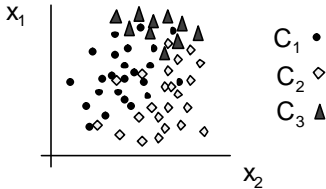
---

---

---

---

### Classification



In general:

$C$  different classes:  $C_1, C_2, \dots, C_c$

$N$ -dimensional feature vectors:

$$\mathbf{x} = [x_1 \ x_2 \ \dots \ x_n]$$

---

---

---

---

---

---

---

---

### Classification

#### Two classes

Classes:  $C_1$  and  $C_2$

Given examples, learn a function  $y = f(\mathbf{x})$

$y = 1$  if  $\mathbf{x}$  is a member of  $C_1$

$y = -1$  if  $\mathbf{x}$  is a member of  $C_2$

#### General case

Classes:  $C_1, C_2, C_3, \dots, C_c$

Learn  $c$  functions using multi-output ANN:  $y_k = f_k(\mathbf{x})$

such that  $y_k = 1$  iff  $\mathbf{x}$  is a member of  $C_k$

---

---

---

---

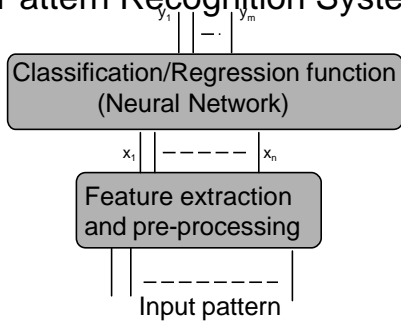
---

---

---

---

## A Pattern Recognition System



---

---

---

---

---

---

---

---

## Pattern Recognition

### 1. Feature extraction (*Representation*)

- What are good features ?
- What is a good combination of features ?
- How many features ?

### 2. Classification/Regression

- Which functions ? (e.g. linear, quadratic...)
- How to learn from examples ?
- How to ensure good generalisation ?

---

---

---

---

---

---

---

---

## Real world Example: Character Recognition for Pen Computing

### Task:

Design a pattern recognition system which can recognise characters as one of the 26 letters of the alphabet as they are written using a pen input device.

**What features might you try to extract ?**

---

---

---

---

---

---

---

---

## Character Recognition Features

- Area, height, width, orientation, elongation ...
- Co-ordinates (relative) of corners, junctions, points of high curvature ...
- Pressure on each pixel, after image processing, e.g. contrast normalisation, blurring ...
- Temporal features: duration, velocity, acceleration

---

---

---

---

---

---

---

---

## Invariance

- Good features are *invariant* under expected transformations such as translation, scaling and rotation e.g.

<u>Feature</u>	<u>Invariant under:</u>
Area	Translation, Rotation
Height, Width	Translation
Elongation	Trans., Rot., Scaling
Velocity	Translation

N.B. "The Curse of Dimensionality" (Bellman 1961) -

- Adding features can *reduce* performance !
- No. features = dimensionality of input space
- Training data is limited
- High dimensional feature spaces will be sparsely populated with data points.

---

---

---

---

---

---

---

---

## Revisit: **Generalization Issues:** Which family of functions ?

- A particular neural network can implement a particular family of functions  $y(\mathbf{x}; \mathbf{w})$ , where  $\mathbf{w}$  denotes the function parameters or 'weights' to be learned from the training data.
- See next

---

---

---

---

---

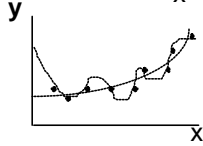
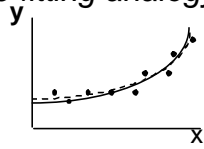
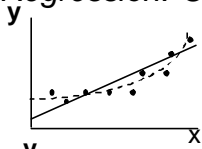
---

---

---

## Which family of functions ?

Regression: Curve fitting analogy



Which best fits the data ?  
Answer: Which will generalise to new (unseen or test) data the best?

---

---

---

---

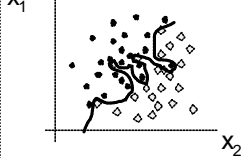
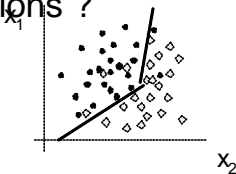
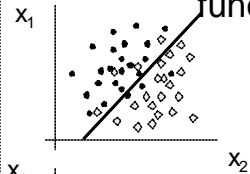
---

---

---

---

## Classification: Which family of functions ?



- Linear ?
- Piece-wise linear ?
- Highly non-linear ?
- Which will ensure correct classification of new data?

---

---

---

---

---

---

---

---

## Which family of functions ?

Depends on:

- The true nature of the 'underlying' function  $f$  which generated the data.
- The amount of training data available

---

---

---

---

---

---

---

---

## Can we assess whether ANN will generalize appropriately?

- **One popular approach:** Divide available training data into 2 parts: use one part (training sub-set) for training network, and second part (test sub-set or also known as **validation set**) for testing generalization performance of network prior to actual testing on real-world (unseen) test data - process known as CROSS-VALIDATION
- One does lose potential training info., but gain the possibility of knowing/predicting generalization performance prior to real testing on unseen data
- An additional possibility:
  - train up many networks on training subset
  - assess models produced by seeing how well they generalise (on the test subset, also known as **validation set**) cf. Assignment A
  - choose the best model.

---

---

---

---

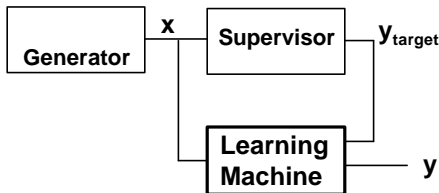
---

---

---

---

## Review: Supervised Learning



---

---

---

---

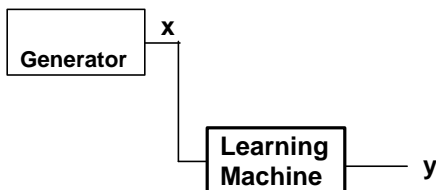
---

---

---

---

## Unsupervised Learning



**Training:** Learn from input data,  $x$  (e.g. extract useful information from data and form a new representation of the data  $x$  (see Handout 2 for more details)

**Testing:** Given any  $x$ , output a value  $y$  'describing'  $x$  (such as in problems of prototyping, clustering similarity - see Handout 2 for more details)

---

---

---

---

---

---

---

---

## Summary

- Review of MLP/BP training Issues
- ANN/MLP applications: Classification/Regression
- Generalisation Issues
- Introduction to Unsupervised Learning
  
- NEXT Lecture 10, Mon, 3<sup>rd</sup> Nov: More on Unsupervised Learning & introduction to Radial Basis Function (RBF) networks
- Tutorial 2: Thurs, 6<sup>th</sup> Nov, during lecture slot (10AM), 2A87B (available on website)
- Practical 2: Thurs, 6<sup>th</sup> Nov, 2-3PM (available on website)
- Assignment due back on 20<sup>th</sup> November!

---

---

---

---

---

---

---

---

## Next Lab (Labsheet 2)

- **Character Recognition Example using the MLP**
- MLP network designed to learn to discriminate between the different letters of the alphabet. The 7 x 5 input layer receives an 'image' of a character. The output layer has 26 units, one for each letter of the alphabet.

---

---

---

---

---

---

---

---