

Lecture 14.

MLP (VI): Model Selection

Outline

- Universal approximation property
- Network structure selection
- Structural Design of MLP
 - Dynamic and Static Growing Methods
 - Dynamic and Static Pruning Methods
- Batch Learning Methods

Configuration of MLP

- The choice of number of hidden nodes is
 - problem and data dependent;
 - even the optimal number is given, training may take long and finding optimal weights may be difficult.
- Methods for changing the number of hidden nodes dynamically are available, but not widely used. Two approaches: Growing vs. pruning.

Universal Approximation of MLP

- Let R be a finite region in the feature space and $g(x)$ be an unknown function that is continuous over R .
 - Let $f(W,x)$ be a 2-layer MLP with sigmoidal activation function at the hidden layer neurons and linear activation function at the output neuron.
 - Then there exist a set of weights W such that $f(W,x)$ can approximate $g(x)$ arbitrarily close.
 - This is an existence theorem.
- A three-layer MLP with two layers of hidden nodes, each with hyper-plane net function and sigmoidal activation function, is capable of learning any classification tasks.
 - The trick is to partition the feature space into unions of convex regions. Each region may be labeled with the same or different class labels. Then a set of hidden nodes can be used to learn each convex region.

What we know so far ...

- For most problems, one or two layers of hidden layers will be sufficient. If there are too many hidden layers, it may take many training epochs for the delta error to be propagated backward so that learning takes place.
- However, the number of hidden nodes per layer really depends on many issues and no good hint is available.
- If too many hidden nodes are used, there may be too much computation during training. If too few hidden nodes are used, the model complexity may be too low to learn the underlying structure of the data.
- Often repeated experimentation is the best way to figure out how many hidden nodes are best for the problem on hand.

Dynamic Net Growing Methods

- (1) Dynamic Node Generation (T. Ash, 1989) – Adding new nodes when output error curve flatten.
- (2) Restricted Coulomb Energy (Reilly et al., 1982) – Add new hidden units when none of the existing hidden units activated by the input.
- (3) Grow and Learn (Alpaydin, 1990) – Hidden units compete to perform classification. If not successful, a new hidden unit is added. A *sleep* procedure is also devised to remove redundancies among hidden units.
- (4) Cascade Correlation (Fahlman & Lebiere, 1990) – When a mapping can not be learned, a new hidden layer with one hidden unit is added while all previous hidden units' weights *frozen*.

Dynamic Net Growing Methods

- When to add a new hidden unit?
 - When total error ceases decreasing;
 - When a specific training pattern can not be learned
- When to stop adding new hidden units?
 - When all training patterns are correctly classified
 - When total approximation error $<$ a preset bound
 - When a combined cost criteria reaches minimum
(Adding more will cause it to increase!)
- What to do after adding each new hidden unit?
 - Retrain the entire net;
 - Train the new weights only, freeze the rest

Dynamic Net Growing Methods

- Local Representation Approach: Lateral inhibition/ competitive learning used so that only a single hidden unit will be activated for a particular input — Similar to the LVQ (SOFM) method
- Approximation Approach:
Total cost = approx. cost (noisy data, classification error) + generalization cost (No. of Hidden units)
- COMMON PROBLEM – Too many redundant hidden units may grow !
SOLUTION – Pruning after growing

Dynamic Net Pruning Methods

- Weight Decaying: (Rumelhart, 1988; Chauvin, 1990; Hanson & Pratt 1990) –

$$\text{Cost } \mathbf{E} = \sum_m (t^{(m)} - z^{(m)})^2 + \lambda \sum_{i,j} W_{ij}^2 / (1 + W_{ij}^2)$$

The second term is for regularization.

- Then $\partial \mathbf{E} / \partial w_{ij} = -2(t^{(m)} - z^{(m)}) \frac{\partial z^{(m)}}{\partial w_{ij}} + \lambda \frac{2w_{ij}}{(1 + w_{ij}^2)^2}$
 For large w_{ij} , the second term ~ 0 — No effect.
 For small w_{ij} , the second term $\sim 2w_{ij}$ — Penalized.

Static Net Pruning Methods (I)

- Skeletonization (Mozer and Smolensky, 1989) – The least relevant units will be removed first.

$$\text{Relevance}(i) = \partial \mathbf{E} / \partial y(i)$$

- Optimal Brain Damage (Le Cun et al. 1990) – The least salient weight is removed.

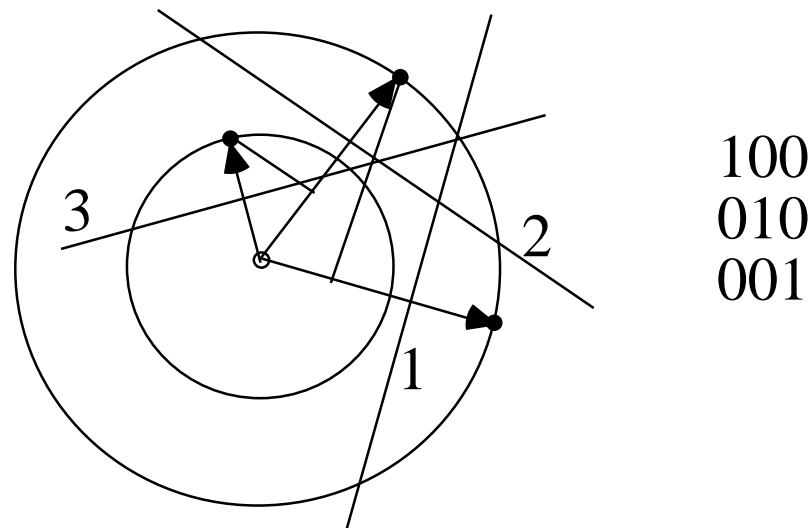
$$\text{Saliency} = \partial^2 \mathbf{E} / \partial [w(i,j)]^2 \Big|_{\partial \mathbf{E} / \partial w(i,j) = 0}$$

Static Net Pruning Methods (II)

- Hidden Unit Reduction Method (Hu et. al 1990) –The set of hidden units whose outputs are most independent are retained.
- Frobinius Norm Approximation Method (Kung & Hu 1990) – Hidden units whose output best approximate the activation of the trained network are retained.

Perfect MLP Training

- A batch computing method to achieve zero MLP training error WITHOUT iterative training. If M samples available, requires $M-1$ hidden units.
- Basic Idea: Use hidden neuron to encode each sample by a unique code: $x \dots x 10 \dots 0$ where $x = 0$ or 1 .



Perfect Training Algorithm

1. Sort all input $x^{(k)}$; $1 \leq k \leq K$ such that $\|x^{(k)}\| \leq \|x^{(k+1)}\|$.
2. For $k = 1$ to $K-1$ Do Add k -th neuron with weight vector w_k , bias q_k s. t.

$$w_k = x^{(k)}, \theta_k = -(1/2)[\|x^{(k)}\| + \max([x^{(k)}]^T x^{(k')}; k' > k)].$$

3. Compute hidden node output matrix $Y_{K \times (K-1)}$ for all inputs. Y matrix has 1 along its main diagonal.
4. Compute weights and bias of output neurons as:

$$[q \ W] = [\underline{1} \ Y]^{-1} f^{-1}(\underline{I})$$

\underline{I} : target vector (all inputs), $f^{-1}(\bullet)$: inverse activation function.

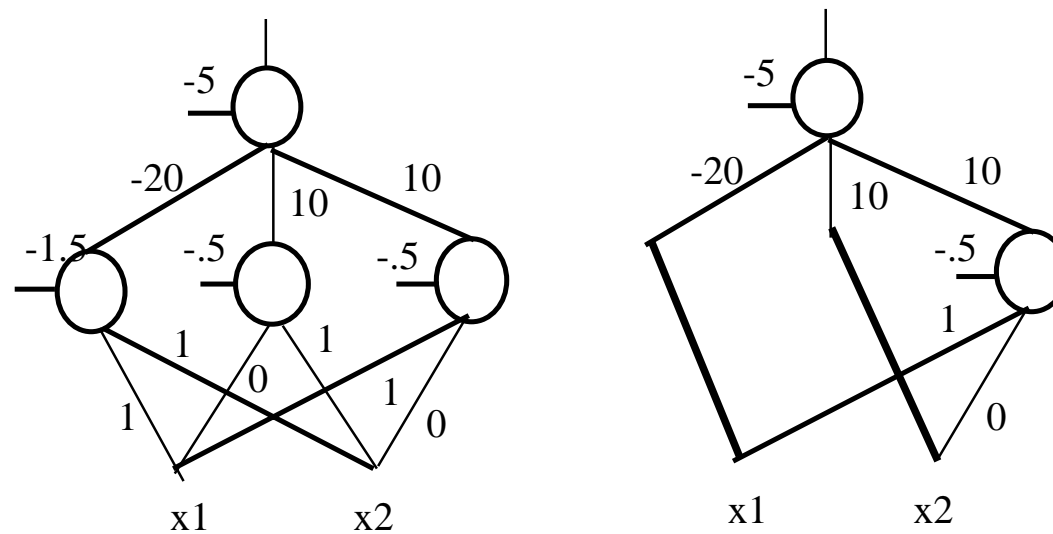
XOR Training Example

$$\mathbf{X} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \\ 0 & 0 \end{bmatrix} \quad \mathbf{T} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \quad \text{and } f^{-1}(\mathbf{T}) = \begin{bmatrix} -5 \\ 5 \\ 5 \\ -5 \end{bmatrix}$$

$$[\boldsymbol{\theta}_{\text{hidden}} \quad \mathbf{W}_{\text{hidden}}] = \begin{bmatrix} -1.5 & 1 & 1 \\ -0.5 & 0 & 1 \\ -0.5 & 1 & 0 \end{bmatrix} \quad \mathbf{Y} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

$$[\boldsymbol{\theta}_{\text{out}} \quad \mathbf{W}_{\text{out}}] = [-5 \quad -20 \quad 10 \quad 10]$$

XOR Result and Pruning



Left two neurons have the same output as X ,
hence can be eliminated w/o affecting the result.