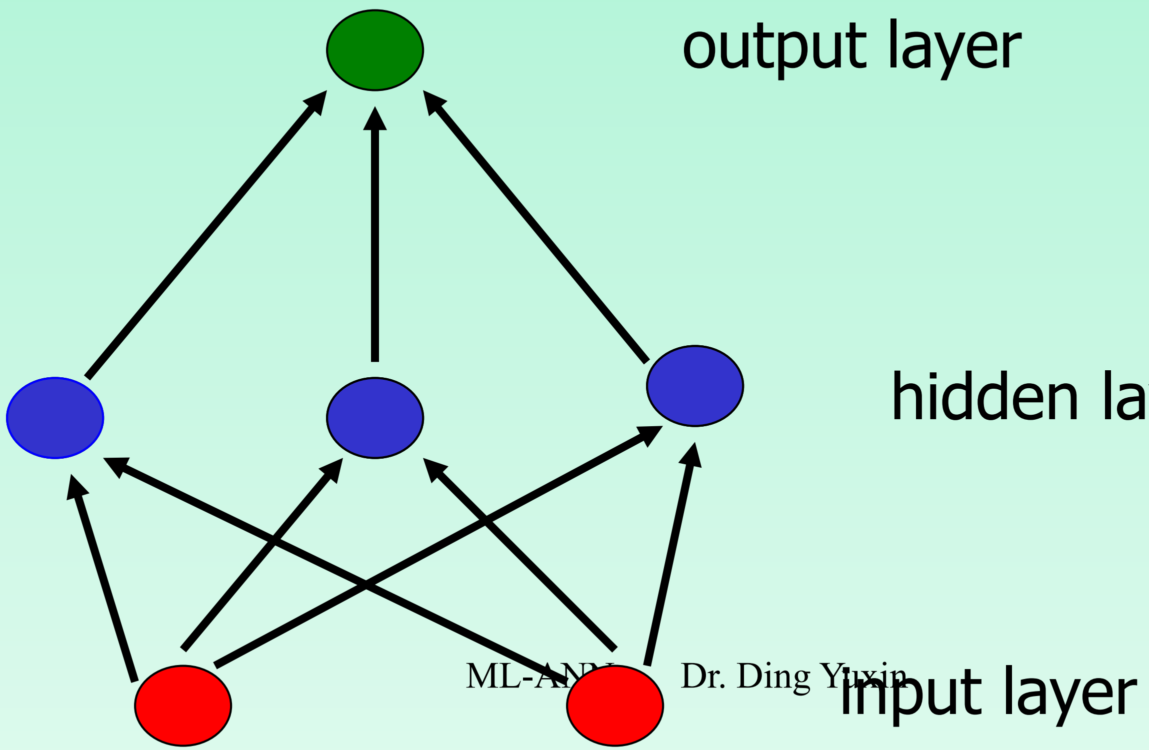
Machine Learning

Chapter 4 Artificial Neural Network （ANN）

4.5 Multilayer Networks And The Backpropagation Algorithm

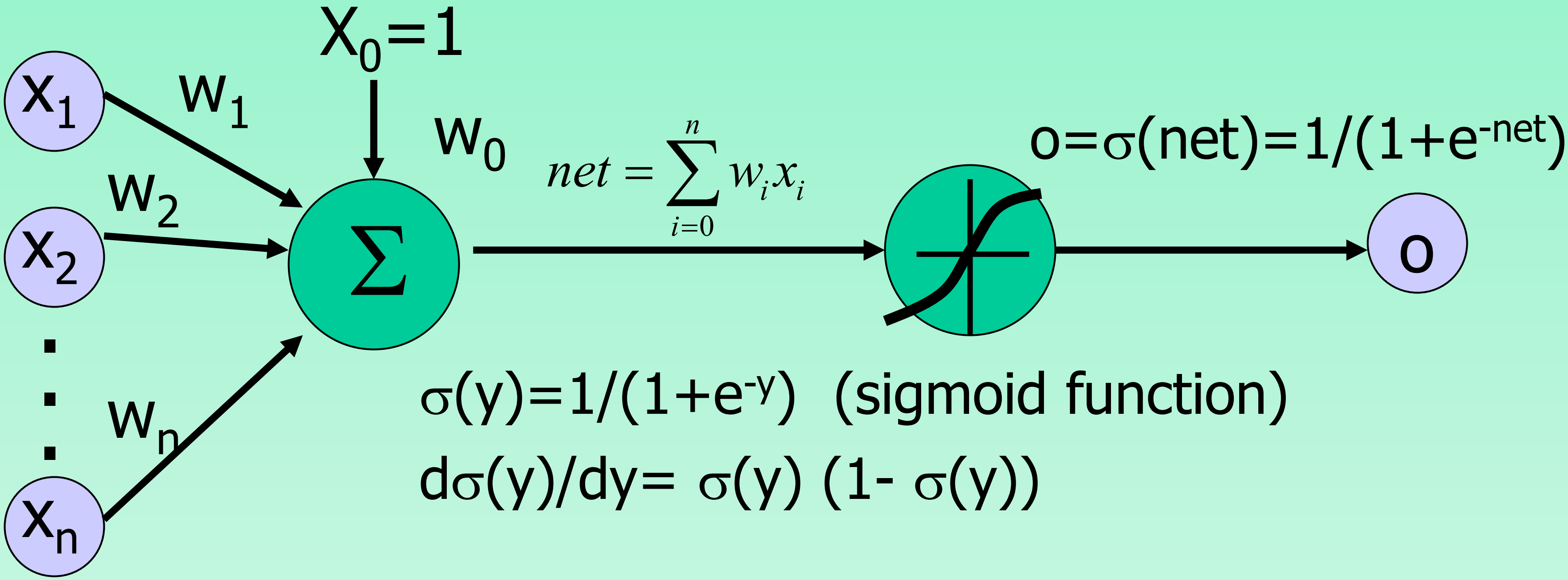
- Multilayer networks are capable of expressing a rich variety of nonlinear decision surface



4.5.1 A Differentiable Threshold Unit

- What type of unit shall is required for constructing multilayer networks?
  - represent highly nonlinear functions
  - suitable for gradient descent (differentiable)
- So the unit (neuron) should be
  - whose output is a nonlinear function of its inputs
  - Whose output is also a differentiable of its inputs

Sigmoid Unit



$\sigma(y)=1/(1+e^{-y})$ (sigmoid function)

$d\sigma(y)/dy= \sigma(y)\,(1- \sigma(y))$

Derive gradient decent rules for training one sigmoid function

$$E(\vec{w}) = \frac{1}{2}\sum_{d\in D}(t_d - o_d)^2$$

$$\partial E/\partial w_i = -\Sigma_d(t_d-o_d)\, o_d\,(1-o_d)\, x_i$$

- Characteristics of Sigmoid unit
  - a smoothed, differentiable threshold function
  - Sigmoid function (logistic function, Squashing function)
  - Its output ranges between 0 and 1
  - Increasing monotonically
  - Its derivative is easily expressed in terms of its output

- Other differentiable functions
  - Increase the steepness of the sigmoid unit(e^{-ky})
  - tanh（双曲正切） $\frac{e^x - e^{-x}}{e^x + e^{-x}}$

4.5.2 The Backpropagation Algorithm

- Learns the weights for a multilayer network
  - Employs gradient descent minimize the squared error between the network output values and the target values
  - Redefining E to sum the errors over all of the network output units

$$E(\vec{w}) = \frac{1}{2} \sum_{d \in D} \sum_{k \in outputs} (t_{kd} - o_{kd})^2$$

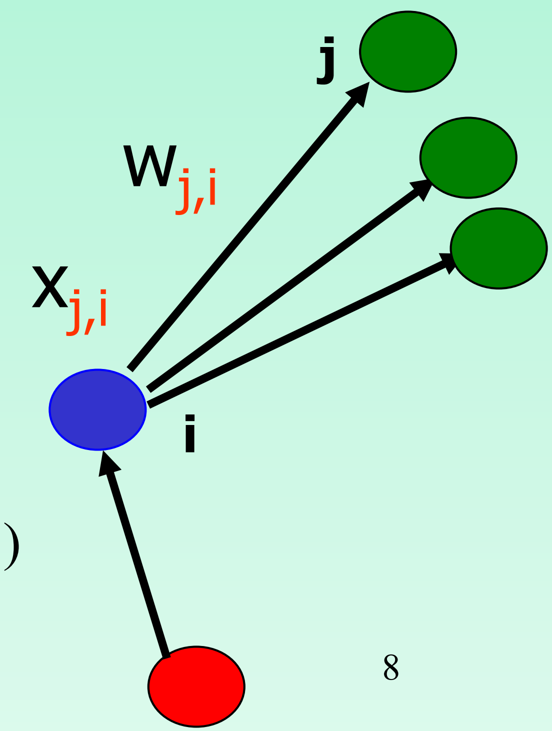
- Difficulties in BP Learning
  – Search a large hypothesis space
  – Error surface can have multiple local minima

- However, in practice BP has been found to produce excellent results

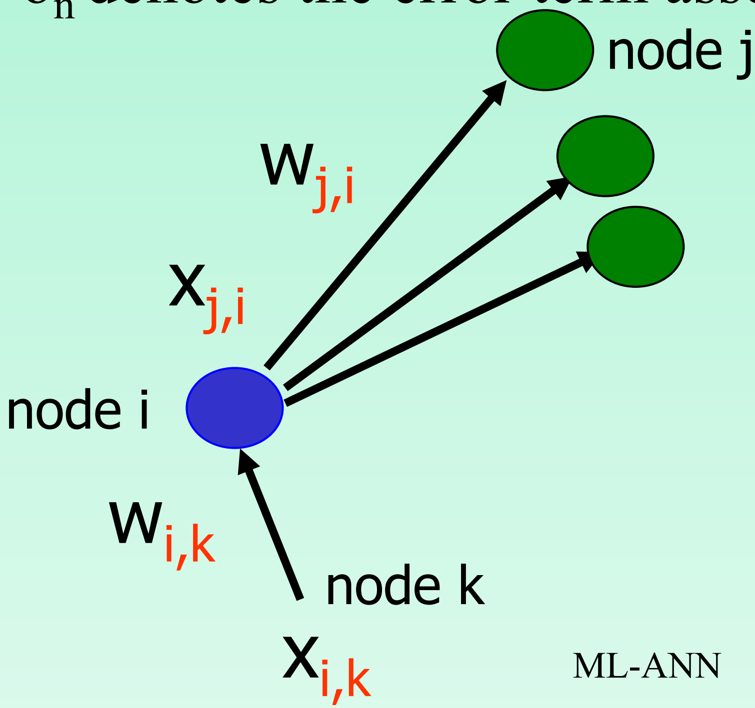
Backpropagation Algorithm

- Initialize each w_i to some small random value
- Until the termination condition is met, Do
  - For each training example $<(x_1,\ldots x_n),t>$ Do
    - Input the instance $(x_1,\ldots,x_n)$ to the network and compute the network outputs o_k
    - For each output unit k
      - $\delta_k=o_k(1-o_k)(t_k-o_k)$
    - For each hidden unit h
      - $\delta_h=o_h(1-o_h)\ \Sigma_{k \text{ in outputs}}\ w_{k,h}\ \delta_k$
    - For each network weight $w_{j,i}$ Do
      - $w_{j,i,}=w_{j,i}+\Delta w_{j,i}$ where
      - $\Delta w_{j,i}= \eta\ \delta_j\ x_{j,i}$

      (Note:$x_{j,i}$： output of unit i, corresponding to $w_{j,i}$)



- Notation:
  - An index is assigned to each node in the network, where a “node” is either an input to the network or the output of some unit in the network
  - x_{ji} denotes the input from node i to unit j
  - w_{ji} denotes the corresponding weight
  - δ_n denotes the error term associated with unit n



- Remarks on the algorithm
  - Network structure
    - feedforward networks containing two layers of sigmoid units
    - units at each layer connected to all units from the preceding layer
  - Learning Method
    - The incremental, or stochastic gradient descent version
    - Weight update ($\Delta w_{j,i} = \eta\ \delta_j\ x_{j,i}$, delta rule: $\Delta w_i = \eta(t - o)x_i$)
      - The learning rate η
      - The input value x_{ji} on $w_{j,i}$
      - The error item of the unit j ($\delta_j = -\frac{\partial E}{\partial net_j}$)

- Understanding δ_j ($\Delta w_{j,i} = \eta\ \delta_j\ x_{j,i}$, $\Delta w_i = \eta(t - o)x_i$)
  - δ_k for output unit k
    - δ_k is $(t_k - o_k)$, multiplied by the factor $o_k(1-o_k)$, (the derivative of the sigmoid function)

      $\delta_k = o_k(1-o_k)(t_k-o_k)$
  - δ_h for hidden unit h
    - no direct target values to calculate the error of hidden units' values.
    - The error is calculate by summing the error terms δ_k for each output unit influenced by h, weighting δ_k by w_{kh}

      $\delta_h = o_h(1-o_h) \sum_{k \in outputs} w_{k,h}\ \delta_k$

- The true gradient descent algorithm of BP
  - summing the $\delta_j x_{ji}$ values over all training examples before altering weight values

- Termination conditions
  - A fixed number of iterations through the loop
  - The error falls below some threshold
  - The error on a separate validation set of examples meets some criterion
  - Termination criterion is important: error, overfitting

4.5.2.1 Adding Momentum

- An variation developed for BP
  - making the weight update on the nth iteration depend partially on the update that occurred during the (n-1)th iteration

    $\Delta w_{ji}(n)=\eta\delta_j x_{ji}+\alpha\Delta w_{ji}(n-1)$

    $0<=\alpha<1$, $\Delta w_{ji}(n-1)$ is called momentum
  - Understanding
    - Momentum tends to keep the ball rolling down in the same direction.
    - Effect 1: keeping the ball rolling through local minima, or along flat regions
    - Effect 2: gradually increasing the step size of the search in regions where the gradient is unchanging, thereby speeding convergence

4.5.2.2 Learning In Arbitrary Acyclic Networks

- Learning algorithm for feedforward networks of arbitrary depth.
  - The δ_r value for a unit r in layer m is computed from the δ_s values at the next deeper layer m+1

$$\delta_r = o_r(1-o_r) \sum_{s \in layer\ m+1} w_{sr}\delta_s$$

  - This algorithm is generalized to any directed acyclic graph. For internal units r:

$$\delta_r = o_r(1-o_r) \sum_{s \in Downstream(r)} w_{sr}\delta_s$$

Downstream(r) is the set of units immediately downstream from unit r in the network: all units whose inputs include the output of unit r.

4.5.3 Derivation of the Backpropagation Rule

- The stochastic gradient descent algorithm of BP
  - for each training example d, weights are updated by descending the gradient of the error E_d with respect to this single example

$$\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial w_{ji}} \qquad E_d(\vec{w}) = \frac{1}{2} \sum_{k \in outputs} (t_k - o_k)^2$$

Notation Specification

- x_{ji}， the ith input to unit j
- w_{ji}， the weight associated with the ith input to unit j
- net_j， the weighted sum of inputs for unit j
- o_j， the output computed by unit j
- t_j， the target output for unit j
- σ， the sigmoid function
- outputs， the set of units in the final layer of the network
- Downstream(j)， the set of units whose immediate inputs include the output of unit j

$$\frac{\partial E_d}{\partial w_{ji}} = \frac{\partial E_d}{\partial net_j}\frac{\partial net_j}{\partial w_{ji}} = \frac{\partial E_d}{\partial net_j} x_{ji}$$, for $\frac{\partial E_d}{\partial net_j}$, two case is considered

- Output units

$$\frac{\partial E_d}{\partial net_j} = \frac{\partial E_d}{\partial o_j}\frac{\partial o_j}{\partial net_j}$$

$$\frac{\partial E_d}{\partial o_j} = \frac{\partial}{\partial o_j}\frac{1}{2}\sum_{k \in outputs}(t_k - o_k)^2$$

$$= \frac{\partial}{\partial o_j}\frac{1}{2}(t_j - o_j)^2$$

$$= \frac{1}{2}2(t_j - o_j)\frac{\partial (t_j - o_j)}{\partial o_j}$$

$$= -(t_j - o_j)$$

$$\frac{\partial o_j}{\partial net_j} = \frac{\partial \sigma(net_j)}{\partial net_j} = o_j(1 - o_j)$$

$$\frac{\partial E_d}{\partial net_j} = -(t_j - o_j)o_j(1 - o_j)$$

$$= -\delta_j$$

$$\Delta w_{ji} = -\eta\frac{\partial E_d}{\partial w_{ji}} = \eta(t_j - o_j)o_j(1 - o_j)x_{ji}$$

$$= \eta\delta_j x_{ji}$$

- Hidden Unit

$$\frac{\partial E_d}{\partial net_j} = \sum_{k \in Downstream(j)} \frac{\partial E_d}{\partial net_k} \frac{\partial net_k}{\partial net_j}$$

$$\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial w_{ji}} = \eta \delta_j x_{ji}$$

$$= \sum_{k \in Downstream(j)} -\delta_k \frac{\partial net_k}{\partial net_j}$$

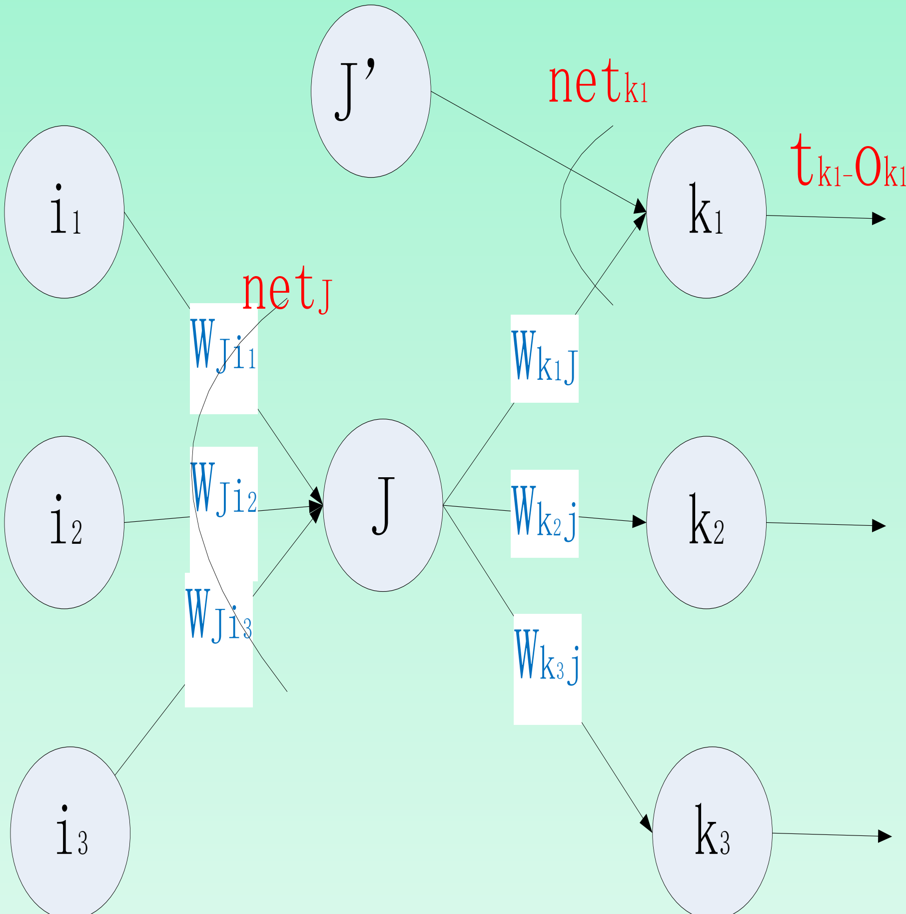
$$= \sum_{k \in Downstream(j)} -\delta_k \frac{\partial net_k}{\partial o_j} \frac{\partial o_j}{\partial net_j}$$

$$= \sum_{k \in Downstream(j)} -\delta_k w_{kj} \frac{\partial o_j}{\partial net_j}$$

$$= \sum_{k \in Downstream(j)} -\delta_k w_{kj} o_j (1 - o_j)$$

$$= -o_j (1 - o_j) \sum_{k \in Downstream(j)} \delta_k w_{kj}$$

$$= -\delta_j$$



4.6.1 Converge and Local Minima

- BP is only guaranteed to converged toward some local minimum in E
  - the error surface may contain many local minima
  - gradient descent can become trapped in any of those

- BP is still a highly effective function approximation method in practice
  - Large amount of weights provide “escape routes” for escaping the local minimum
  - The manner network weights evolve
    - in the early stage, network weights=0, the network represents a smooth function (approximating a linear function)
    - With the weights growing, they can represent highly nonlinear functions (have more local minima in the weight space)
    - At this point, weights have already moved close enough to the global minimum that even the local minima are acceptable

- Heuristics to the problem of local minima:
  - Add a momentum term
  - Use stochastic gradient rather than true gradient descent
  - Train multiple networks, selected the best (performance over a validation set).

4.6.2 Representational Power of Feedforward Networks

- Bool function: using two layers of units.
  - one scheme for representing an arbitrary bool function：
    - One input vector only activates one hidden unit, implements the output unit as an OR gate

- Bounded Continuous Functions:
  - two layers of units: sigmoid units at the hidden layer + linear units at the output layer
  - arbitrarily small error
- Arbitrary functions:
  - three layers of units (two sigmoid layers + one linear output layer)
- Note: the network weight vectors reachable by gradient descent from the initial weight values may not include all possible weight vector

4.6.3 Hypothesis Space Search and Inductive Bias

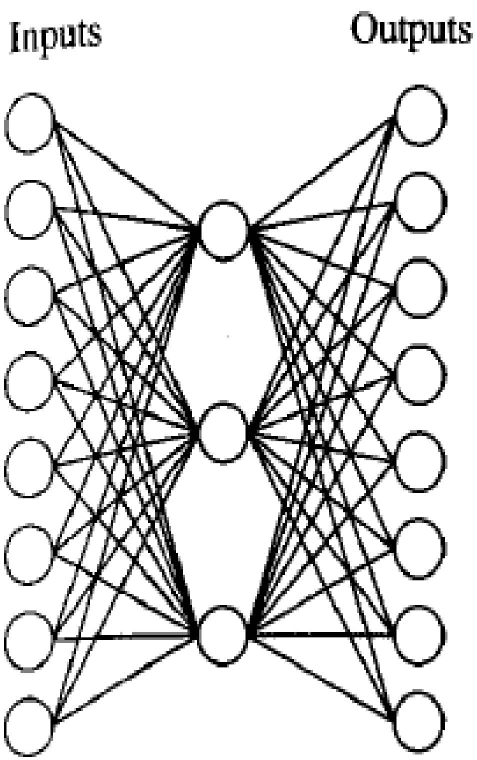
- Hypothesis Space of BP
  - n-dimensional Euclidean and continuous space
  - E is differentiable
    - So the gradient descent algorithm can organize the space search for the best h

- Inductive Bias of BP
  - depends on
    - 1. the gradient descent search
    - 2. the way in which the weight space spans the space of representable functions.

  - roughly characterize as smooth interpolation between data points

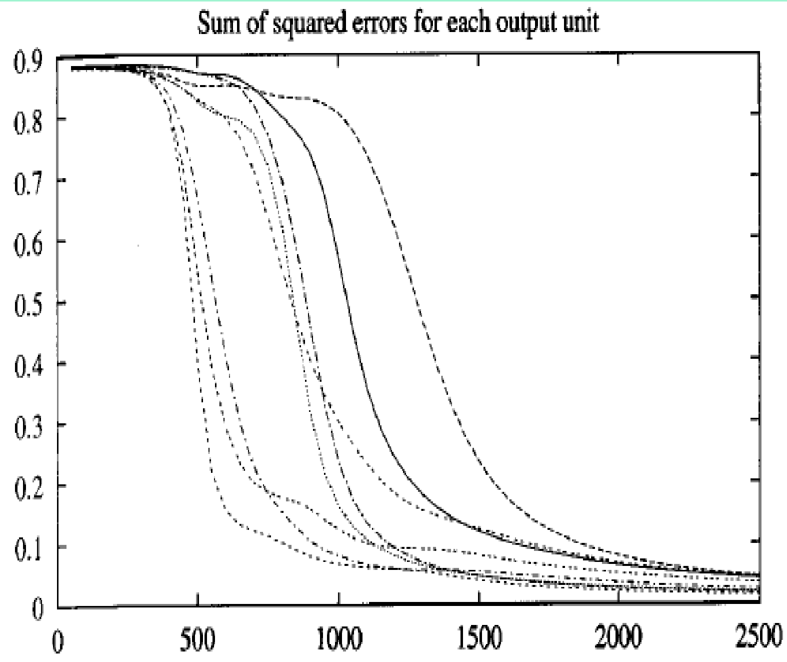
4.6.4 Hidden Layer Representations

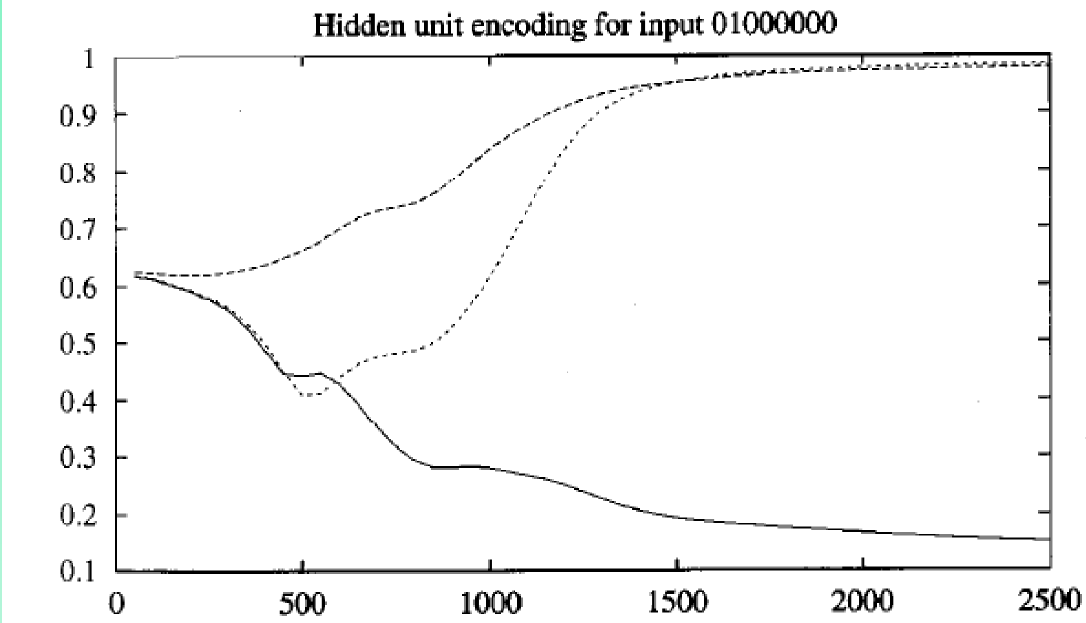
- Useful intermediate representations at the hidden unit layers
  - new hidden layer features
    - not explicit in the input
    - properties of the input instances that are most relevant to learning the target function
  - When more layers of units are used, more complex features can be invented
  - It is a key feature of ANN learning.
    - allows the learner to invented features not explicitly introduced by the human designer

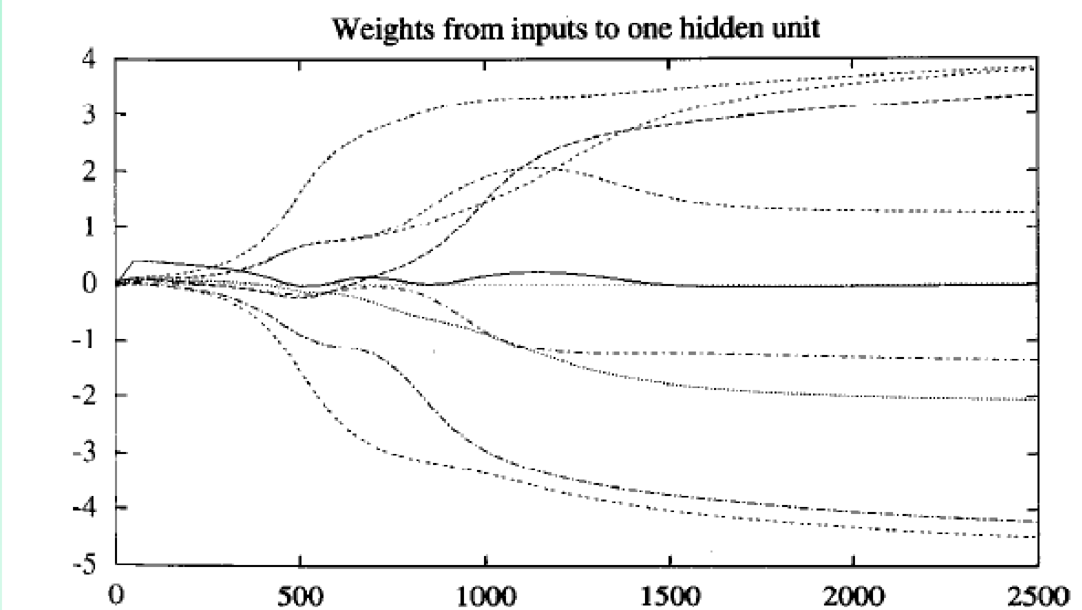


Input		Hidden Values				Output
10000000	→	.89	.04	.08	→	10000000
01000000	→	.15	.99	.99	→	01000000
00100000	→	.01	.97	.27	→	00100000
00010000	→	.99	.97	.71	→	00010000
00001000	→	.03	.05	.02	→	00001000
00000100	→	.01	.11	.88	→	00000100
00000010	→	.80	.01	.98	→	00000010
00000001	→	.60	.94	.01	→	00000001

FIGURE 4.7

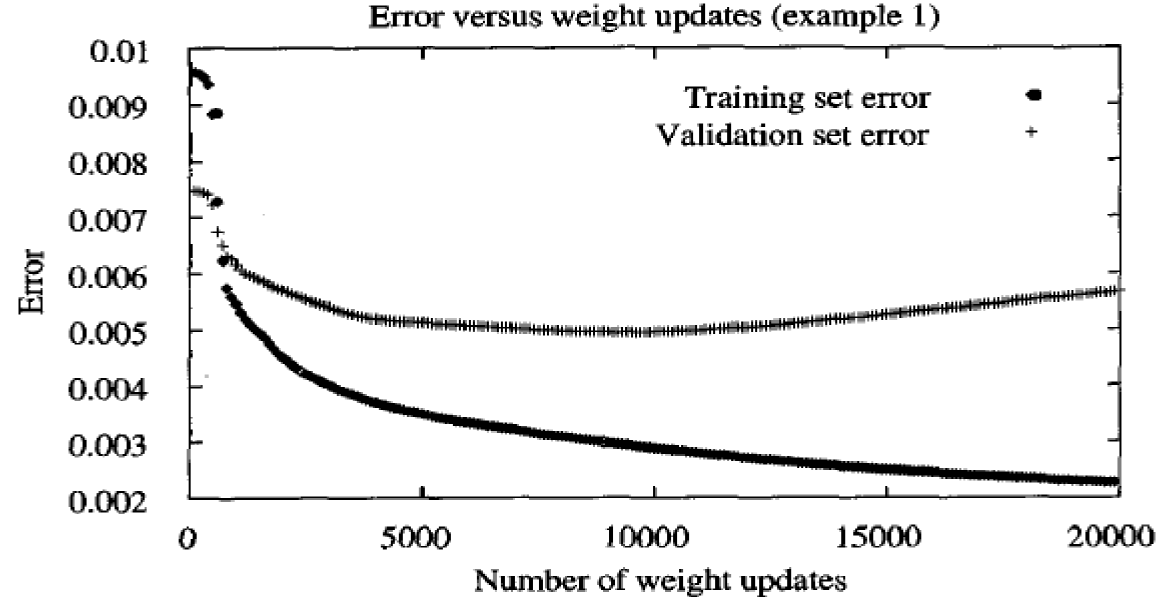

Sum of squared errors for each output unit
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
500
1000
1500
2000
2500

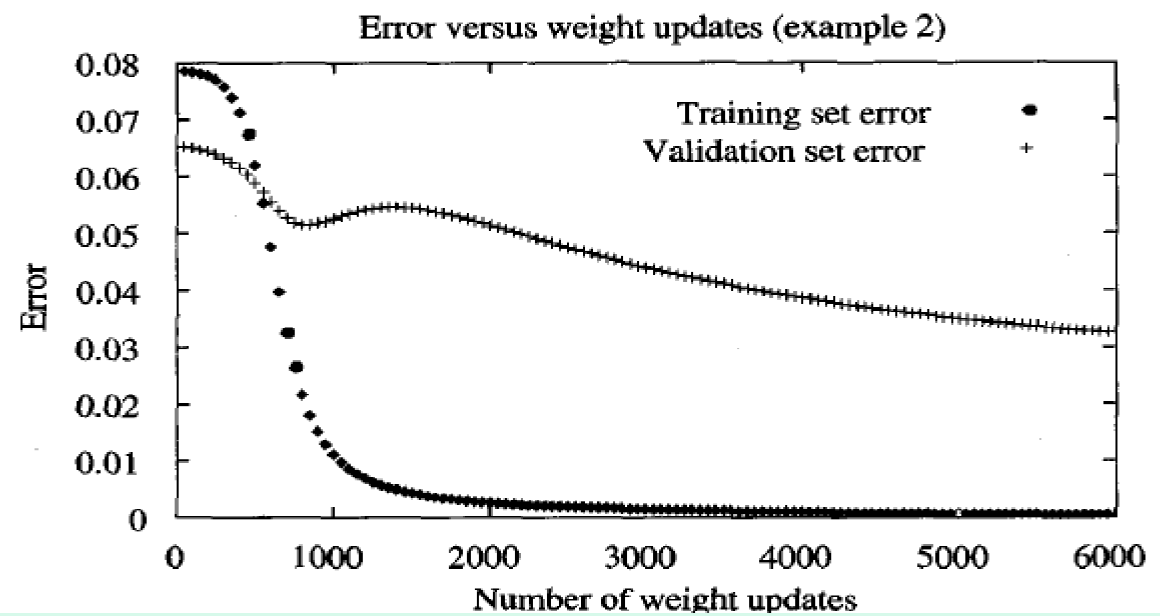

Hidden unit encoding for input 01000000
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
500
1000
1500
2000
2500


Weights from inputs to one hidden unit
4
3
2
1
0
-1
-2
-3
-4
-5
0
500
1000
1500
2000
2500

4.6.5 Generalization, Overfitting, and Stopping Criterion

- Condition for terminating the weight update loop
  - One choice: stop when the error E falls below a threshold
    - Problem: overfitting
- Generalization accuracy
  - the accuracy with which it fits examples beyond the training data
- Overfitting(Fig 4-9, next page)
  - the weights are being tuned to fit idiosyncrasies(个体特有的习性) of the training examples
  - The large number of weight parameters provides many degrees of freedom for fitting such idiosyncrasies


Error versus weight updates (example 1)
0.01
0.009
0.008
0.007
0.006
0.005
0.004
0.003
0.002
Error
Training set error
Validation set error
0
5000
10000
15000
20000
Number of weight updates


Error versus weight updates (example 2)
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
Error
Training set error
Validation set error
0
1000
2000
3000
4000
5000
6000
Number of weight updates

- Why does overfitting tend to occur during later iterations
  - the weights are initialized to small random value, only smooth surface are describable
  - As training proceeds, weights begin to grow, the complexity of the learned surface increases
  - Given enough weight-tuning iterations, overly complex surface is created to fit noise

- Overcome the overfitting problem
  - weight decay
    - Decrease each weight by some small factor during
      - equivalent to include a penalty term corresponding to the total magnitude of the network weights in E.
      - the motivation is to keep weights values small, to bias learning against complex decision surface(see section 4.8.1)
  - Cross-validation approach
    - drive the gradient descent search using the training set
    - monitors the error using a validation set
    - Need more training samples

– k-fold cross validation

- Used to avoid overfitting for small training sets.
- The m examples are partitioned into k disjoint subsets, each of size m/k.
- run k times, each time using a subset as the validation set and combing the other subsets for training set
- Cross-validation approach for BP is used to determine the number of iterations i that yield the best performance, calculate the mean $\bar{i}$
- A final run of BP is performed $\bar{i}$ iterations

4.7 An Illustrative Example: Face Recognition

- Training data
  - Images of 20 different people
  - 32 images per person
    - Different expression
      - Happy, sad, angry, neutral
    - Face direction
      - Left, right, straight ahead, up
    - Others
      - wearing sunglasses, background behind the person, clothing
  - In total 624 greyscale images
    - resolution of 120x128
    - pixel value： 0 --- 255
- task： learning the direction a person is facing

4.7.2 Design Choices

- Input encoding
  - Representation 1: extract edges, regions of uniform intensity, or other feature.
    - problem： lead to a variable number of features, whereas the ANN has a fixed number of inputs
  - Representation 2: one network input per pixel
    - Fixed number of input
    - the pixel intensity values were linearly scaled to range from 0 to 1(so that network input would have values in the same interval as the hidden unit and output unit activations)

4.7.2 Design Choices

- Output encoding
  - using a single output unit
  - Using four distinct output units
    - the highest-valued output is taken as the network prediction, (called as a 1-of-n output encoding)
- Why choosing the 1-of-n output encoding
  - Provides more degree of freedom for representing the target function
  - Provide a confidence measure

- Target value for 4 output units
  - One obvious choice <1,0,0,0>...
  - We use values of 0.1 and 0.9, <0.9,0.1,0.1,0.1>...
  - Why avoiding use 0 and 1
    - Sigmoid unit can not produce such output values
    - Avoid forcing the weights to grow without bound
    - Values of 0.1 and 0.9 are achievable using a sigmoid unit with finite weights

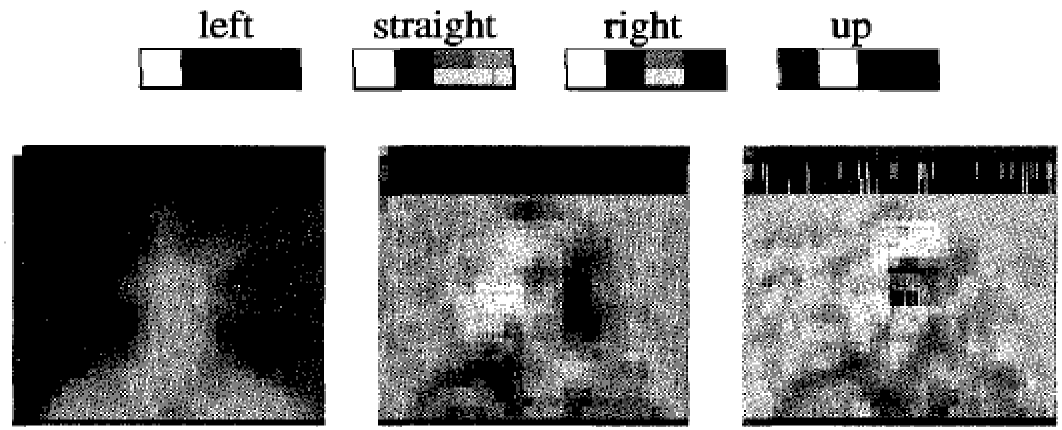
- Network graph structure
  - How many units and how to interconnect them
    - a layered network , feedforward connections
      - two layers of sigmoid units
    - Number of hidden units
      - 3, test accuracy of 90%, 5 minutes on Sun SParc5 , 260 training images
      - 30, one to two percent higher , nearly 1 hour
  - Number of hidden units
    - some minimum number of hidden units is required
    - the extra hidden units do not dramatically affect generalization accuracy
    - Overfit: increasing the number of hidden units often increase the tendency to overfit the training data

- Other learning parameters
  - Learning rate :0.3, momentum : 0.3
    - Lower values produce roughly equivalent generalization accuracy, but longer training times
    - Too high values, training fails to converge
  - Full gradient descent was used
  - Weights in the output units were initialized to small random values
  - Input unit weights were initialized to zero
  - The number of training iterations:
    - decide by a training set and a separate validation set.
  - The final network :
    - having the highest accuracy over the validation set
  - The final reported accuracy:
    - measured over a third set of test examples

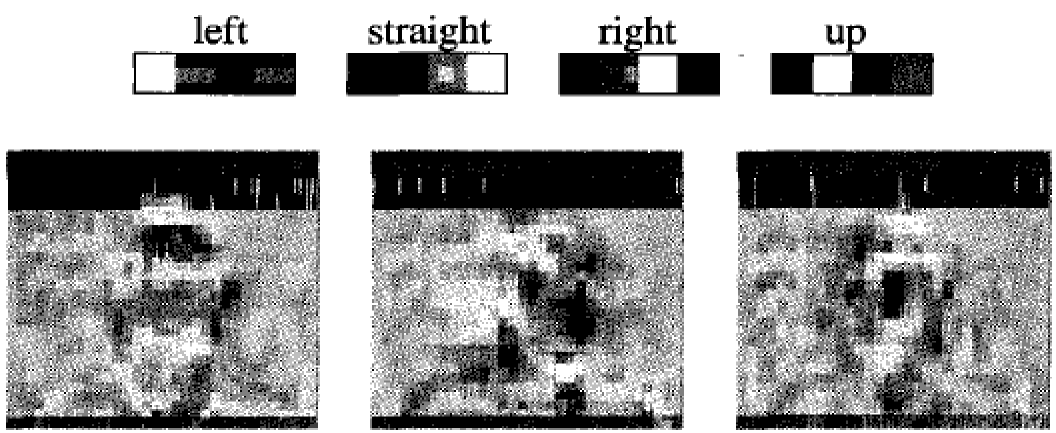
4.7.3 Learned Hidden Representations

- Output weights after 1 iteration
  - Weights from input units to hidden units after 1 iteration (see next page)
- Output weights after 100 iteration
  - Weights from input units to hidden units after 100 iteration (see next page)

30 × 32 resolution input images



Network weights after 1 iteration through each training example



Network weights after 100 iterations through each training example

4.8.1 Alternative Error Functions

- Adding a penalty term
  - seek weight vectors with small magnitudes, thereby reduce the risk of overfitting

$$E(\vec{w}) = \frac{1}{2}\sum_{d\in D}\sum_{k\in outputs}(t_{kd} - o_{kd})2 + \gamma\sum_{i,j} w_{ji}^2$$

- Adding a term for error in the slop or derivative of the target function

$$E(\vec{w}) = \frac{1}{2}\sum_{d\in D}\sum_{k\in outputs}\left[(t_{kd} - o_{kd})^2 + \mu\sum_{j\in inputs}\left(\frac{\partial t_{kd}}{\partial x_d^j} - \frac{\partial o_{kd}}{\partial x_d^j}\right)^2\right]$$

- Minimize the cross entropy of the network with respect to the target values

  – It is for learning a probabilistic function

Cross entropy：

$$-\sum_{d \in D} t_d \log o_d + (1 - t_d)\log(1 - o_d)$$

  – Chapter 6 discusses when and why the most probable h is the one that minimizes this cross entropy

4.9 summary and further reading

- learning real-valued and vector-valued functions over continuous and discrete-valued attributes
- robust to noise in the training data
- BP is the most common algorithm for ANN learning
- H considered by BP is the space of all functions that can be represented by assigning weights to the given, fixed network of interconnected units
- Feedforward networks containing 3 layers of units are able to approximate any function to arbitrary accuracy
- BP searches the H using gradient descent to iteratively reduce the error in the network to fit training examples

- Gradient descent converge to a local minimum.
- gradient descent is a potentially useful method for searching many continuously parameterized hypothesis spaces where the training error is a differentiable function
- BP can invent new features that are not explicit in the input
- Overfitting and Cross-validation

- Homework:4.2,4.3, 4.5,4.7 (p.91 or p.125)