

# 31YB Lecture 11:

Monday, 10 Nov

## Overview

- More on Radial Basis Function (RBF) Network
- Training RBFs
- Comparison: RBF vs. MLP

---

---

---

---

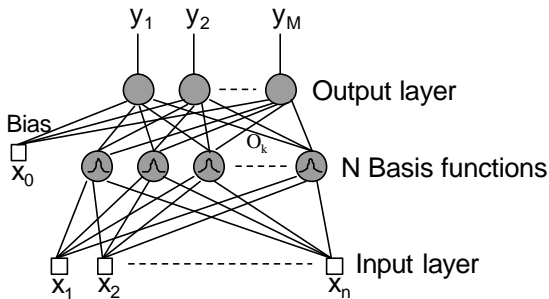
---

---

---

---

### Recap: Radial Basis Function Network



Here the RBF's are shown as Gaussian functions.  
 P.B. There are NO weights connecting Input Layer and Hidden layer (comprising basis functions) – see next slide

---

---

---

---

---

---

---

---

### Recap: Gaussian basis functions: (NOT examinable)

Each RBF hidden unit output,  $o_k$ , for  $k=1, \dots, N$ , is given by:

$$o_k = f(\mathbf{x} - \boldsymbol{\mu}_k) = \exp \left\{ -\frac{\|\mathbf{x} - \boldsymbol{\mu}_k\|^2}{2\sigma_k^2} \right\}$$

- Each hidden unit has parameters  $(\boldsymbol{\mu}_k, \sigma_k^2)$ ;  $\boldsymbol{\mu}$  is called the mean or expected value of Gaussian distribution, and  $\sigma^2$  is called the variance (it controls the spread of the distribution about  $\boldsymbol{\mu}$ ) – see next slide example
- Each hidden unit is *centred* on  $\boldsymbol{\mu}$  which can be thought of as a *prototype* - (known/stored input)
- The unit's output  $o_k$  only gets large when network input  $\mathbf{x}$  is close to its own stored prototype  $\boldsymbol{\mu}_k$  (as in Gaussian distribution)- closeness is measured by e.g. Euclidean distance  $\|\cdot\|^2$ , which for a 1-dimensional input,  $x_1$  is  $\|\mathbf{x} - \boldsymbol{\mu}\|^2 = (x_1 - m)^2$
- Each RBF network output,  $y_j$ , for  $j=1, \dots, M$ , is given by:

$$y_j = \sum_{i=0}^N w_i o_i = \sum_{k=0}^N w_k f(\mathbf{x} - \boldsymbol{\mu}_k) = \sum_{k=0}^N w_k \exp \left\{ -\frac{\|\mathbf{x} - \boldsymbol{\mu}_k\|^2}{2\sigma_k^2} \right\}$$

---

---

---

---

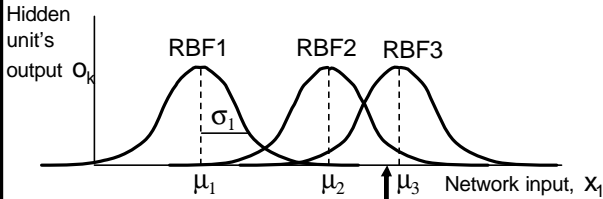
---

---

---

---

## Recap: Radial Basis Functions: Example



- ♦ 3 RBF hidden units with centres or means  $\mu_1$ ,  $\mu_2$  &  $\mu_3$
- ♦ The input value indicated by arrow above results in largest output value  $o_3$  from RBF3 (since input value is nearest to  $\mu_3$ ); and  $o_3 > o_2 > o_1$  (since input is farthest away from centre  $\mu_1$  of RBF1).
- ♦ Overall RBF network output is weighted sum of the above three hidden unit outputs

---

---

---

---

---

---

---

---

## Recap: Notes: (see Hand-out 3 for more details)

- Each RBF node in the hidden layer responds to input only in some subspace of the input space. When input is far away from its own centre  $\mu$ , (many radii (standard deviations,  $\sigma$ ) away), then the output of that unit will be so small as to be ignorable
- Each RBF has a receptive field, that is, an area of the input space to which it responds
- **BIOLOGICAL PLAUSIBILITY:**
- RBF is more BIOLOGICALLY PLAUSIBLE, since many sensory neurons respond only to some small subspace of the input space, and are silent in response to all other inputs.
- There is considerable evidence that neurons in visual cortex display RECEPTIVE FIELDS of this form: they are maximally sensitive to some specific stimulus, & their output falls off as presented stimulus moves away from this "best" stimulus.

---

---

---

---

---

---

---

---

## RBF nets are 'Universal Approximators'

- As with MLPs, RBF nets can approximate any reasonable function (classification or regression), given a sufficient number of hidden units.
- But this theoretical result does not tell us how to determine and train the network.

---

---

---

---

---

---

---

---

## Training RBF nets

RBF networks are trained by

- deciding on how many hidden units there should be
- deciding on their centres and the sharpnesses (standard deviation) of their Gaussians
- training up the output layer.

Can train in 2 stages

1. Hidden layer

- estimate parameters for each hidden unit  $k$  (whose output depends on distance between input and a stored prototype  $Q_k = f(x - \mu_k)$ )
- e.g. Gaussian parameters:  $m_k, \sigma_k^2$
- *Unsupervised* training process

2. Output layer

- set the weights (including bias weights)
- the same as training a single layer perceptron: each unit's output depends on weighted sum of inputs,  $y_j = \sum_{i=1}^n w_{ij} \phi_i$
- using for example, the Delta Rule (DR) or Least Mean Squares (LMS) algorithm
- *Supervised* training process

---

---

---

---

---

---

---

---

## Training the hidden layer: basis function optimisation

- First decide how many hidden units
- Then learn parameters ( $m_k, \sigma_k^2$ ) from data
- One approach is to try to represent the probability density (pdf) of the input data
- Curse of dimensionality
  - *may need very many units for high-dim. data*
  - *density estimation problematic*

---

---

---

---

---

---

---

---

## Basis function optimisation: some practical methods

- I: Subset of data points
- II: Clustering algorithms
- III: Gaussian mixture models (NOT Examinable)

---

---

---

---

---

---

---

---

## Basis function optimisation I: Subset of data points

- Choose random subset of data points and assign basis function centres,  $m_k$ , to these.
- Use heuristic to set width parameters,  $\sigma_k$ 
  - set all  $\sigma_k$  equal to some multiple of the average distance between centres.
  - or allow widths to vary: e.g. set  $\sigma_k$  based on average distance to nearest neighbours.

---

---

---

---

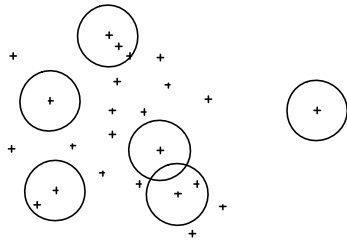
---

---

---

---

## Basis function optimisation I: Subset of data points



- ◆ 6 basis functions (hidden units) with equal  $\sigma$
- ◆ Centred on random 6 of 24 training data points

---

---

---

---

---

---

---

---

## Basis function optimisation I: Subset of data points

- simple and fast
- useful as initialisation for better, iterative methods
- poor density estimation

---

---

---

---

---

---

---

---

## Basis function optimisation II: Clustering

- Partition data into clusters
- Assign a basis function to each cluster
- This models distribution of input data more accurately than the random subset method
- Clustering algorithms are iterative
  - e.g. K-means clustering

---

---

---

---

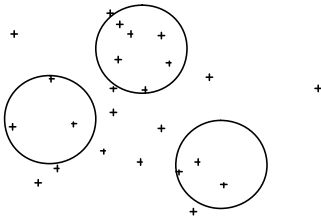
---

---

---

---

## Example of Clustering



- ◆ Data points assigned to 3 separate clusters
- ◆ One basis function per cluster

---

---

---

---

---

---

---

---

## More on Clustering

- Partition data into separate clusters
- Cluster membership is neither probabilistic nor fuzzy:
  - $P(k)$  is always exactly 0 or 1
- Clustering algorithms are iterative
  - e.g. K-means clustering...

---

---

---

---

---

---

---

---

## K-Means Clustering

- Partitions data into  $K$  disjoint sets or clusters  $S_1, S_2, \dots, S_K$
- Error for  $k^{\text{th}}$  cluster:

$$E_k = \sum_{\mathbf{x} \in S_k} \|\mathbf{x} - \boldsymbol{\mu}_k\|^2$$

- Want to minimise total error:

$$E = \sum_{k=1}^K E_k$$

---

---

---

---

---

---

---

---

## K-Means Algorithm (Lloyd 1982)

1. Assign points at random to the  $K$  sets
2. **Repeat**
  - Compute mean vector,  $\mathbf{m}_k$ , of each set
  - Reassign each point to the set with nearest mean (*i.e.* choose  $k$  with smallest  $\|\mathbf{x} - \mathbf{m}_k\|$ )

**Until** no change in sets

It can be shown that at each iteration, the value of  $E$  will not increase

---

---

---

---

---

---

---

---

## Basis function optimisation: supervised learning possible too!

- Setting up hidden layer using density estimation/unsupervised learning takes no account of target outputs -> sub-optimal
- Hence, usually only output layer is then trained in a supervised mode, using e.g. the Delta Rule
- However, it is also possible to train *both* layers together using supervised training (cf MLP). Sacrifices speed and interpretability.
- Can be used for fine-tuning after initial unsupervised training

---

---

---

---

---

---

---

---

## MLP

v

## RBF

- Hidden unit activation is constant on a *hyperplane*
- *Distributed* representation: typically many hidden units contribute to the output for a given input
- Difficult to interpret weights
- Training can be very *slow*
- Accuracy often a little better than RBFs
- Hidden unit activation is constant on a *hypersphere*
- *Localised* representation: typically only a few hidden units are active for a given input
- Interpretation as prototypes
- Layers can be trained separately -> *fast* training
- Easy to identify outliers (data points distant from all the training data)

---

---

---

---

---

---

---

---

## More on the RBF :

- RBF's are useful for both classification & regression (mapping), as MLP/BP systems. The concepts of training & testing are same.
- Note that if the widths (or radii)  $\sigma$  is very small, then training pertains to using each RBF unit for just one point in training set => lack of generalisation, as in MLP. Similarly, if  $\sigma$  is too large, one can get overgeneralization.
- Using supervised learning to choose centres & widths can optimise performance.
- If one knows that a problem is particularly sensitive to some small area of input space,
  - or, if one is having particular problems with data from some part of the input spaceone can choose to place an RBF unit with centre there: *thus domain knowledge can be used to guide network construction*

---

---

---

---

---

---

---

---

## Relationship of MLP, RBFs to Fuzzy Sets/Fuzzy Logic (NOT Examinable)

- Fuzzy set theory is the theory of sets in which elements are members of the set with a set membership function, which has a value between 0 and 1, where
  - 0 means definitely NOT a member
  - and 1 means definitely a memberand intermediate values have intermediate meanings
- An MLP output unit with a logistic output could be interpreted as a set-membership function..
- Another interpretation: use actual level of output as strength of network's conviction in some decision
- RBF units have outputs between 0 & 1, & this can be considered as directly encoding fuzzy-set information
- If output of RBF network is between 0 & 1 (as per logistic unit), output can be interpreted in same way as MLP.

---

---

---

---

---

---

---

---

## Summary

- RBF networks
- Training: Basis function optimisation
- Comparison with MLPs

### Next lecture (12), thurs, 13 Nov

- Temporal Learning & Recurrent Neural Nets

**Note:** Practical 3 (on RBF): Thurs, 13th Nov, 1:30-3PM (also any queries on your assignment will be welcome!)

---

---

---

---

---

---

---

---