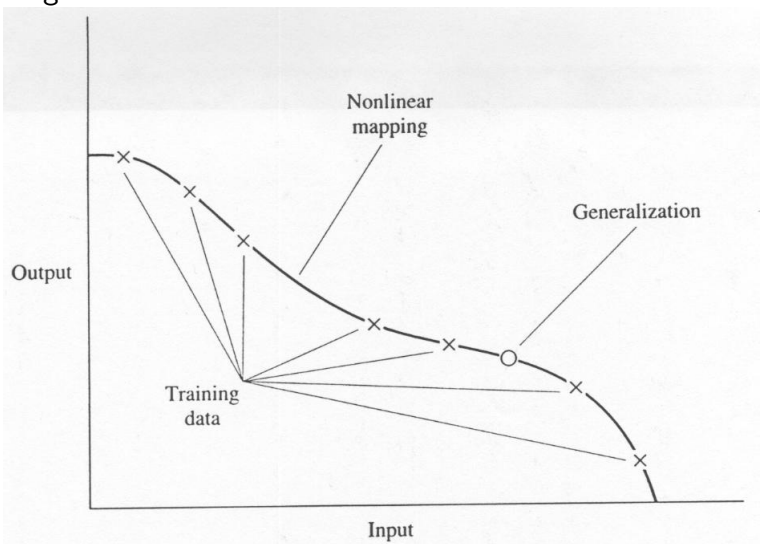
4.12 Generalization

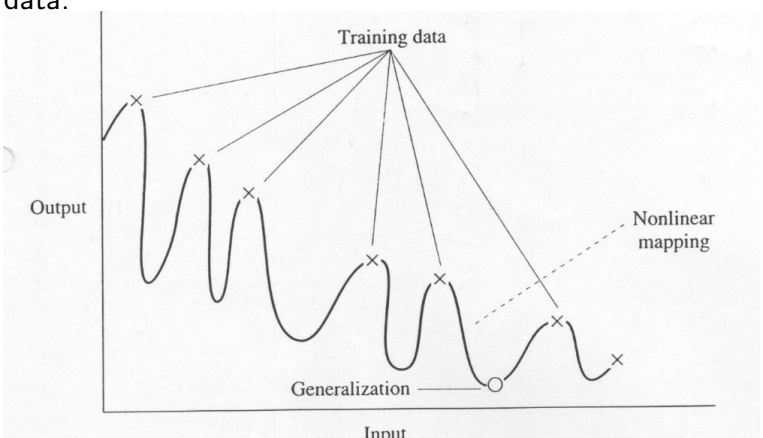
- In back-propagation learning, as many training examples as possible are typically used.
- It is hoped that the network so designed generalizes well.
- A network *generalizes* well when its input-output mapping is (almost) correct also for test data.
- The test data is not used in creating or training the network.
- Assumption: test data comes from the same population (distribution) as the training data.

- Training of a neural network may be viewed as a curve fitting (nonlinear mapping) problem.
- The network can simply be considered as a nonlinear input-output mapping.
- Generalization can be studied in terms of the nonlinear interpolation ability of the network.
- MLP networks with continuous activation functions perform useful interpolation because they have continuous outputs.

- An example of good generalization for a data vector not used in training.



- Using too many training examples may lead to a poor generalization ability.
- This is called *overfitting* or *overtraining*.
- The network then learns even unwanted noise present in the training data.



- More generally, it learns "features" which are present in the training

set but actually not in the underlying function to be modeled.

- Basic reason for overfitting: there are more hidden neurons than necessary in the network.
- Similar phenomena appear in other modeling problems if the chosen model is too complicated, containing too many free parameters.
- For example least-squares fitting, autoregressive modeling, etc.
- *Occam's razor* principle in model selection: select the simplest model which describes the data adequately.
- In the neural network area, this implies choosing the smoothest function that approximates the input-output mapping for a given error criterion.
- Such a choice generally demands the fewest computational resources.

Sufficient Training Set Size for a Valid Generalization

- Three factors affect generalization:
  1. The size and representativeness of the training set.
  2. The architecture of the neural network.
  3. The physical complexity of the problem at hand.
- Only the first two factors can be controlled.
- The issue of generalization may be viewed from two different perspectives:
  - *The architecture of the network is fixed.* Determine the size of the training set for a good generalization.
  - *The size of the training set is fixed.* Determine the best architecture of network for achieving a good generalization.
- Here we focus on the first viewpoint, hoping that the fixed architecture matches the complexity of the problem.

- Distribution-free, worst-case formulas are available for estimating the size of sufficient training set for a good generalization performance.
- See section 2.14; skipped in this course.
- However, these formulas give often poor results.

- *A practical condition for a good generalization:*
- The size N of the training set must satisfy the condition

$$N = O\left(\frac{W}{\varepsilon}\right)$$

- Here W is the total number of free parameters (weights and biases) in the network.
- ε denotes the fraction of classification errors permitted on test data (as in pattern classification).
- $O(.)$ is the order of quantity enclosed within it.
- Example: If an error of 10% is permitted, the number of training examples should be about 10 times the number of free parameters.
- Justifications for this empirical rule are presented in the next section.

4.13 Approximations of Functions

- A MLP network trained with back-propagation is a practical tool for performing a *general nonlinear input-output mapping.*
- Let m_0 be the number of input nodes (neurons), and $M = m_L$ the number of output nodes.
- The input-output mapping of the MLP network is from m_0-dimensional input space to M-dimensional output space.
- If the activation function is infinitely continuously differentiable, the mapping is also.
- A fundamental question: *What is the minimum number of hidden layers in a MLP network providing an approximate realization of any continuous mapping?*

Universal Approximation Theorem

- The *universal approximation theorem* for a nonlinear input-output mapping provides the answer.
- The theorem is presented in Haykin's book, pp. 208-209.
- Its essential contents are as follows:
- *Let $\varphi(.)$ be a nonconstant, bounded, and monotonically increasing continuous function.*
- *Let I_{m_0} denote the m_0-dimensional unit hypercube $[0,1]^{m_0}$, and $C(I_{m_0})$ the space of continuous functions on I_{m_0}.*
- *For any given function $f \in C(I_{m_0})$ and $\varepsilon > 0$, there exist an integer M and sets of real constants α_i, b_i, and w_{ij}, where $i = 1, \ldots, m_1$ and $j = 1, \ldots, m_0$ so that:*

- *The function*

$$F(x_1, \ldots, x_{m_0}) = \sum_{i=1}^{m_1} \alpha_i \varphi \left(\sum_{j=1}^{m_0} w_{ij} x_j + b_i \right) \tag{1}$$

is an approximate realization of the function $f(.)$.

- *That is,*

$$\mid F(x_1, \ldots, x_{m_0}) - f(x_1, \ldots, x_{m_0}) \mid < \varepsilon$$

for all $x_1, x_2, \ldots, x_{m_0}$ *that lie in the input space.*

- The universal approximation theorem is directly applicable to multilayer perceptrons.

- The logistic function $\varphi(v) = 1/[1 + \exp(-v)]$ is a nonconstant, bounded, and monotonically increasing function.

- Furthermore, Eq. (1) represent the output of a MLP network described as follows:

1. The network has m_0 input nodes with inputs $x_1, x_2, \ldots, x_{m_0}$, and a single hidden layer consisting of m_1 neurons.
2. Hidden neuron i has synaptic weights $w_{i1}, \ldots, w_{im_0}$, and bias b_i.
3. The network output is a linear combination of the outputs of the hidden neurons, with $\alpha_1, \ldots, \alpha_{m_1}$ defining the weights of the output layer.

- The universal approximation theorem is an *existence* theorem.
- In effect, the theorem states that *a MLP network with a single hidden layer is sufficient for uniform approximation with accuracy* ε.
- However, the theorem does not say that a single hidden layer is optimal with respect to:
  - learning time
  - ease of implementation
  - generalization ability (most important property).

Bounds on Approximation Errors

- This theoretical treatment is not essential in this course.
- A result worth mentioning: the size of the hidden layer m_1 must be large for getting a good approximation.

Curse of Dimensionality

- This part is skipped, too, though it contains some interesting results.
- One important matter: multilayer perceptrons are more effective than for example polynomials or trigonometric functions in approximation.
- That is, the number of terms required for sufficient approximation grows slower with the dimension of the problem.
- The reason is basically that nonlinearities are used in an efficient way in MLP networks.

Practical Considerations

- The universal approximation theorem is important from a theoretical viewpoint.
- It gives a rigorous mathematical foundation for using multilayer perceptrons in approximating nonlinear mappings.
- However, the theorem is not constructive.
- It does not actually tell how to specify a MLP network with the stated approximation properties.
- Some assumptions made in the theorem are unrealistic in most practical applications:
  - The continuous function to be approximated is given.
  - A hidden layer of unlimited size is available.
- A problem with MLP's using a single hidden layer:
  the hidden neurons tend to interact globally.

- In complex situations, improving the approximation at one point typically worsens it at some other point.
- With two hidden layers, the approximation (nonlinear mapping) process becomes more manageable.
- One can proceed as follows:
    1. *Local features* are extracted in the first hidden layer.
       - The input space is divided into regions by some neurons.
       - Other neurons in the first hidden layer learn the local features characterizing these regions.
    2. *Global features* corresponding to each region are extracted in the second hidden layer.
       - Neurons in this layer combine the outputs of the neurons describing certain region.
- This procedure somehow corresponds to piecewise polynomial (spline) approximation in curve fitting.

4.14 Cross-Validation

- It is hoped that an MLP network trained with back-propagation learns enough from the past to generalize to the future.
- How the network parameterization should be chosen for a specific data set?
- This is a model selection problem: choose the best one of a set of candidate model structures (parameterizations).
- A useful statistical technique for model selection: *cross-validation*.
- The available data set is first randomly partitioned into a training set and a test set.
- The training set is further partitioned into two disjoint subsets:
  - *Estimation subset*, used to select the model.
  - *Validation subset*, used to test or validate the model.

- The motivation is to validate the model on a data set different from the one used for parameter estimation.
- In this way, the training set can be used to assess the performance of various model.
- The "best" of the candidate models is then chosen.
- This procedure ensures that a model which might in the worst case end up with overfitting the validation subset is not chosen.
- The use of cross-validation is appealing when one should design a large network with good generalization ability.
- For MLP networks, cross-validation can be used to determine:
  - the optimal number of hidden neurons.
  - when it is best to stop training.

- These matters are described in the next subsections:
  - Model Selection
  - Early Stopping Method of Training
  - Variants of Cross-Validation

- They are skipped in this course.

4.16 Virtues and Limitations of Back-Propagation Learning

- Back-propagation (BP) is the most popular algorithm for supervised training of multilayer perceptrons (and neural networks in general).
- It is basically a gradient (derivative) technique, not an optimization method.
- Back-propagation has two distinct properties:
  - It is *simple* to compute locally.
  - It performs *stochastic* gradient descent in weight space.
- These two properties are responsible for the advantages and disadvantages of back-propagation learning.

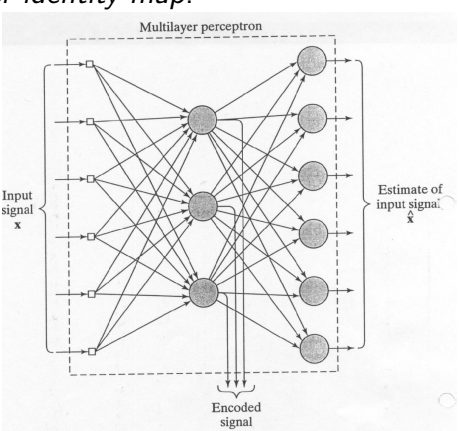
Connectionism

- BP is an example of *connectionist paradigms*.
- They use local computations only for processing information in a neural network.
- *Locality constraint:* a computing neuron needs information only from neurons connected physically to it.
- Local computations are preferred in the design of artificial neural networks for three principal reasons:
  1. Biological neural networks use local computations.
  2. Local computations are fault-tolerant against hardware errors.
  3. Local computations favor the use of computationally efficient parallel architectures.
- BP has been realized using VLSI architectures and parallel computers (point 3).

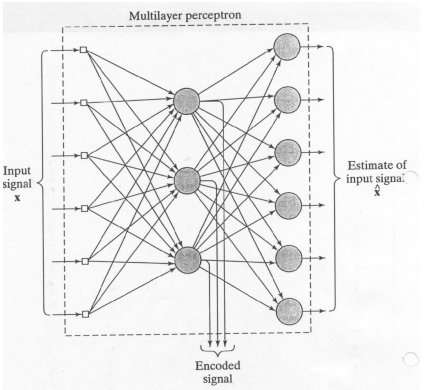
- Also point 2 holds on certain conditions.
- However, back-propagation learning is not biologically plausible (point 1) for several reasons given in Haykin's book, p. 227.
- Anyway, BP learning is important from an engineering point of view.

Feature Detection

- The hidden neurons of a multilayer perceptron trained by BP act as feature detectors (Section 4.9).
- This property can be exploited by using a MLP network as a *replicator or identity map*.



Replicator network with a single hidden layer used as an encoder.

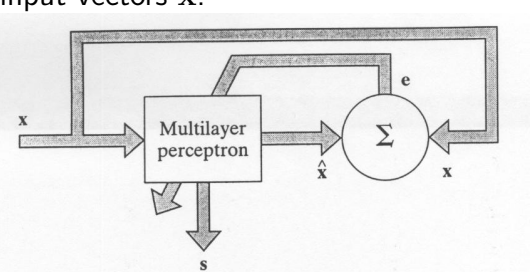


- Structural constraints:
  - The input and output layers have the same size, m.
  - The size of the hidden layer, M, is smaller than m.
  - The network is fully connected.

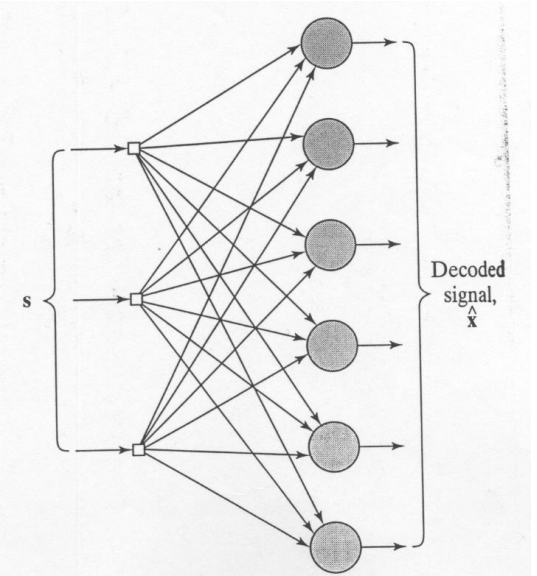
- The desired response is the same as the input vector $\mathbf{x}$.

- The actual output $\hat{\mathbf{x}}$ is the estimate of $\mathbf{x}$.

- The network is trained otherwise normally by using as an error vector $\mathbf{e} = \mathbf{x} - \hat{\mathbf{x}}$.
- Actually this is a form of *unsupervised learning* (there is no teacher).
- The replicator network performs *data compression* in its hidden layer.
- After learning, the network provides a coding/decoding system for the input vectors $\mathbf{x}$.



- Block diagram for the supervised training of the replicator network



- Decoder part of the replicator network

Function Approximation

- A multilayer perceptron trained with the back-propagation algorithm is a nested sigmoidal scheme.
- Its output vector $\mathbf{y}$ can be written in the form

$$\mathbf{y} = \mathbf{F}(\mathbf{x}, \mathbf{W}) = \mathbf{f}_L(\mathbf{W}_L \mathbf{f}_{L-1}(\mathbf{W}_{L-1} \mathbf{f}_{L-2}(\ldots \mathbf{f}_1(\mathbf{W}_1 \mathbf{x}))))$$

- $\mathbf{W}_1, \ldots, \mathbf{W}_L$ are the weight matrices of the L layers of the network, including bias terms.
- $\mathbf{f}_i(.)$ are vector-valued functions; their each component is the common sigmoid activation function $\varphi(.)$.
- $\mathbf{W}$ denotes the set of all the weight matrices.
- The dimensions of the weight matrices $\mathbf{W}_i$ and vectors $\mathbf{f}_i$ are generally different in different layers $i = 1, \ldots, L$.
- However, they must fit each other.

- The mapping $\mathbf{F}(\mathbf{x}, \mathbf{W})$ is an *universal approximator.*
- Multilayer perceptrons can approximate not only smooth, continuously differentiable functions, but also piecewise differentiable functions.

Computational Efficiency

- The *computational complexity* of an algorithm is usually measured by counting the number of multiplications, additions and storage required per iteration.
- The back-propagation algorithm is *computationally efficient*.
- This means that its computational complexity is polynomial as a function of adjustable parameters.
- If a MLP network contains a total of W weights, the computational complexity of the BP algorithm is *linear* in W.

Sensitivity Analysis

- The sensitivity analysis of the input-output mapping provided by BP can be carried out efficiently.
- A more detailed discussion of this property is skipped in our course.

Robustness

- The back-propagation algorithm is locally robust on certain conditions.
- This means that disturbances with small energy can only give rise to small estimation errors.

Convergence

- The back-propagation algorithm uses an instantaneous estimate of the gradient of the error surface.
- The direction of the instantaneous gradient fluctuates from iteration to iteration.
- Such stochastic approximation algorithms converge slowly.
- Some fundamental causes of the slow convergence:
    1. If the error surface is fairly flat along a weight dimension, many iterations may be required to reduce the error significantly.
    2. The direction of the negative instantaneous gradient vector may point away from the minimum.
       - This leads to a correction in a wrong direction for that iteration.
- The slow convergence of the back-propagation algorithm may make it computationally excruciating.

- Basic reason: large-scale neural network training problems are inherently very difficult and ill-conditioned.
- No supervised learning strategy is alone feasible.
- Suitable preprocessing may be necessary in practice.
- For example Principal Component Analysis or whitening.
- See Haykin's Chapter 8, and a problem in exercises.

Local Minima

- The error surface has *local minima* in addition to the global minimum.
- It is clearly undesirable if the learning process terminates at a local minimum.
  - Especially if this is located far above the global minimum.
- Basic reason for the existence of local minima: nonlinearities.
- If linear activation functions were used in BP, no local minima exist, but the network can then learn only linear mappings.

Scaling

- *Scaling problem:* How well the network behaves as the computational task increases in size and complexity?
- One can typically consider:
  - The time required for training.
  - The best attainable generalization performance.
- There exists many possible ways to measure the complexity or size of a computational task.
- Most useful measure: predicate order.
- Predicate is a binary function $\psi(X)$ having only two values 0 and 1, or FALSE and TRUE.

- An empirical study: how well a MLP network trained with back-propagation learns the *parity function*

$$\psi(X) = 1, \text{ if } \mid X \mid \text{ is an odd number}$$

$$\psi(X) = 0, \text{ if } \mid X \mid \text{ is an even number}$$

- The order of the parity function is equal to the number of inputs.
- It turned out that the time BP required to learn the parity function scales exponentially with the number of inputs.
- An effective method of alleviating the scaling problem:
  - Incorporate prior knowledge into the design of the network.