ARTIFICIAL NEURAL NETWORK

Unit-2: Perceptron

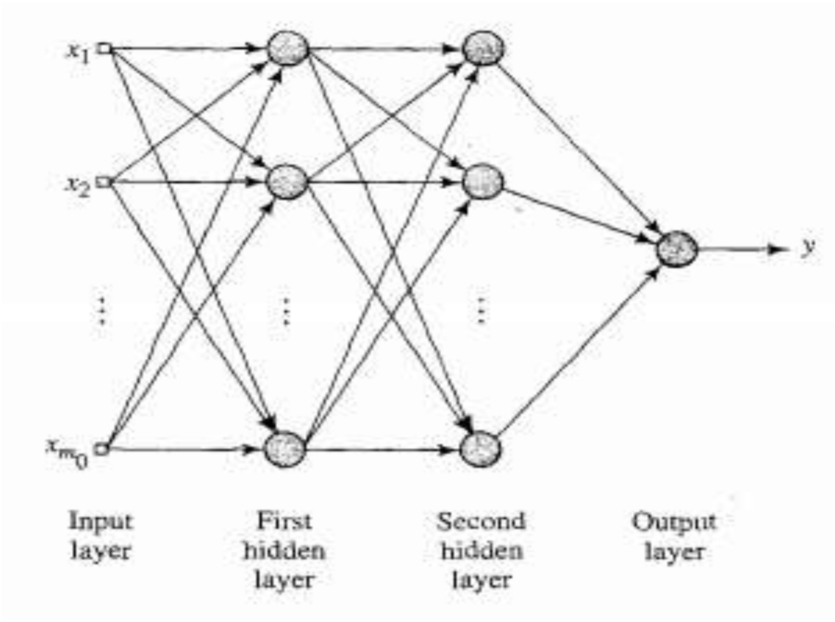
Ms. Swetha R.
Department of Electronics and Communication Engineering
PES University

BPA-Feature detection

- Hidden layer neurons play a important role in the operation of MLP with BPA because they act as a feature detectors.
- As the laerning process progresses, the hidden neurons begin to discover the salient features that characterizes the traing data.
- Performing non linear transformation from the input space to hidden space.
- For example,non linearly separable classes will be easily separated in the hidden space

Artificial Neural Network-Perceptron

Back-propagation and differentiation



The above MLP is parameterised by an architecture A (representing a discrete parameter) and a weight vector W

Artificial Neural Network-Perceptron

Back-propagation and differentiation

Let Alj denote the part of the architecture extending from the input layer to node j in layer (l=1,2,3). Accordingly we may write

$$F(W,X)=\varphi(A_{31})$$

$$\frac{\partial F(W,X)}{\partial w_{3lk}}=\varphi'(A_{3l})\varphi(A_{2k})$$

$$\frac{\partial F(W,X)}{\partial w_{2kj}}=\varphi'(A_{3l})\varphi'(A_{2k})\varphi(A_{1j})w_{3lk}$$

$$\frac{\partial F(W,X)}{\partial w_{1ji}}=\varphi'(A_{3l})\varphi'(A_{1j})X_j\left[\sum_k w_{3lk}\varphi'(A_{2k})w_{2kj}\right]$$

Back-propagation and differentiation

The sensitivity of F(w,x) is

$$S_w^F = \frac{\partial F / F}{\partial w / w}$$

Home Work:

1. Jacobian Matrix
2. Hessian Matrix

Multi-Layer Perceptron: Generalization

- A network is said to be generalised when the input-ouput mapping computed by the network is correct for test data not used in the training.
- To understand this concept, let us assume NN is performing the non-linear input-output mapping: Curve fitting problem. This is nothing but interpolation problem
- Training data helps in learning process.
- when a test data is presented to NN, network computes the ouput.
- When a network learns to many input-output samples the network may be end up with memorizing the training set- over fitting/over trained

Multi-Layer Perceptron: Generalization

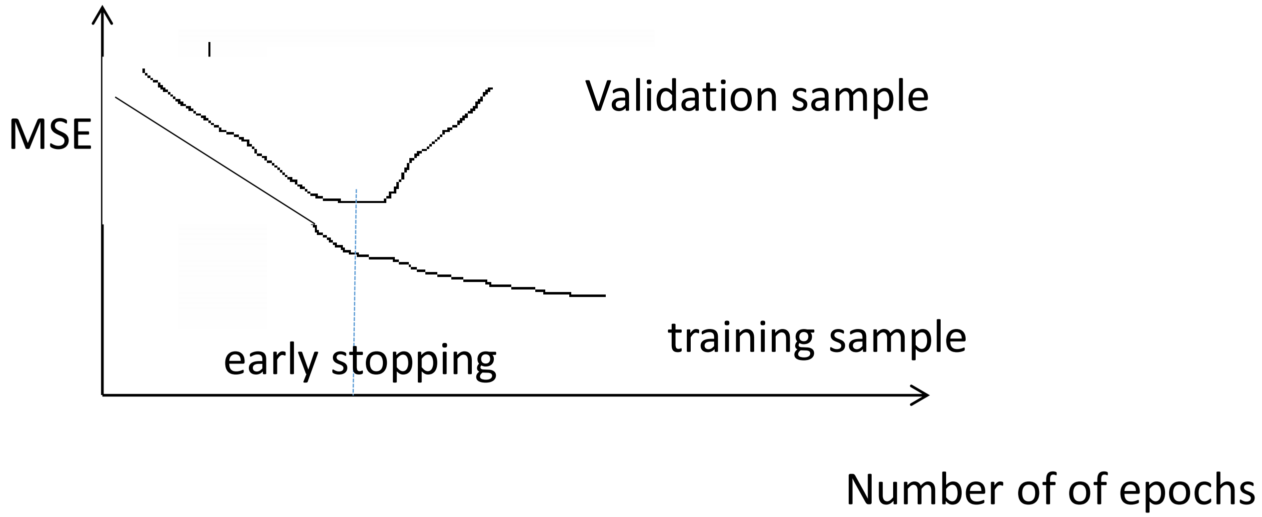
- When the network is over trained, it looses the ability of generalize.
- Generalization is effected by following factors:
  - Size of the training data and how close to the interest
  - Architecture of the NN
  - Complexity of the problem
- For good generalization the size of the training set N is given by N=O(W/e)
- where e is error permitted on the test data and O(.) is the order of the quantity enclosed within.

Multi-Layer Perceptron: Cross Validation

- The available input data is randomly divided into 2 sets: training set and test set.
- The training set is further divided into 2 disjoint sets:
- Estimation subset used to select the model
- Validation subset used to validate the model
- Cross-validation is useful in desigining a large NN with good gerneralisation properties.

Artificial Neural Network-Percepton

Multi-Layer Perceptron: Cross Validation



Multi-Layer Perceptron: Network Pruning Methd

- Choosing a suitable architecture for a NN for a given problem is very tedious job.
- Over sized topology has the following drawbacks:
  - high demand on the computational resource
  - increase in training time
  - non convergence of parameter
  - decrease in the generalization capability
  - shoots up the cost of hardware
  - less eficient.

Artificial Neural Network-Percepton

Multi-Layer Perceptron: Network Pruning Methd

- We can overcome these problems in following ways:
- Growing approach.
- Pruning Method
- Pruning method:
  - Deletion
  - Regularization
- Hessian based Network Pruning:
- The basic idea is to take the 2nd derivative of error susrface

Multi-Layer Perceptron: Network Pruning Methd

- First we construct a local model of error surface by predicting the effect of perturbation in W
- Consider

$$Eav(w+\Delta w) = Eav(w) + \frac{\partial Eav}{\partial w}\frac{\Delta w}{1!} + \Delta w^T \frac{\partial^2 Eav}{\partial w^2}\frac{\Delta w}{2!} + H.O.T$$

$$Eav(w+\Delta w) = Eav(w) + \frac{\partial Eav}{\partial w}\frac{\Delta w}{1!} + \Delta w^T H \frac{\Delta w}{2!} + H.O.T$$

- Next we need to identify the set parameters whose deletion from the network will cause least increase in the Eav.

Multi-Layer Perceptron: Network Pruning Methd

- Assumptions:
- Set 2nd term in the equation to 0 after training
- error surface around the local minimal is highly quadratic.

$$\Delta Eav = Eav(w+\Delta w) - Eav = \frac{1}{2}\Delta w^T H \Delta w$$

Goal: set one of the synapatic weight to zero to minimize the increamental increase in Eav to do so we choose

$$u_i^T \Delta w + w_i = 0$$

Multi-Layer Perceptron: Network Pruning Methd

- minimize the quadratic form with respect to incremental change in the weight vector subject to the constraint that

$$u_i^T \Delta w + w_i = 0$$

$$S = \frac{1}{2} \Delta w^T H \Delta w + \lambda \left(u_i^T \Delta w + w_i \right)$$

$$S_i = \frac{w_i^2}{2\left[H^{-1}\right]_{i,j}}$$

- **Home work:**

1. Virtues and limitation of back-propagation learning

2. BPA as an approximation of functions

THANK YOU

Ms. Swetha R.
Department of Electronics and Communication Engineering

swethar@pes.edu

+91 80 2672 1983 Extn 753