Chapter 6

Multilayer Neural Networks

Introduction

- A number of methods introduced in chap 5 provides
  - The class of solutions that can be obtained – comprising hyperplane decision boundaries – is simply not general enough in demanding applications.
    - There are many problems for which linear discriminants are insufficient for minimum error.
- With a clever choice of nonlinear φ functions, we can obtain arbitrary decision regions;
  - In particular those leading to minimum error.
  - The central difficulty: choosing the appropriate nonlinear functions.
  - What we seek is a way to learn the nonlinearity at the same time as the linear discriminant.
    - The parameters governing the nonlinear mapping are learned at the same time as those governing the linear discriminant.

Introduction

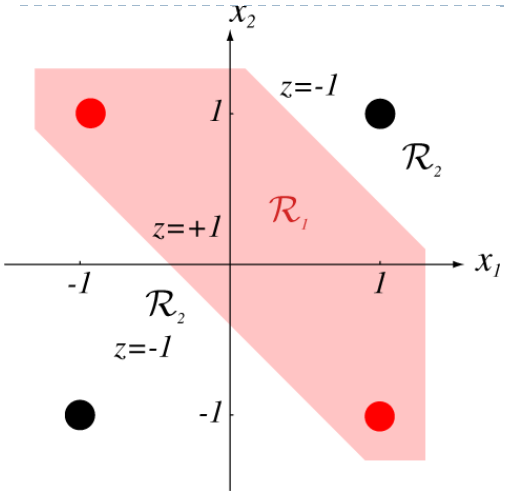
- NNs implement linear discriminant, but in a space where the inputs have been mapped nonlinearly.
- Backpropagation
  - One of the most popular methods for training multilayer networks
    - A natural extension of the LMS – generalized delta rule.
- Network architecture or topology
  - Plays an important role for neural net classification
  - #of hidden layers/units, feedback connecntions

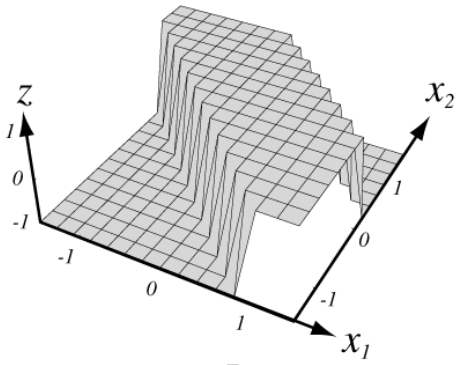
Introduction

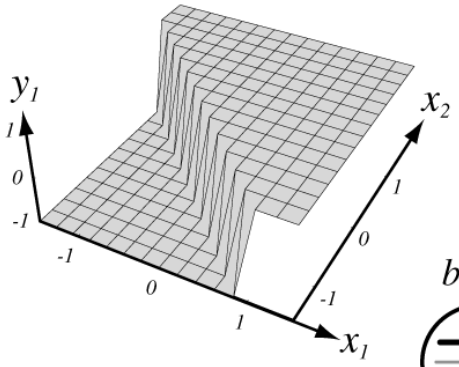
- Regularization
  - A deep problem in the use of neural network techniques.
    - Selecting or adjusting the complexity of the network.
    - Number of inputs and outputs are determined by the number of feature space and number of categories. But the number of weights or parameters in the network is not.
    - Too many free parameters: poor generalization
    - Too few parameters: the training data cannot be learned adequately.

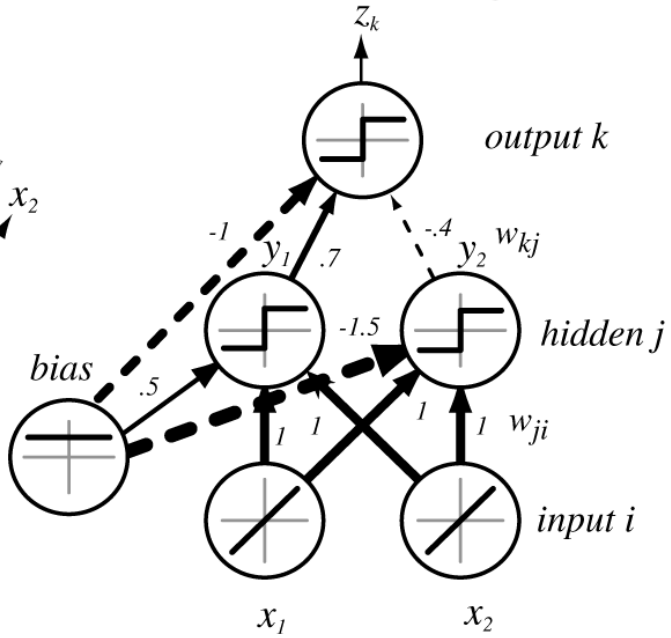
- It is crucial to remember that neural networks do not exempt designers from intimate and detailed knowledge of the data and problem domain.
  - Networks provides a powerful and speedy tool for building classifiers, and as with any tool or technique, one gains intuition and expertise through analysis and repeated experimentation over a broad range of problems.

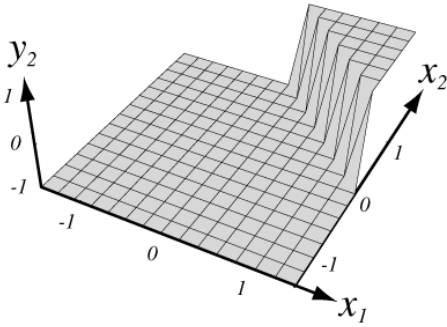
Feedforward Operation and Classification











Feedforward Operation and Classification

- Hidden layer
  - Their activations are not directly *seen* by the external environment.
- Bias unit
  - A single unit that is connected to each unit other than the input units.
- Neurons
  - The function of units is loosely based on properties of biological neurons.
- Net activation
  - Each hidden unit computes the weighted sum of its inputs to form its scalar net activation: inner product of the input with the weights at the hidden unit.

$$net_j = \sum_{i=1}^{d} x_i w_{ji} + w_{j0} = \sum_{i=0}^{d} x_i w_{ji} = \mathbf{w}_j^t \mathbf{x}$$

Feedforward Operation and Classification

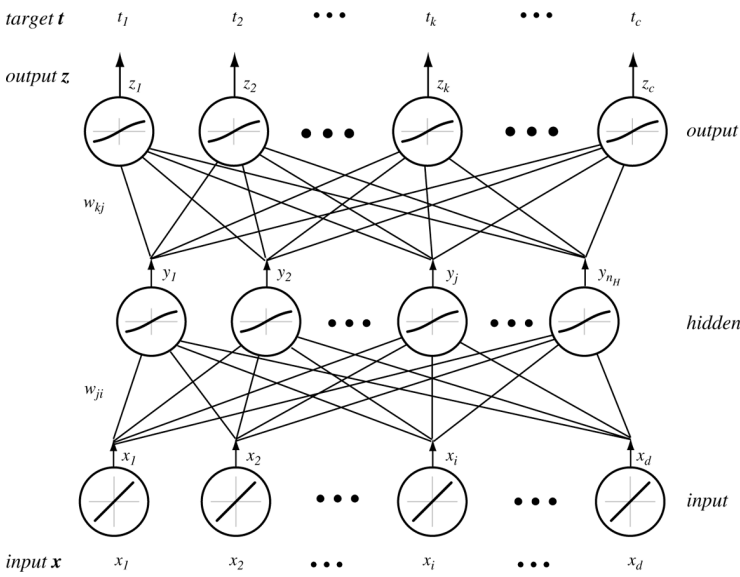
- Synapse
  - In analogy with neurobiology, such weights or connections are called synapses.
  - The values of the connections are called synaptic weight.
- Activation function
  - Nonlinear activation function $f(net)$

$$y_j = f(net_j)$$

- Output units

$$net_k = \sum_{j=1}^{n_H} y_j w_{kj} + w_{k0} = \sum_{j=0}^{n_H} y_j w_{kj} = \mathbf{w}_k^t \mathbf{y}$$

$$z_k = f(net_k)$$



Feedforward Operation and Classification

- General feedforward Operation
  - Nonlinear multilayer networks have expressive power than similar ones that otherwise lack hidden units.
    - Given sufficient number of hidden units of a general type any function can be so represented.

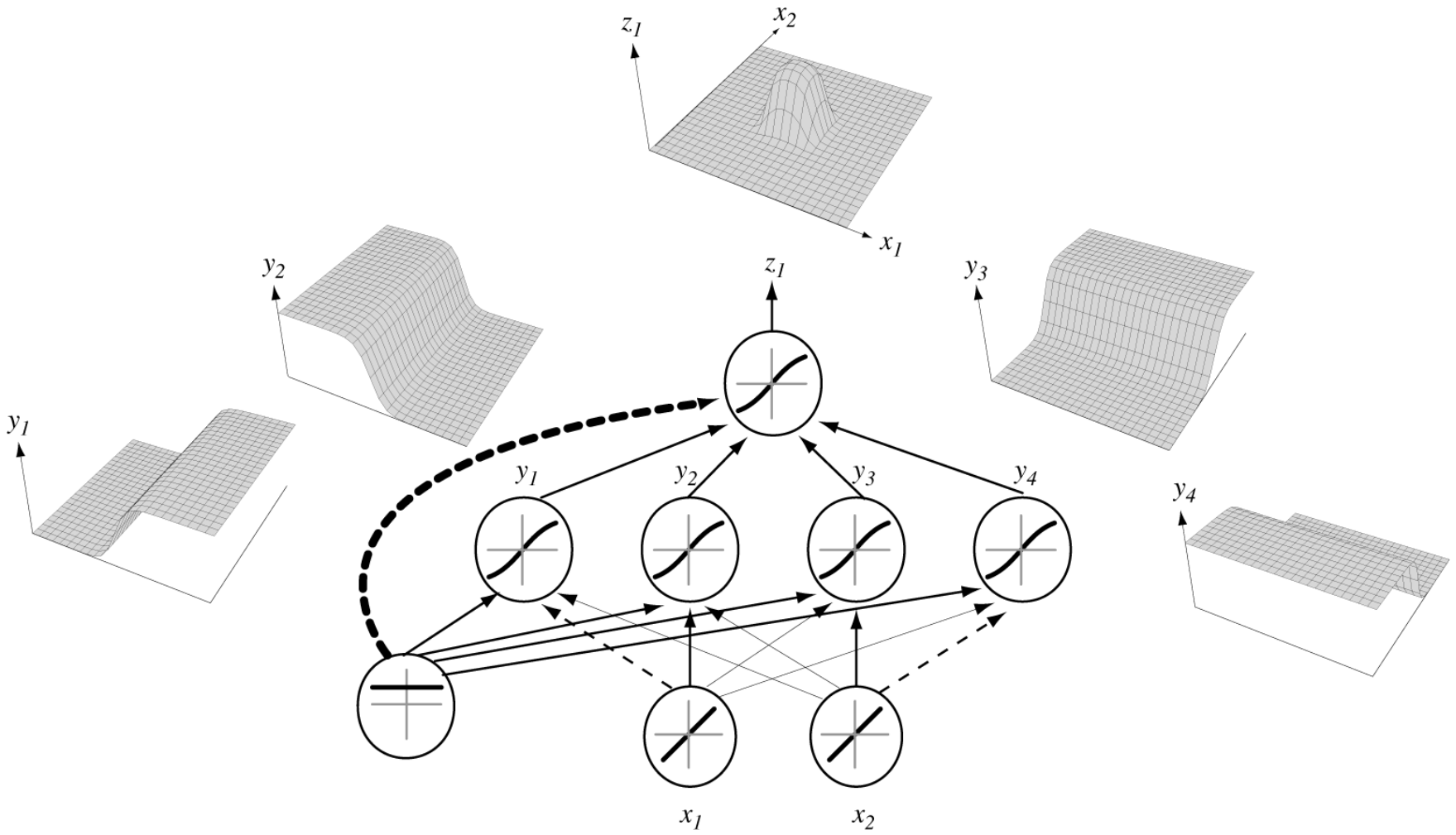
$$g_k(\mathbf{x}) \equiv z_k = f\left(\sum_{j=1}^{n_H} w_{kj} f\left(\sum_{i=1}^{d} w_{ji} x_i + w_{j0}\right) + w_{k0}\right)$$

  - The activation functions are required to be continuous and differentiable.

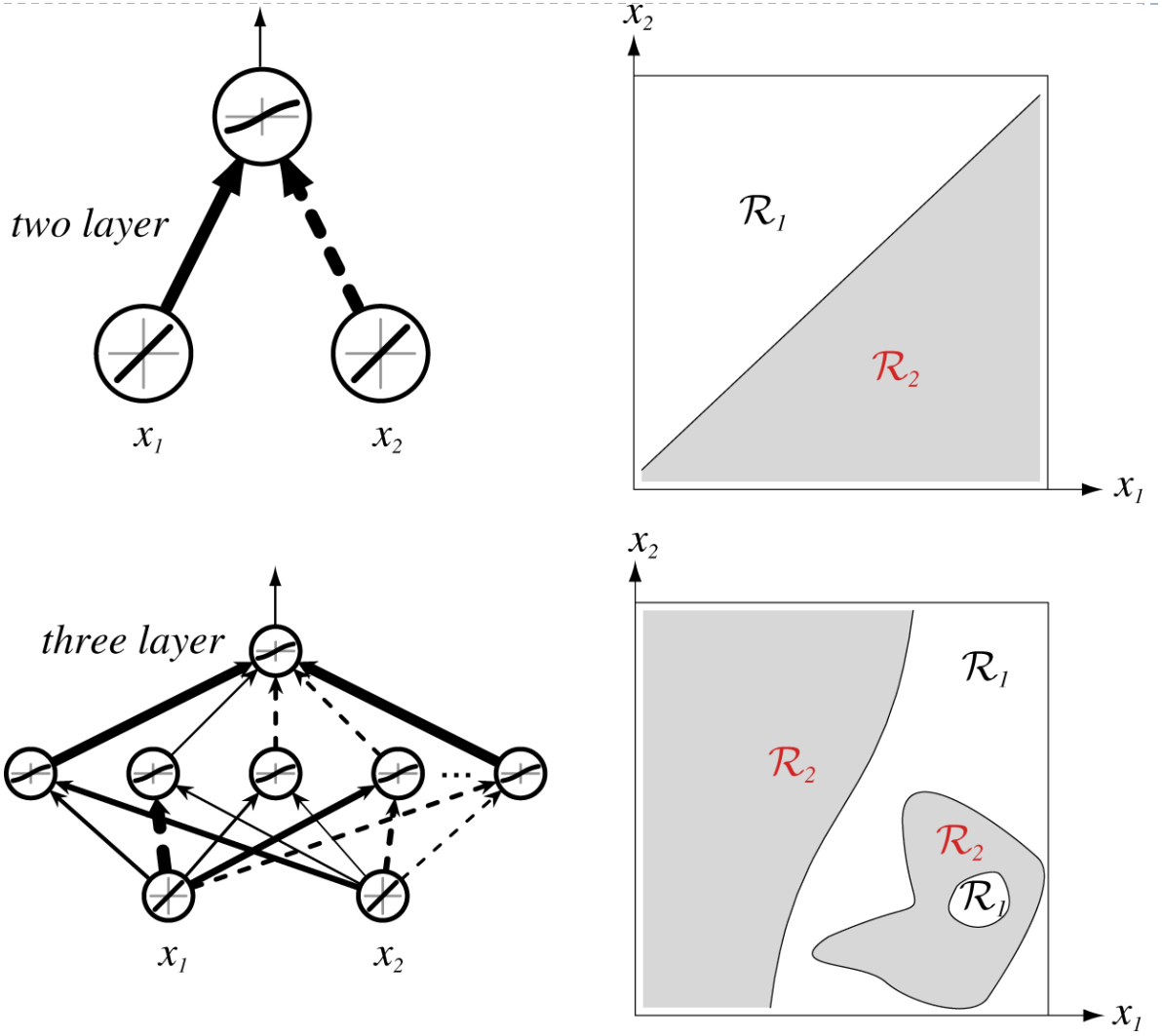
Feedforward Operation and Classification

- Expressive Power of Multilayer Networks
  - It is natural to ask if every decision can be implemented by such a three-layer network. The answer is "yes".
    - Any continuous function from input to output can be implemented in a three-layer net, given sufficient number of hidden units n_H, proper nonlinearities, and weights.
    - A more intuitive proof of the universal expressive power – Fourier theorem: any continuous function can be approximated arbitrarily closely by a possibly infinite sum of harmonic functions.

Feedforward Operation and Classification



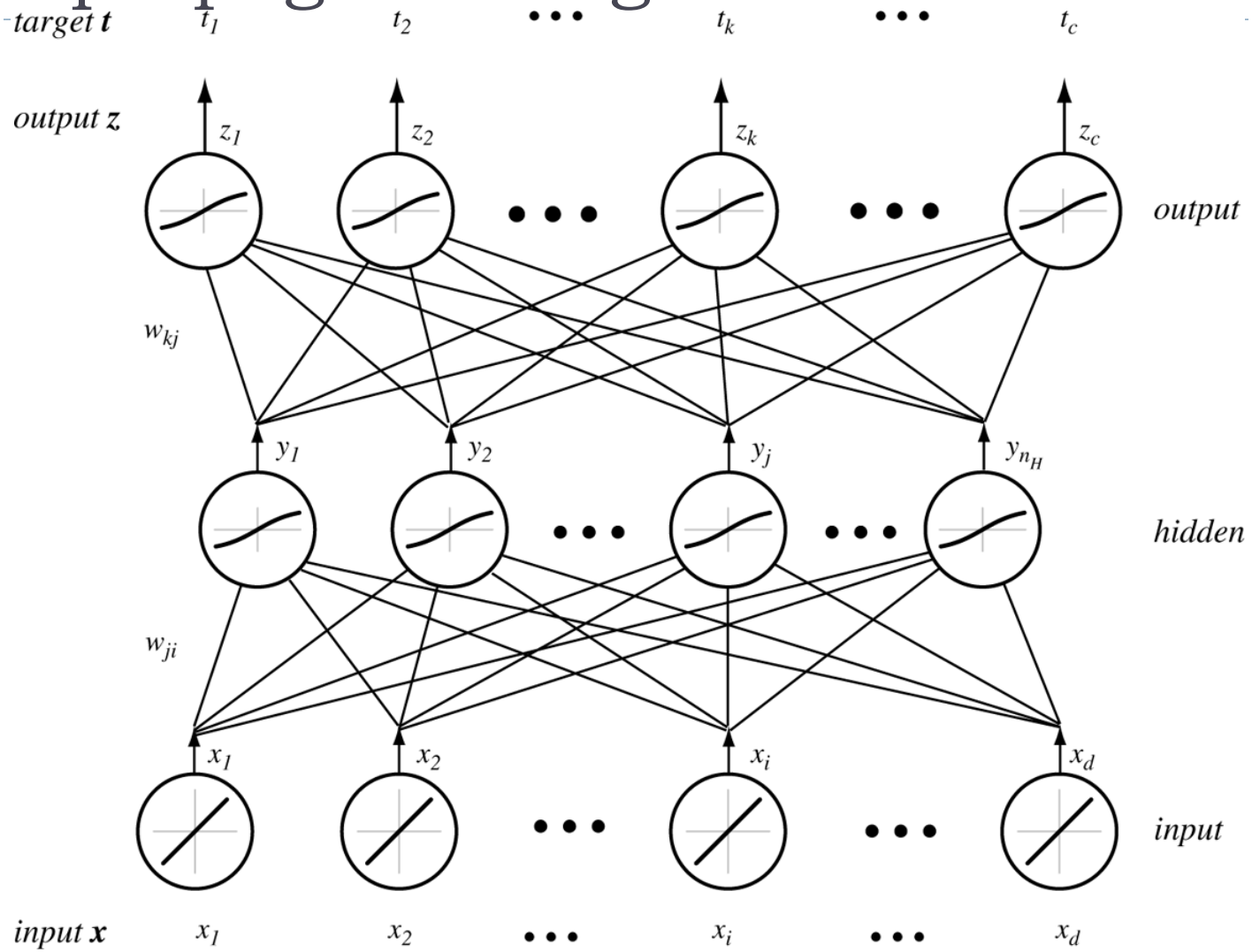
Feedforward Operation and Classification



Backpropagation Algorithm

- The crucial problem of setting the weights based on training patterns and the desired output
- Backpropagation
  - one of the simplest and most general methods for supervised training of MLP.
  - Natural extension of the LMS algorithm
  - LMS works for two-layer systems could be evaluated for each output unit.
- How should the input-to-hidden weights be learned, the one governing the nonlinear transformation of the input vectors?
  - No explicit teacher to state what *the hidden unit's output* should be – *the credit assignment problem*.
  - BP allows us to calculate an effective error for each hidden unit.
  - Two primary modes of operation: *feedforward* and *learning*.
  - Supervised learning consists of presenting an input pattern and changing the network parameters to bring the actual outputs closer to the desired teaching or target values.

Backpropagation Algorithm



Backpropagation Algorithm

- Network learning
  - Starting with an untrained network.
    - A training pattern is presented to the input layer.
    - The corresponding outputs are compared to the target values: *error*
    - The weights are adjusted to reduce the measure of error.
  - Training error

$$J(\mathbf{w}) \equiv \frac{1}{2}\sum_{k=1}^{c}(t_k - z_k)^2 = \frac{1}{2}\|\mathbf{t}-\mathbf{z}\|^2$$

$$\Delta \mathbf{w} = -\eta \frac{\partial J}{\partial \mathbf{w}} \qquad \Delta w_{pq} = -\eta \frac{\partial J}{\partial w_{pq}}$$

    - BP is based on gradient descent: *the weights are initialized with random values*, and then they are changed in a direction that will reduce the error.

$$\mathbf{w}(m+1) = \mathbf{w}(m) + \Delta \mathbf{w}(m)$$

m: indexes the particular pattern presentation

    - η: the learning rate indicates the relative size of the change in weights.

Backpropagation Algorithm

- Network learning(cont.)
  - Sensitivity
    - Describes how the overall error changes with the unit's net activation.

$$\frac{\partial J}{\partial w_{kj}} = \frac{\partial J}{\partial net_k}\frac{\partial net_k}{\partial w_{kj}} = -\delta_k \frac{\partial net_k}{\partial w_{kj}} = -\delta_k y_j$$

sensitivity

Eq. (4)

$$net_k = \sum_{j=1}^{n_H} y_j w_{kj} + w_{k0} = \sum_{j=0}^{n_H} y_j w_{kj} = \mathbf{w}_k^t \mathbf{y}$$

  - *hidden-to-output*

$$\delta_k = -\frac{\partial J}{\partial net_k} = -\frac{\partial J}{\partial z_k}\frac{\partial z_k}{\partial net_k} = (t_k - z_k) f'(net_k)$$

$$\Delta w_{kj} = -\eta \frac{\partial J}{\partial w_{kj}} = \eta \delta_k y_j = \eta (t_k - z_k) f'(net_k) y_j$$

in the case that the output unit is linear then it is simply the LMS rule

Backpropagation Algorithm

- Network learning(cont.)
  - *input-to-hidden*
    - It is the crux of the solution to the credit assignment problem.

$$\frac{\partial J}{\partial w_{ij}} = \frac{\partial J}{\partial y_j}\frac{\partial y_j}{\partial net_j}\frac{\partial net_j}{\partial w_{ji}}$$

$$\frac{\partial J}{\partial y_j} = \frac{\partial}{\partial y_j}\left[\frac{1}{2}\sum_{k=1}^{c}(t_k - z_k)^2\right]$$

$$= -\sum_{k=1}^{c}(t_k - z_k)\frac{\partial z_k}{\partial y_j}$$

$$= -\sum_{k=1}^{c}(t_k - z_k)\frac{\partial z_k}{\partial net_k}\frac{\partial net_k}{\partial y_j}$$

$$= -\sum_{k=1}^{c}(t_k - z_k) f'(net_k) w_{kj}$$

$$net_j = \sum_{i=0}^{d} x_i w_{ji} = \mathbf{w}_j^t \mathbf{x}$$

$$\frac{\partial net_j}{\partial w_{ji}} = x_i$$

Backpropagation Algorithm

- Network learning(cont.)
  - *input-to-hidden*

    the sensitivity for a hidden unit

$$\delta_j \equiv -\frac{\partial J}{\partial net_j} = -\frac{\partial J}{\partial y_j}\frac{\partial y_j}{\partial net_j} = \sum_{k=1}^{c}(t_k - z_k) f'(net_k) w_{kj} \cdot f'(net_j) = f'(net_j)\sum_{k=1}^{c} w_{kj}\delta_k$$

    - The sum of the individual sensitivities at the output units weighted by the hidden-to-output weights, all multiplied by $f'(net_j)$.

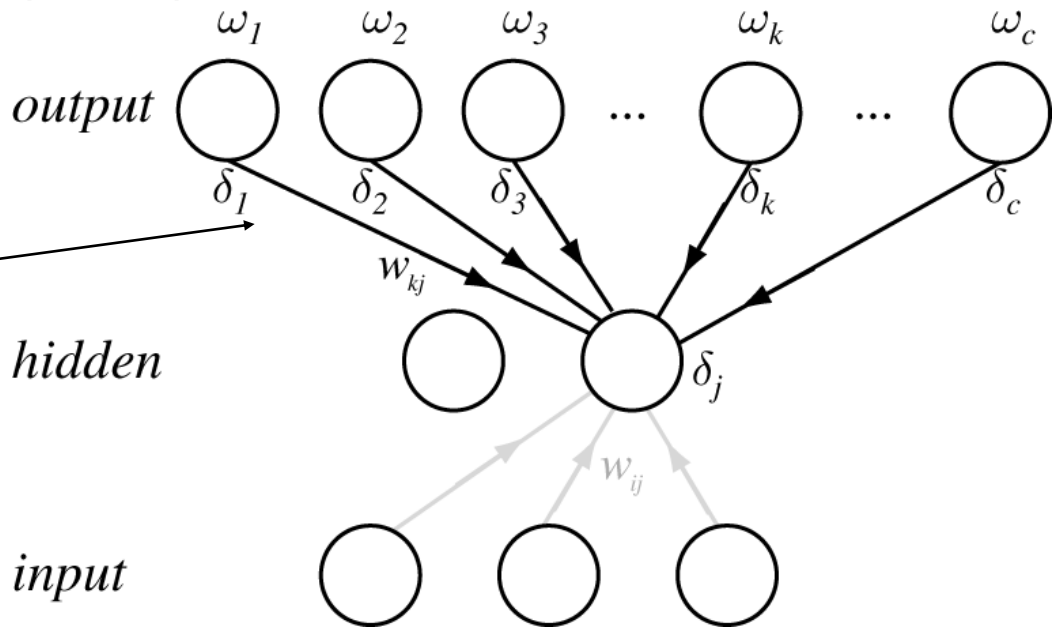
$$\Delta w_{ji} = \eta x_i \delta_j = \eta \underbrace{\left[\sum_{k=1}^{c} w_{kj}\delta_k\right] f'(net_j)}_{\delta_j} x_i \qquad \left(\Delta \mathbf{w} = -\eta \frac{\partial J}{\partial \mathbf{w}}\right)$$

    - *Backpropagation of errors:* during training an error must be propagated from the output layer back to the hidden layer in order to perform the learning of the input-to-hidden weights.
      - *Continuous functions* allows the computation of derivatives of the criterion function with respect to all model weights.

Backpropagation Algorithm

- Network learning(cont.)
  - *input-to-hidden*

output sensitivities are propagated "back" to the hidden units



$$\Delta w_{ji} = \eta x_i \delta_j = \eta \underbrace{\left[\sum_{k=1}^{c} w_{kj}\delta_k\right] f'(net_j)}_{\delta_j} x_i$$

- Initialization of the weights with random values:
  - Supposed that the weights are initialized with zeros!

Backpropagation Algorithm

- BP can be generalized directly to feedforward networks in which
  - Input units include a bias unit
  - Input units are connected directly to output units as well as to hidden units.
  - There are more than three layers of units.
  - There are different nonlinearities for different layers.
  - Each unit has its own nonlinearities.
  - Each unit has a different learning rate.

Backpropagation Algorithm

- Training Protocols
  - Supervised training consists in presenting to the network training set finding the output of the net and adjusting the weights so as to make the actual output.
  - Three useful training protocols:
    - Stochastic – patterns are chosen randomly from the training set, and the weights are updated for each pattern presentation.
    - Batch – all patterns are presented to the network before learning takes place.
    - On-line – each pattern is presented once and only once; no use of memory for storing the patterns.
  - Epoch:
    - one epoch corresponds to a single presentations of all patterns in the training set.
    - The number of epochs is an indication of the relative amount of learning.

Backpropagation Algorithm

- Training Protocols
  - Stopping criterion
    - When *the change* in the criterion function $J(\mathbf{w})$ is smaller than some preset value θ.
    - We need to consider an error defined over the entirety of patterns in the training set.

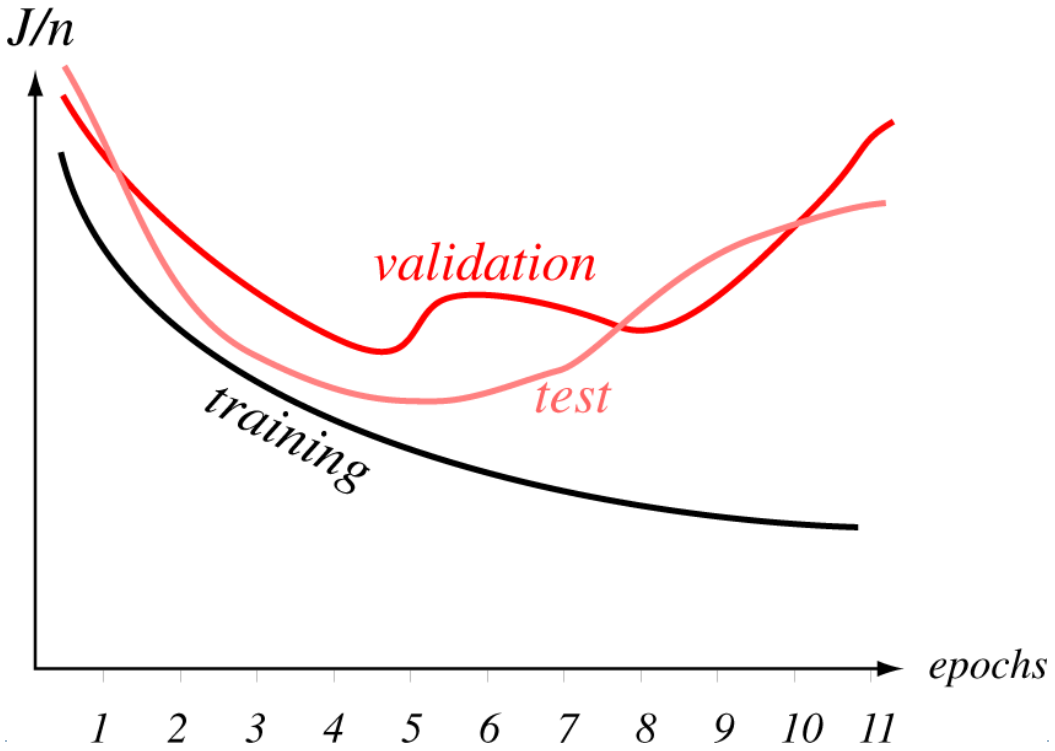
$$J = \sum_{p=1}^{n} J_p$$

Backpropagation Algorithm

- Learning Curves
  - Before training has begun, the error on the training set is typically high.
  - Through learning the error becomes lower : *learning curve*
  - The training error ultimately reaches an asymptotic value which depends on
    - The Bayes error, the amount of training data, and the expressive power in the network.
  - The average error on an independent test set is virtually always higher than on the training set, and while it generally decreases, it can increase or oscillate.
  - In addition to the use of the training set, there are two conceptual uses of independently selected patterns:
    - Test set – used to state the performance of the network.
    - Validation set – used to decide when to stop training.

Backpropagation Algorithm

- Learning Curves
  - Validation set : indirect representative of novel patterns yet to be classified.
    - Can be used in a stopping criterion.

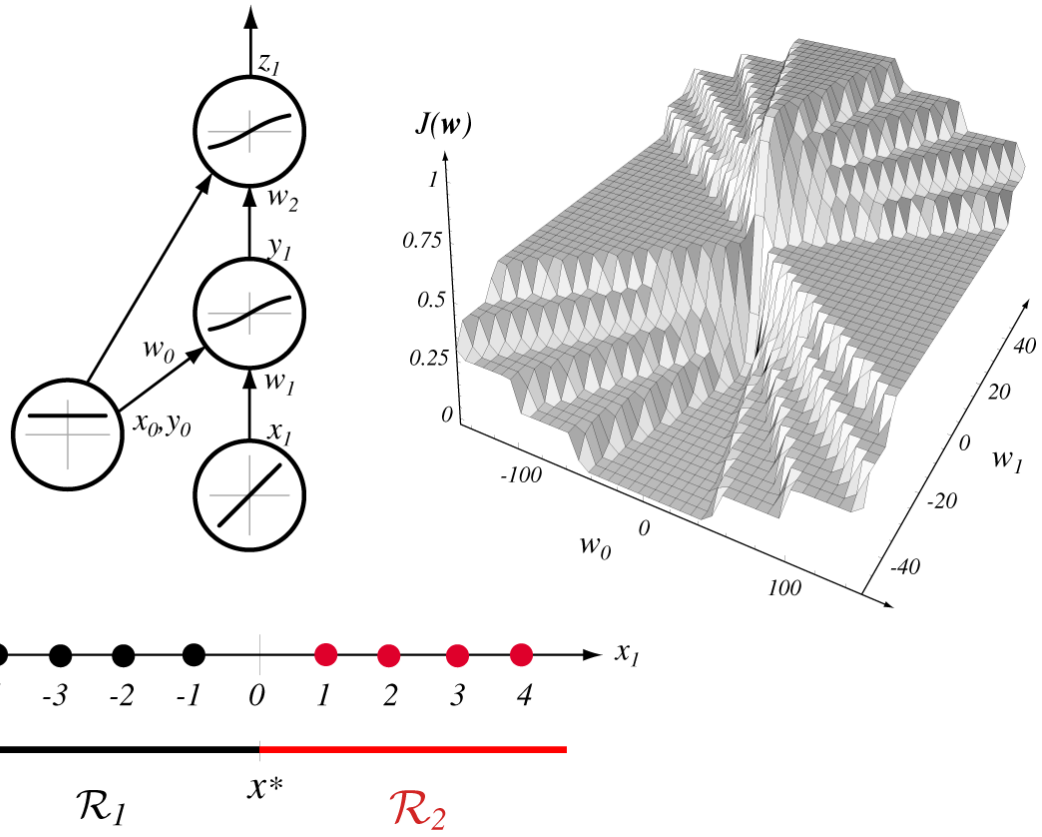


Error Surfaces

- By studying error surfaces, $J(\mathbf{w})$, we can gain understanding and intuition about the algorithm.
  - Local minima vs. Global minimum
  - Presence of plateaus
    - The error varies only slightly as a function of weights.

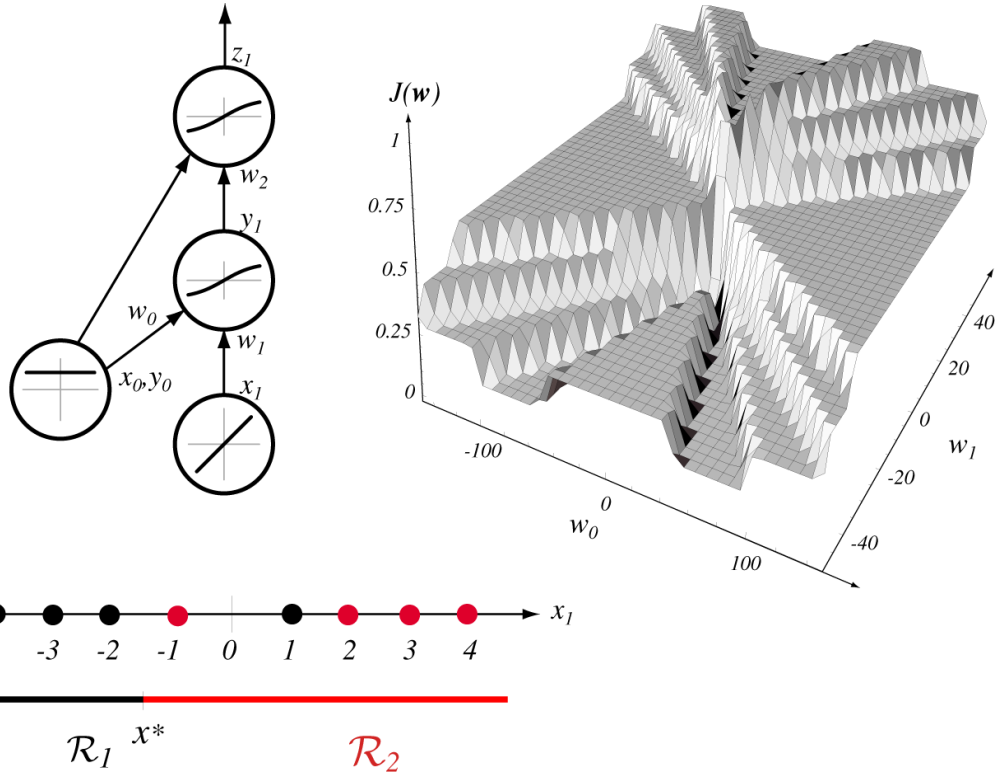
- Some small networks
  - The simplest three-layer nonlinear network.
    - 1-1-1 sigmoidal network (and bias).
    - The data shown are linearly separable.
    - During learning, the weights descend to the global minimum and the problem is solved.

Error Surfaces



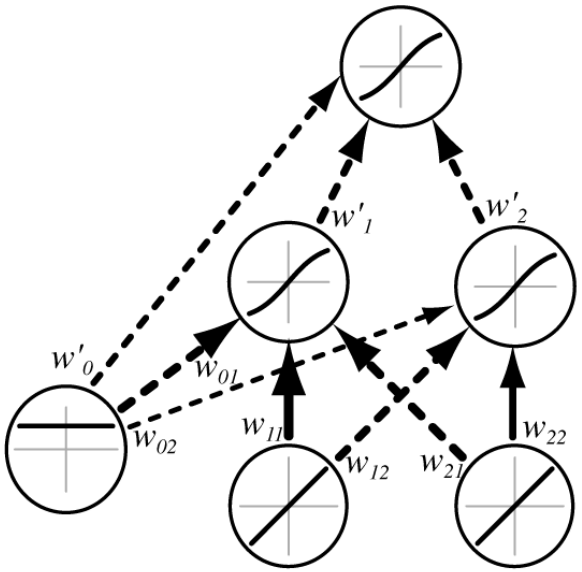
Error Surfaces

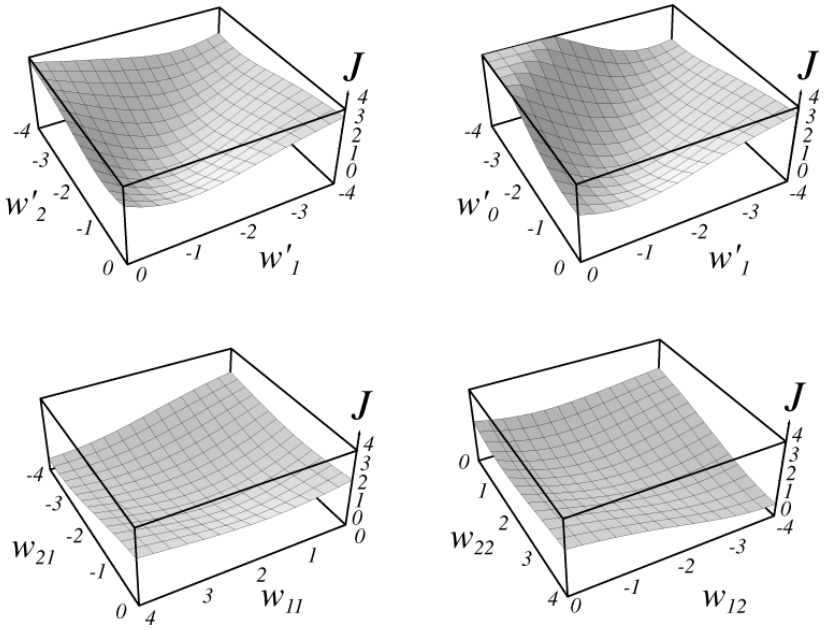
- Some small networks
  - The simplest three-layer nonlinear network.
    - The same network applied to another one that is not linearly separable.



Error Surfaces

- The Exclusive-OR



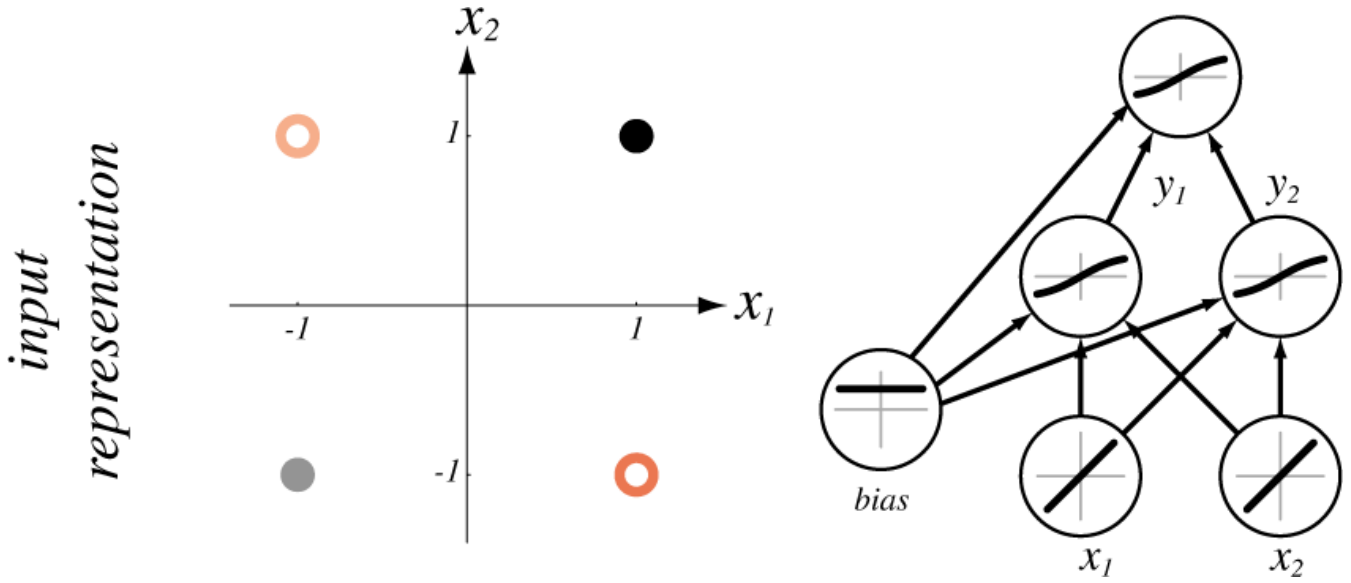


Error Surfaces

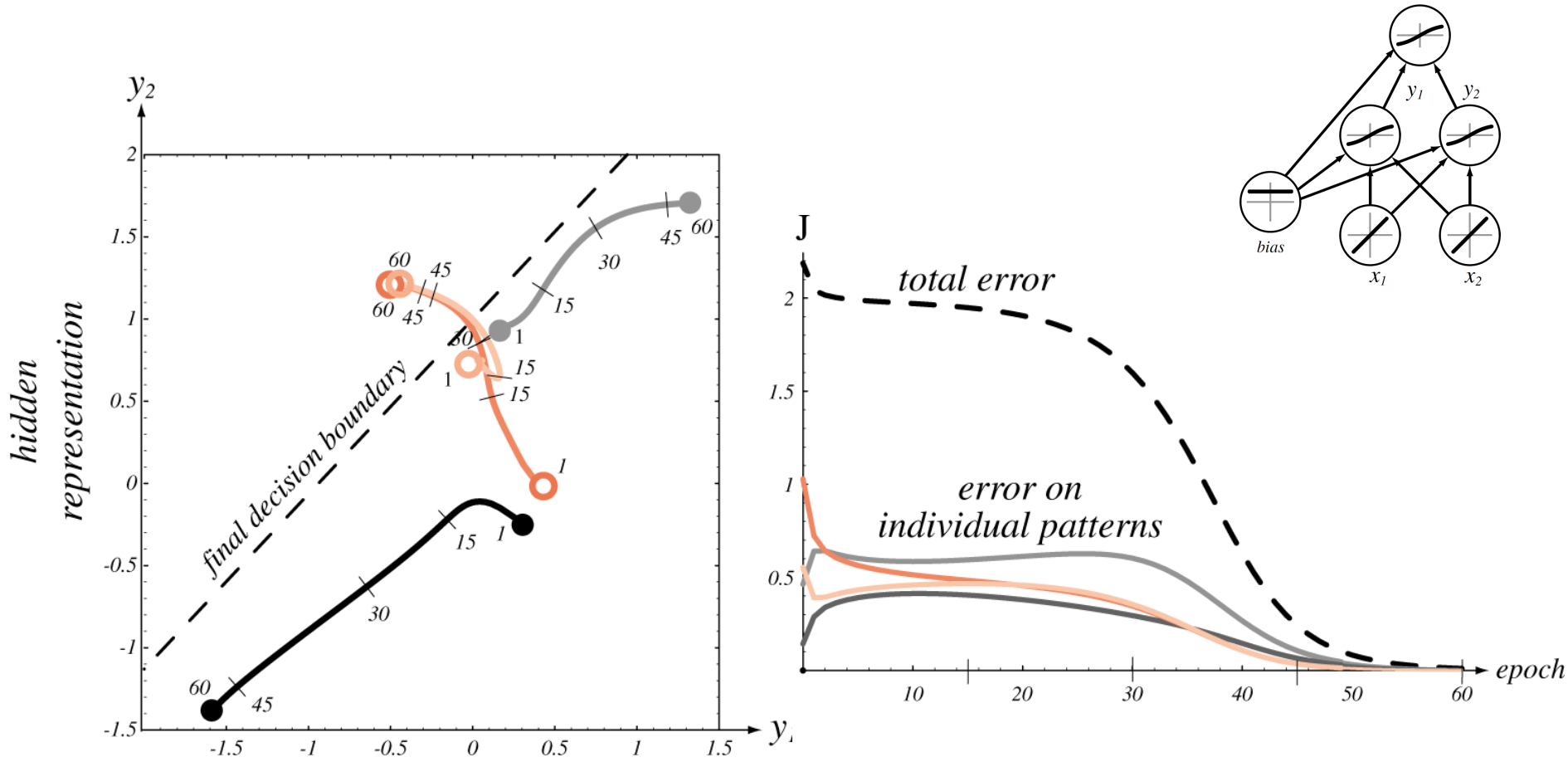
- Larger Networks
  - For a network with many weights solving a complicated high-dimensional classification problem, the error varies quite gradually as a single weight is changed.
  - Whereas in low-dimensional spaces, local minima can be plentiful, in high dimension, the problem of local minima is different: the high-dimensional space may afford more ways for the system to get around a local maximum during learning.
    - The more superfluous the weights, the less likely it is a network will get trapped in local minima.
    - Networks with an unnecessarily large number of weights are undesirable because of the dangers of overfitting.

Backpropagation as Feature Mapping

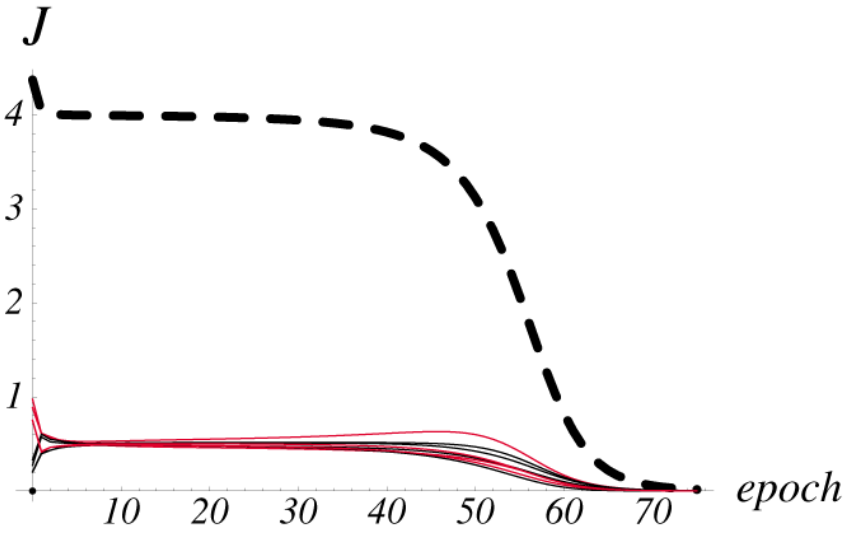
- The novel computational power provided by MPL
  - Nonlinear warping of the input to the representation at the hidden units.
- The nonlinearities of the hidden units warp and distort the mapping from input to the hidden unit space.
  - The nonlinearly separable problem at the inputs is transformed into a linearly separable at the hidden units.

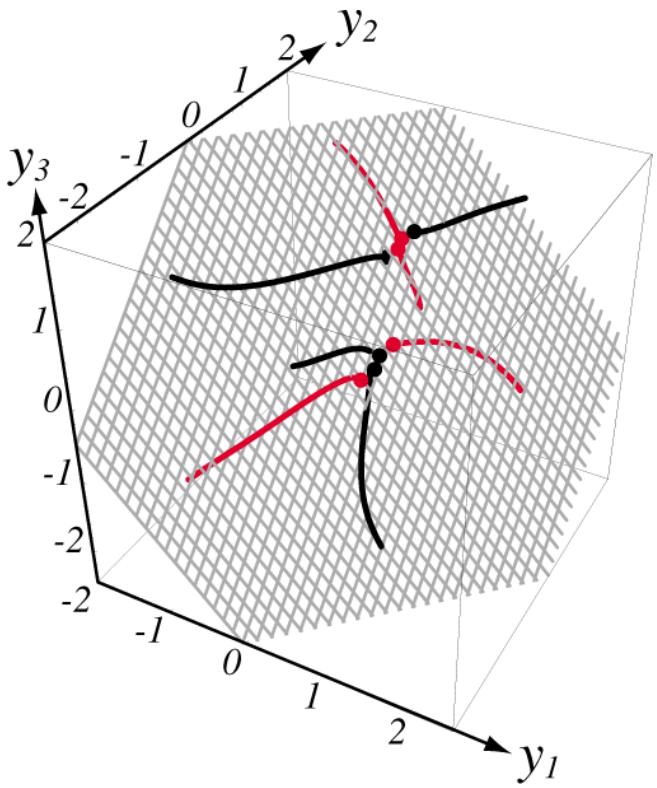


Backpropagation as Feature Mapping

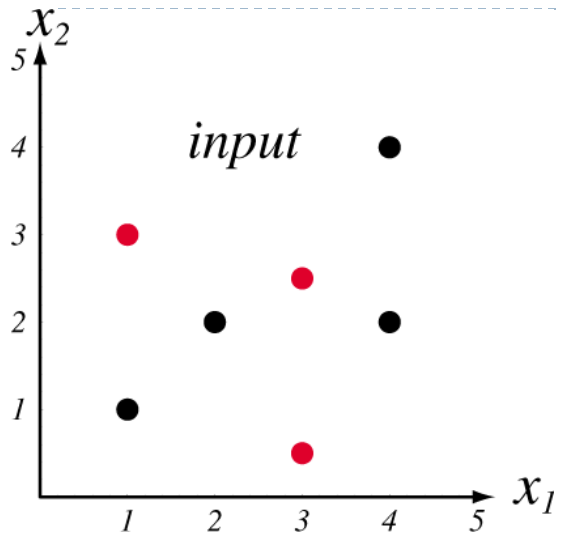


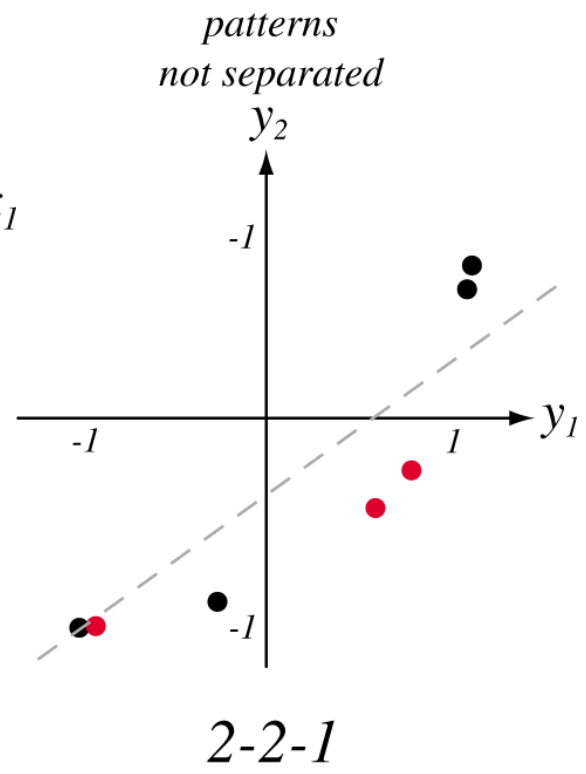
Backpropagation as Feature Mapping

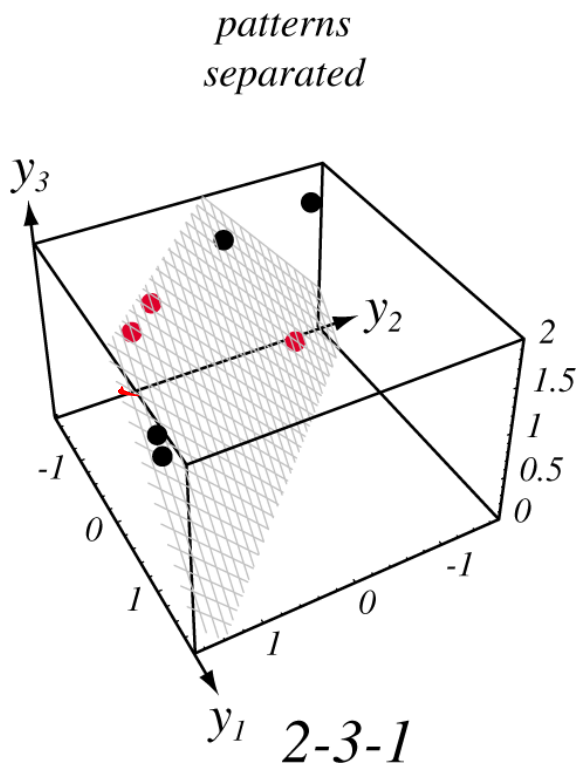




Backpropagation as Feature Mapping

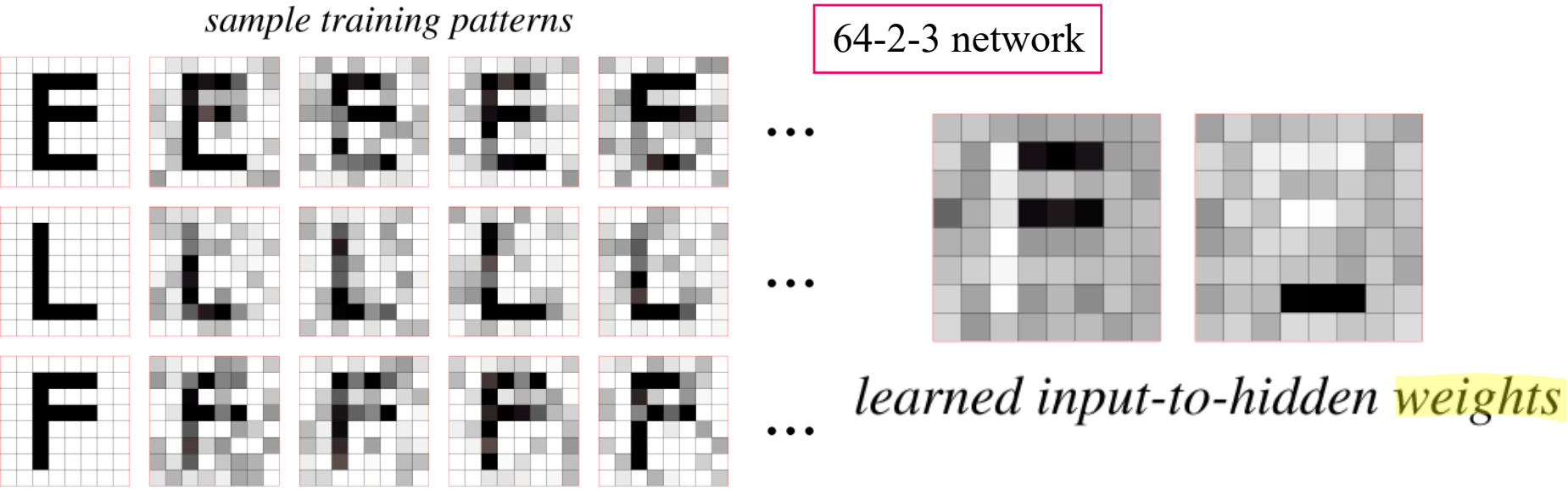






Backpropagation as Feature Mapping

- Representation at the hidden layer – weights
  - Representation of learned weights
    - Because the hidden-to-output weights merely leads to a linear discriminant, it is the input-to-output weights that are most instructive.
    - To think of the hidden units as finding feature groupings useful for the linear classifier implemented by the hidden-to-output layer weights,



Backpropagation, Bayes theory and Probability

- Bayes discriminants and neural networks

$$P(\omega_k|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_k)P(\omega_k)}{\sum_{i=1}^{c} p(\mathbf{x}|\omega_k)P(\omega_i)} = \frac{p(\mathbf{x},\omega_k)}{p(\mathbf{x})}$$

$g_k(\mathbf{x};\mathbf{w})$ Output of the kth output unit

$$g_k(\mathbf{x}) = P(\omega_k|\mathbf{x}) \qquad t_k(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} \in \omega_k \\ 0 & \text{otherwise} \end{cases}$$

$$J(w) = \sum_{\mathbf{x}} [g_k(\mathbf{x};\mathbf{w}) - t_k]^2$$

$$= \sum_{\mathbf{x}\in\omega_k} [g_k(\mathbf{x};\mathbf{w}) - 1]^2 + \sum_{\mathbf{x}\notin\omega_k} [g_k(\mathbf{x};\mathbf{w}) - 0]^2$$

$$= n\left\{ \frac{n_k}{n} \frac{1}{n_k} \sum_{\mathbf{x}\in\omega_k} [g_k(\mathbf{x};\mathbf{w}) - 1]^2 + \frac{n-n_k}{n} \frac{1}{n-n_k} \sum_{\mathbf{x}\notin\omega_k} [g_k(\mathbf{x};\mathbf{w}) - 0]^2 \right\}$$

Backpropagation, Bayes theory and Probability

- Bayes discriminants and neural networks

$$\lim_{n\to\infty}\frac{1}{n}J(\mathbf{w}) \equiv \tilde{J}(\mathbf{w})$$

$$= P(\omega_k)\int\left[g_k(\mathbf{x};\mathbf{w}) - t_k\right]^2 p(\mathbf{x}|\omega_k)d\mathbf{x} + P(\omega_{i\neq k})\int g_k^2(\mathbf{x};\mathbf{w})p(\mathbf{x}|\omega_{i\neq k})d\mathbf{x}$$

$$= \int g_k^2(\mathbf{x};\mathbf{w})p(\mathbf{x})d\mathbf{x} - 2\int g_k(\mathbf{x};\mathbf{w})p(\mathbf{x},\omega_k)d\mathbf{x} + \int p(\mathbf{x},\omega_k)d\mathbf{x}$$

$$= \int\left[g_k(\mathbf{x};\mathbf{w}) - P(\omega_k|\mathbf{x})\right]^2 p(\mathbf{x})d\mathbf{x} + \underbrace{\int P(\omega_k|\mathbf{x})P(\omega_{i\neq k}|\mathbf{x})p(\mathbf{x})d\mathbf{x}}_{\text{independent of } \mathbf{w}}$$

$$\sum_{k=1}^{c}\int\left[g_k(\mathbf{x};\mathbf{w}) - P(\omega_k|\mathbf{x})\right]^2 p(\mathbf{x})d\mathbf{x}$$

  - In the limit of infinite data the outputs of the trained network will approximate the true a posteriori probabilities in a least-squares sense.

$$g_k(\mathbf{x};\mathbf{w}) \approx P(\omega_k|\mathbf{x})$$

Backpropagation, Bayes theory and Probability

- Outputs as Probabilities
  - Given infinite amounts of training data, the outputs will represent probabilities. However, if the conditions do not hold, the outputs will not represent probabilities.
    - There is not even a guarantee that they sum to 1.0.
  - Softmax
    - A smoothed or continuous version of a winner-take-all.

$$z_k = \frac{e^{net}}{\sum_{m=1}^{c} e^{net_m}}$$

    - The use of softmax is appropriate when the network is to be used for estimating probabilities.

Practical Techniques for Improving BP

- Up to here, a number of practical considerations has been ignored for the sake of simplicity.
  - A naïve application of the procedure may lead to very slow convergence, poor performance or other unsatisfactory results.
  - A number of practical suggestions for training networks by BP will be presented – based on a number of plausible heuristics.

- Active function
  - General properties we seek:
    - Must be nonlinear – otherwise the 3-layer network provides no computational power above that of a two-layer net.
    - Saturate – have some maximum and minimum output value. This will keep the weights and activations bounded, and thus keep the training time limited.
    - Continuity and smoothness – $f()$ and $f'()$ be defined.
    - Monotonicity is another convenient, but nonessential.

Practical Techniques for Improving BP

- Active function (cont.)
  - Sigmoid
    - The most widely used activation function.
    - One class of functions that has all the above desired properties.
    - Hyperbolic tangent
    - Smooth, differentiable, nonlinear, and saturating.
    - f'() can be easily expressed in terms of f() itself.

- Parameters for the sigmoid

$$f(net) = a\tanh(b\,net) = a\left[\frac{e^{+b\,net} - e^{-b\,net}}{e^{+b\,net} + e^{-b\,net}}\right]$$

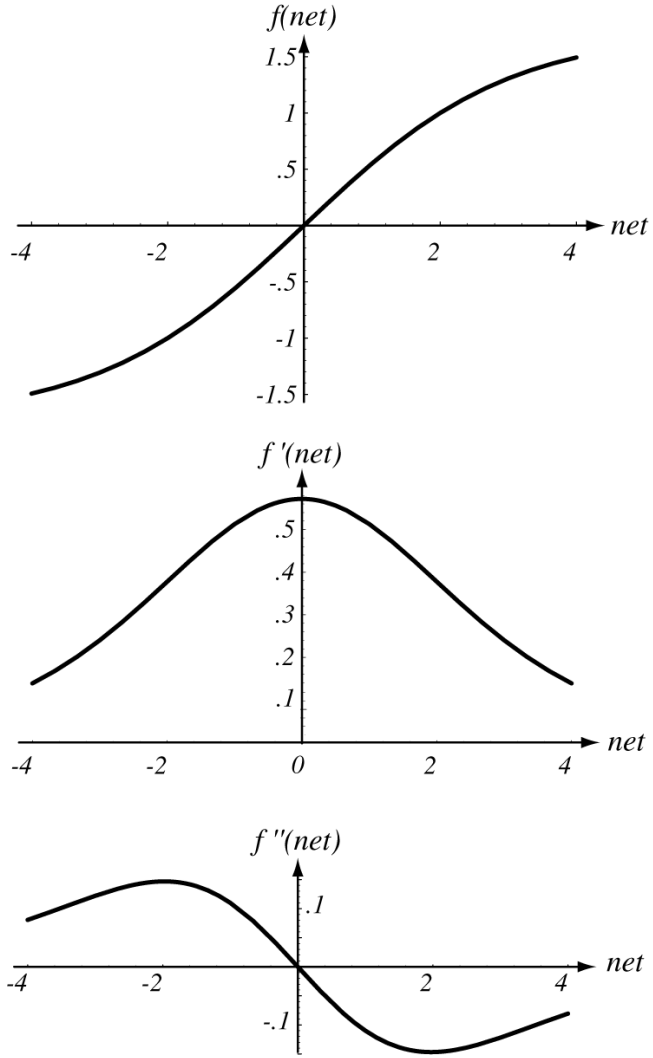
Practical Techniques for Improving BP

- Parameters for the sigmoid

$$f(net) = a\left[\frac{e^{+b\,net} - e^{-b\,net}}{e^{+b\,net} + e^{-b\,net}}\right]$$

$$a = 1.716$$

$$b = 2/3$$



Practical Techniques for Improving BP

- Scaling Input
  - Classifying fish based on weight (in grams) and length (in meters).
    - Such a representation will have serious drawbacks for a neural network classifier.
  - We do not want our classifier prefer one of features over the other, because they differ solely in the arbitrary representation.
  - To avoid such difficulties,
    - the input patterns should be shifted so that the average over the training set of each feature is zero.
    - The full data set should be scaled to have the same variance in each feature component.
      - Standadization can only be done for stochastic and batch learning protocol.

Practical Techniques for Improving BP

- Target values
  - During the training we used a one-of-*c* representation for the target vector. But the output units saturate at ± 1.716.
    - For any finite value of net_k, the output could never reach these saturation values, and there would be error – weights would become extremely large as net_k would be driven to $\pm$infinity.
  - The difficulty can be avoided by using teaching values of +1 for the target category and –1 for the non-target categories.

$$\mathbf{t} = \begin{bmatrix} -1 & -1 & +1 & -1 \end{bmatrix}^t$$

    - This target representation yields efficient learning for categorization.

Practical Techniques for Improving BP

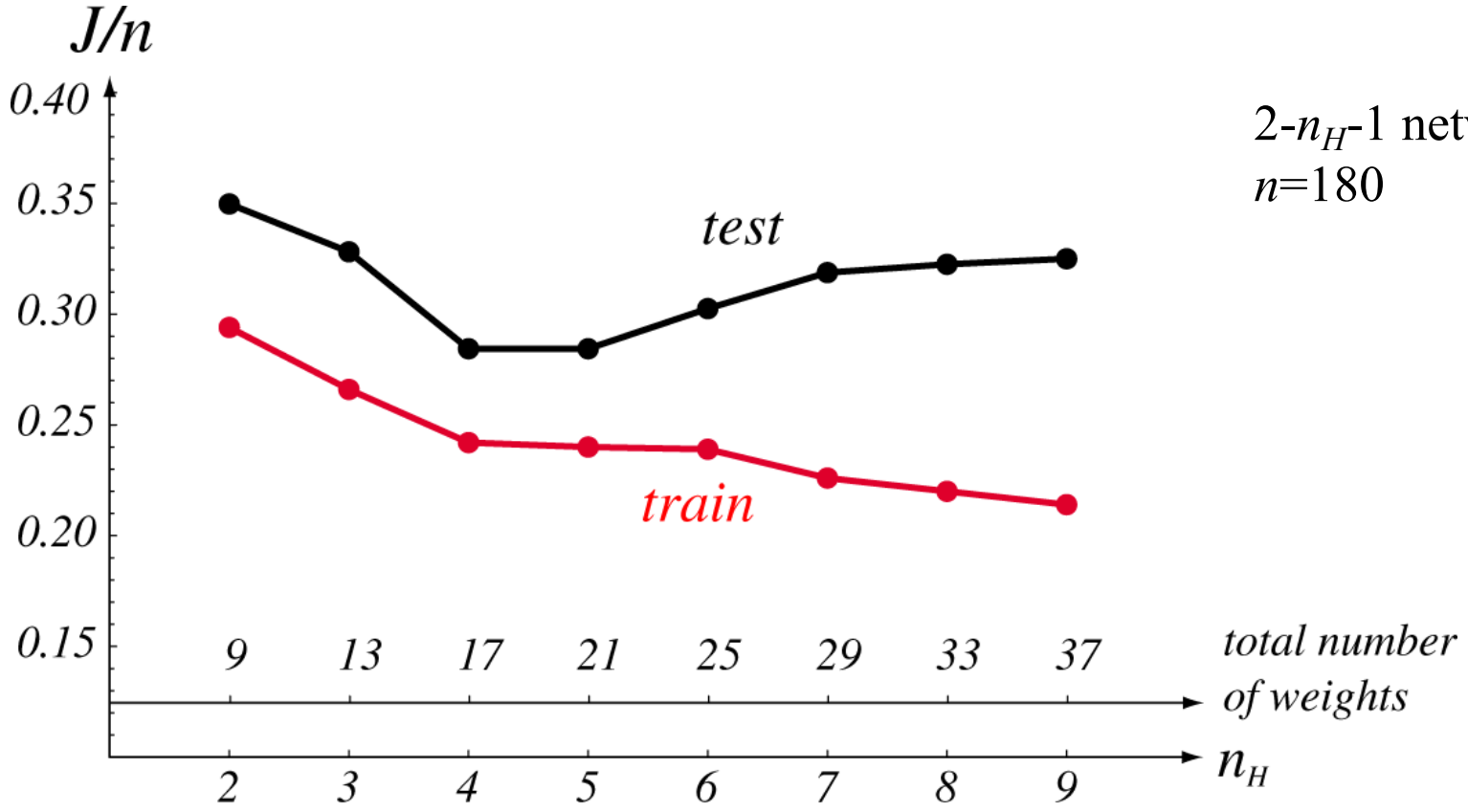
- Training with Noise
  - When the training set is small, one can generate virtual or surrogate training patterns and use them as if they were normal training patterns.
  - In the absence of problem-specific information, such surrogate patterns should be made by adding *d*-dimensional Gaussian noise to true training points.
- Manufacturing data
  - If we have knowledge about the sources of variation among patterns, we can manufacture training data that conveys more information than does the method of training with uncorrelated noise.
    - In an OCR, an input image may be presented rotated by various amounts.

Practical Techniques for Improving BP

- Number of Hidden Units
  - While the number of input units and output units are dictated by the dimensionality of the input vectors and the number of categories, *the number of hidden units is not simply related to such obvious properties of the classification problem*.
  - The number of hidden units governs the expressive power of the net – and thus the complexity of the decision boundary.
    - If the patterns are well-separated or linearly separable, then few hidden units are needed.
  - There is no foolproof method for setting the number of hidden units before training.
  - The number of hidden units determines the total number of weights in the net – the number of degrees of freedom.
  - A rule of thumb
    - the total number of weights in the net is roughly n/10, where n is the total number of training points.
  - Start with a large number of hidden units and decay weights.

Practical Techniques for Improving BP

- Number of Hidden Units



2-n_H-1 network
n=180

Practical Techniques for Improving BP

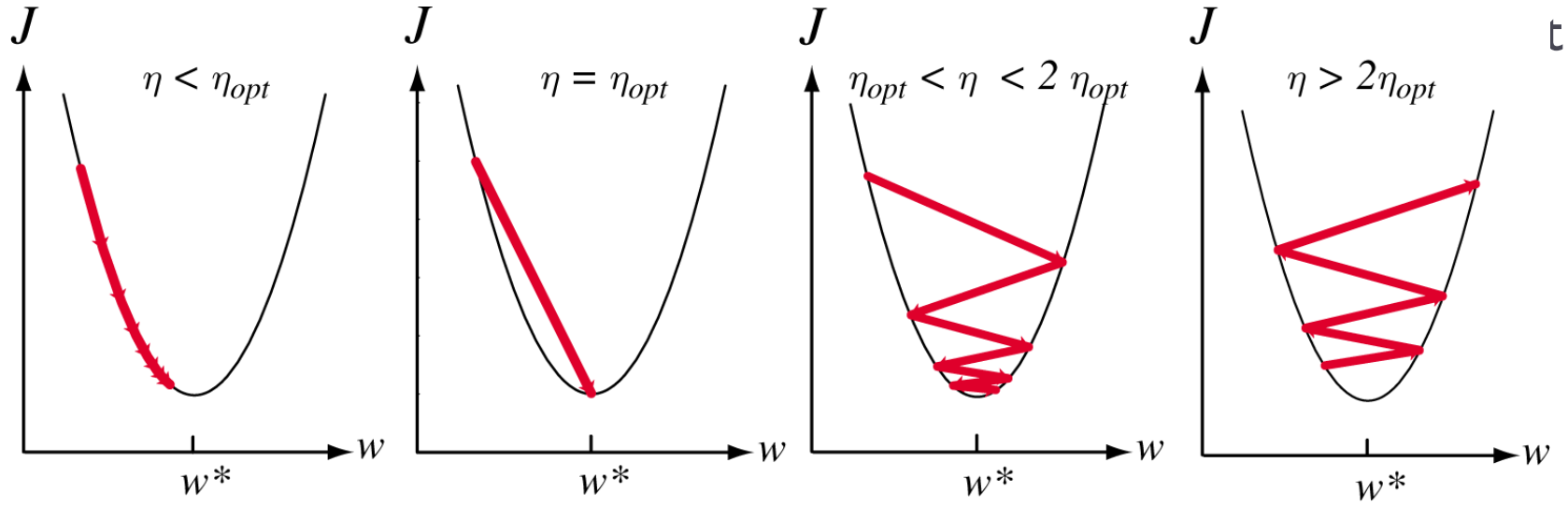
- Initializing weights
  - We cannot initialize the weights to 0, otherwise learning cannot take place.
  - We seek to set the initial values in order to have fast and uniform learning.
  - In setting weights in a given layer, we choose weights randomly from a single distribution to help ensure uniform learning.

$$-1/\sqrt{d} < \omega_{ji} < 1/\sqrt{d}$$

$$-1/\sqrt{n_H} < \omega_{kj} < 1/\sqrt{n_H}$$

Practical Techniques for Improving BP

- Learning rates
  - Learning rate can affect the quality of the final network.
  - If some of weights converge significantly earlier than others then the network may not perform equally well throughout the full range of inputs, or equally well for the patterns in each category.



Practical Techniques for Improving BP

- Momentum
  - Momentum
    - Loosely based on the notion from physics that moving objects tend to keep moving unless acted upon by outside forces.
  - Plateaus in error surfaces
    - The slope is very small.
    - There are too many weights depends only weakly upon one of them.

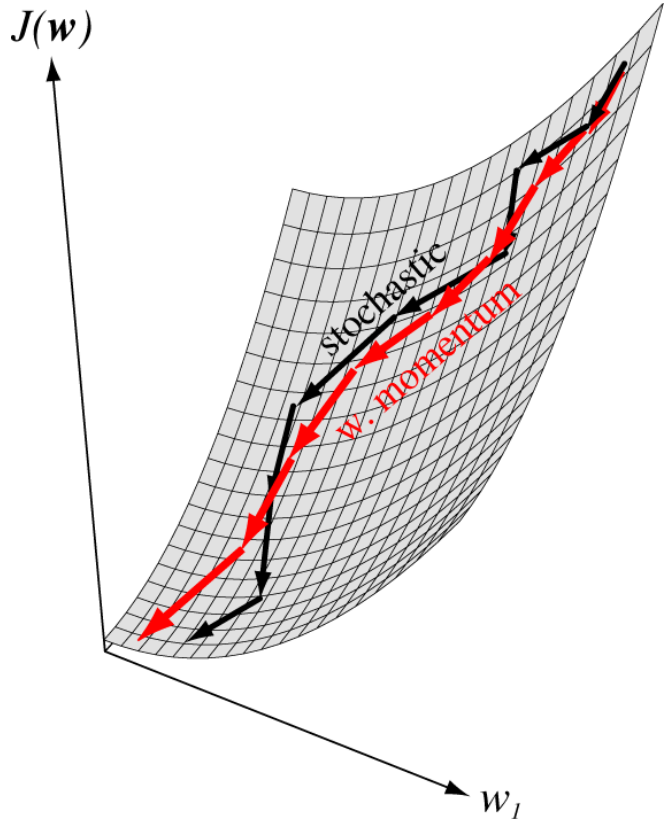
$$\mathbf{w}(m+1) = \mathbf{w}(m) + (1-\alpha)\Delta\mathbf{w}_{bp}(m) + \alpha\Delta\mathbf{w}(m-1)$$

- If α = 0 : standard backpropagation

  α = 1 : backpropagation is ignored

Practical Techniques for Improving BP

- Momentum



Practical Techniques for Improving BP

- Weight Decay
  - One method of simplifying a network and avoiding overfitting is to impose a heuristic that the weights should be small
    - To start with a network with "too many" weights and "decay" all weights during training
    - Small weights – nearly linear
    - Weights that are not needed for reducing the criterion function become smaller and smaller.

$$\omega^{new} = \omega^{old}(1-\varepsilon)$$

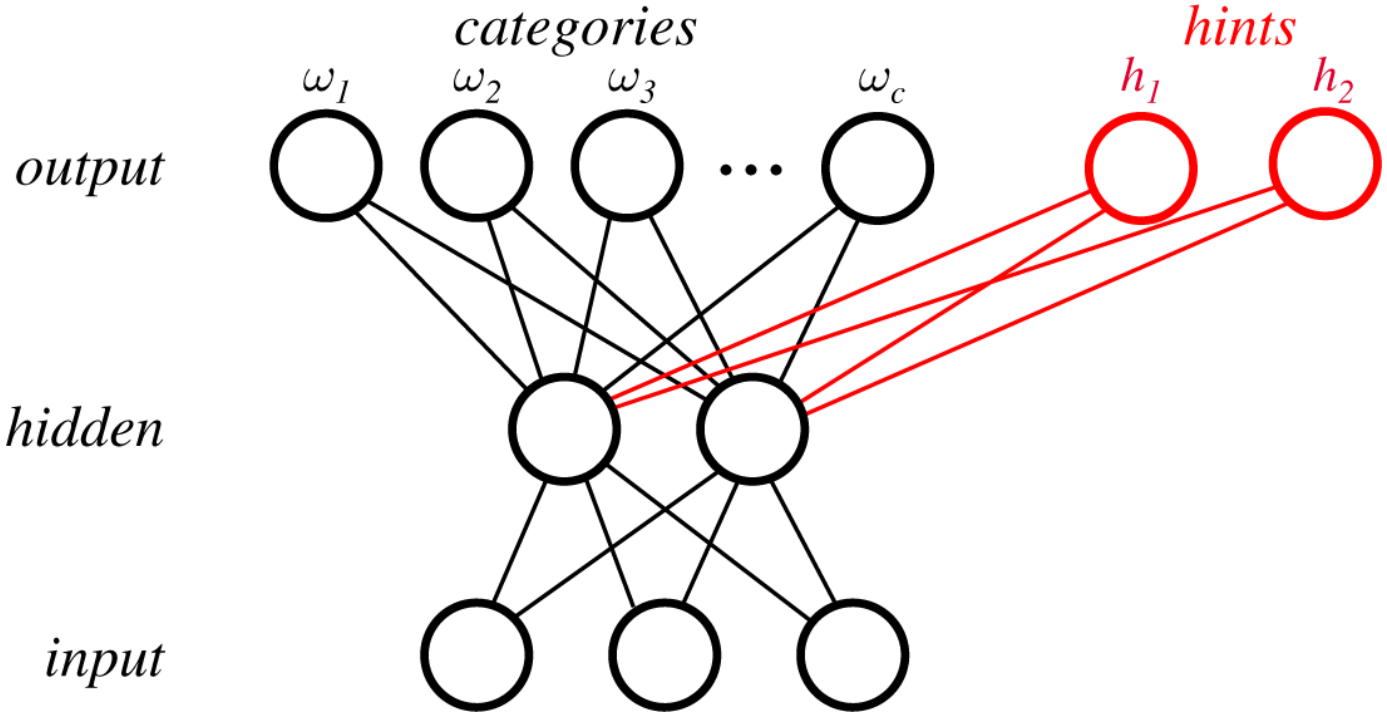
$$J_{ef} = J(\mathbf{w}) + \frac{2\varepsilon}{\eta}\mathbf{w}^t\mathbf{w}$$

Practical Techniques for Improving BP

- Hints
  - Often we have insufficient training data for the desired classification accuracy and we would like to add information or constraints to improve the network.
  - Learning with hints – to add output units for addressing an ancillary problem.
    - During training, the target vector must be lengthened to include components for the hint outputs.
    - During classification, the hint units are not used – they and their hidden-to-output weights can be discarded.
  - The benefit provided by hints is in improved feature selection.
    - The feature groupings useful for the hint task are likely to aid category learning.
      - /g/, /ii/

Practical Techniques for Improving BP

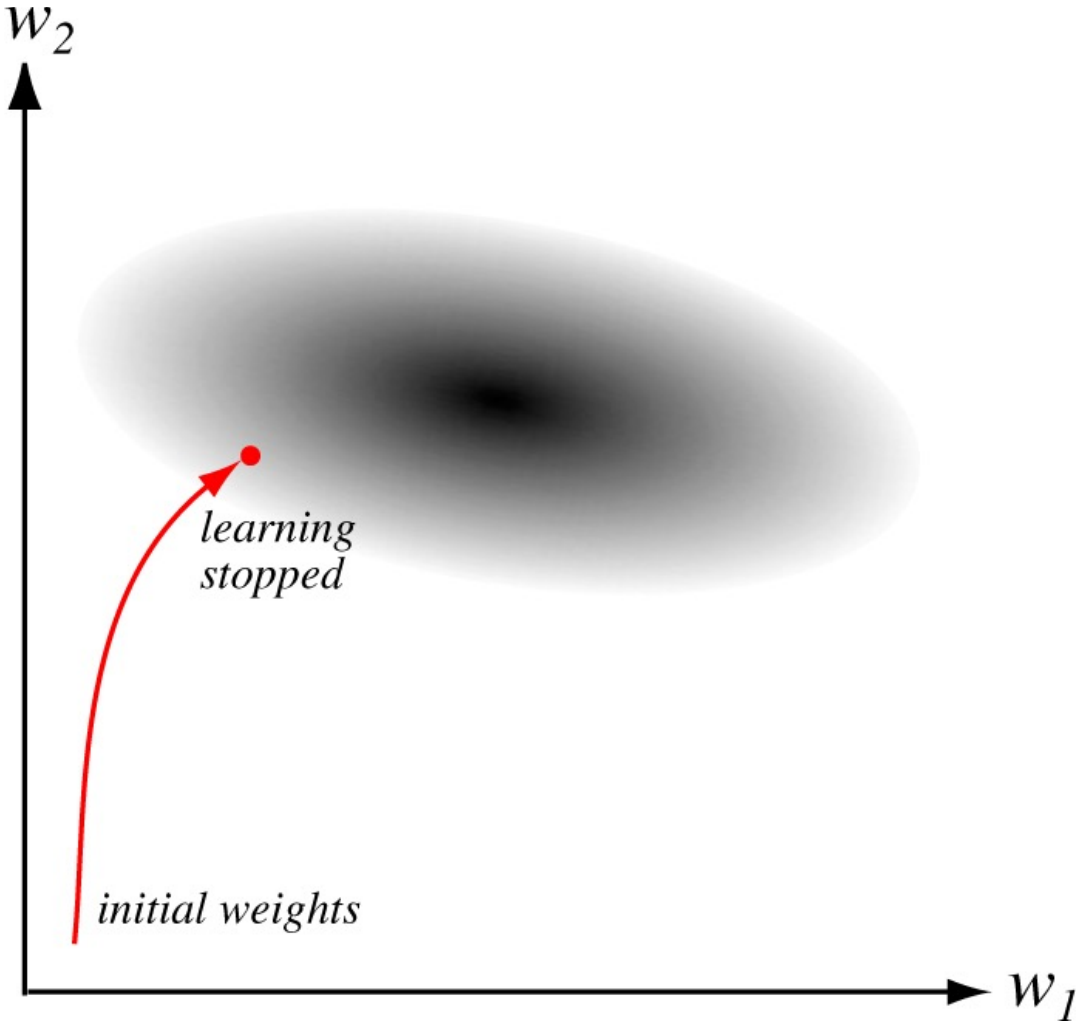
- Hints



Practical Techniques for Improving BP

- On-line, stochastic or batch training
  - Online learning is to be used when the amount of training data is so large, or when memory costs are so high.
  - Batch learning is typically slower than stochastic learning.
  - For most applications – especially ones employing large redundant training sets – stochastic training is to be preferred.
- Stopped training
  - To avoid overfitting.
  - Stopping the training before gradient descent is complete can help avoid overfitting.
  - It is hard to know beforehand what stopping criterion should be.
  - A far simpler method is to stop training when the error on *a separate validation set* reaches a minimum.

Practical Techniques for Improving BP



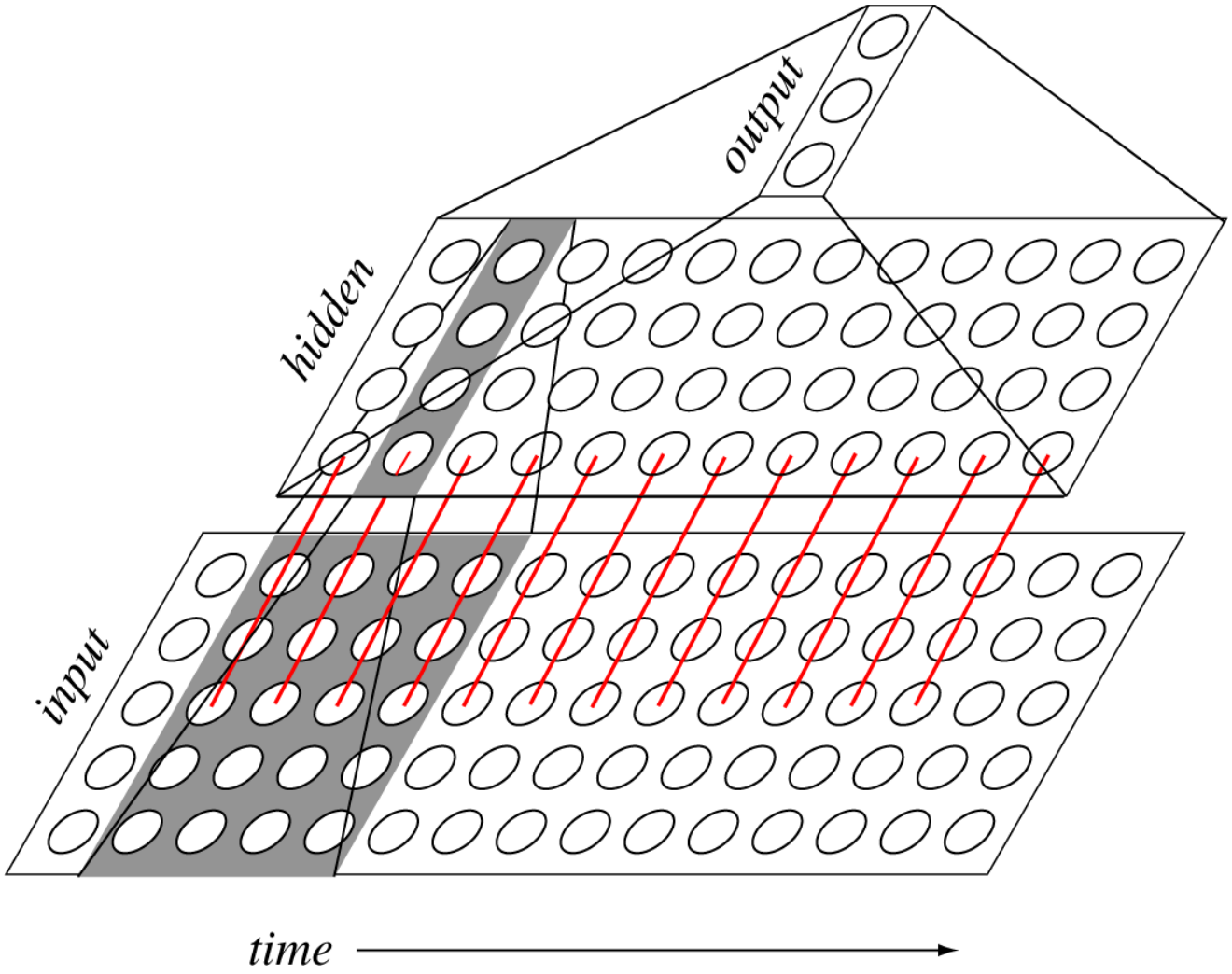
Practical Techniques for Improving BP

- Number of hidden layers
  - Because three layers suffice to implement any arbitrary function, we would need special problem conditions or requirements to recommend the use of more than three layers.
    - Translation, rotation or other distortion invariance.
- Criterion function
  - The squared error is the most common training criterion.
    - Simple to compute – nonnegative, simplifying the proofs of some theorems.
  - Other training criteria have benefits.
    - Cross entropy for n patterns.
    - Minkowski error.

Additional Networks and Training Methods

- Convolutional Networks
  - If we demand that our classifier be insensitive to translations of the pattern, we can effectively replicate the recognizer at all such translations. – Time Delay Neural Networks (TDNNs)
  - TDNN
    - Each hidden unit accepts input from a restricted spatial range of positions in the input layer – delayed locations accept inputs from the input layer that are similarly shifted.
    - Developed for use in speech and other temporal phenomena.

Additional Networks and Training Methods



Regularization, Complexity adjustment and Pruning

- The number of inputs and outputs of a three-layer network are determined by the problem itself.
- We do not know ahead of time the number of hidden units or weights.
  - A danger of overfitting:
    - too many weights and thus degrees of freedom and train too long
- Regularization
  - To make a new criterion function that depends not only on the classical training error, but also on classifier complexity.
    - To balance error on the training set with complexity

$$J = J_{pat} + \lambda J_{reg}$$

- Pruning weights
  - To eliminate weights that are least needed.