27-642/SDE 733
Artificial Neural Networks
The BP Paradigm

Dr. Deborah A. Stacey

January, 1993

The Perceptron Paradigm

Historical Background

- introduced by Rosenblatt in 1957.
- based on a model of the retina.
- the retina contains several light sensors arranged as a matrix.
- the sensor outputs are connected to a set of processing elements which recognize particular patterns.
- the outputs of the PEs go to a threshold logic unit which does not "fire" until a certain level and type of input occurs.
- this model was based on experimental evidence that certain shapes of similar complexity can be more easily learned and recalled than others – visual hardware is tuned to particular shapes.

The Elementary Perceptron

- has the ability to learn to recognize simple patterns.
- the **elementary perceptron** is a two-layer, heteroassociative, nearest-neighbour pattern matcher.
- can accept both continuous valued and binary input.
- the perceptron learns offline, operates in discrete time and stores the pattern pairs (A_k, B_k), $k = 1, 2, \ldots, m$ using the **perceptron error-correction (or convergence) procedure**, where the k^{th} pattern pair is represented by the analog valued vector $A_k = (a_1^k, \ldots, a_n^k)$ and the bipolar $[-1, +1]$ valued vector $B_k = (b_1^k, \ldots, b_p^k)$.
- a perceptron decides whether an input belongs to one of two classes.
- the net divides the space spanned by the input into two regions separated by a hyperplane or a line in 2-D (a decision plane).

The Perceptron Convergence Procedure

Step 1. Initialize weights and thresholds.

- set the connection weights w_i and the threshold value θ to small random values.

Step 2. Present new input and desired output.

- present new continuous valued input $x_0, x_1, \ldots, x_{n-1}$ along with the desired output $d(t)$.

Step 3. Calculate actual output.

$$y(t) = f_n\left(\sum_{i=0}^{n-1} w_i(t)x_i(t) - \theta\right)$$

Step 4. Adapt weights.

- when an error occurs the connection weights are adapted by the formula:

$$w_i(t+1) = w_i(t) + \eta[d(t) - y(t)]x_i(t)$$

  where η is a positive gain fraction that ranges from 0.0 to 1.0 and controls the adaption rate.

Step 5. Repeat by going to Step 2.

The Gain Term (η)

- ranges from 0.0 to 1.0.
- controls the adaption rate.
- must be adjusted to satisfy the conflicting requirements of
  1. fast adaptation for real changes in the input distributions and
  2. averaging of past inputs to provide stable weight estimates.

Perceptron Convergence

- Rosenblatt proved that if the inputs presented from the two classes are separable then the perceptron convergence procedure converges and positions the decision hyperplane between those two classes in finite time.
- problem: decision boundaries may oscillate continuously when the inputs are not separable and distributions overlap.

Encoding

First Order Learning

1. Initial hyperplane placement
   - assign values in the range $[+1, -1]$ to all the F_A to F_B inter-layer connections, w_{ij}, and to each F_B PE threshold, θ_j.
2. Hyperplane adjustment
   - for each pattern pair (A_k, B_k) do
     (a) transfer A_k to the F_A PEs, filter the activations through W and calculate the new F_B activation values
     $$b_j = f(\sum_{i=1}^{n} w_{ij} a_i - \theta_j)$$
     where the bipolar step function, $f()$, is defined as
     $$f(x) = \begin{cases} +1 & \text{if } x > 0 \\ -1 & \text{otherwise} \end{cases}$$
     (b) compute the error between the computed and desired F_B values
     $$d_j = b_j^k - b_j$$
     (c) adjust the weights
     $$\Delta w_{ij} = \alpha a_i d_j$$
     where α is a positive constant controlling the learning rate (gain factor).
3. Repeat Step 2 until the error-correction value d_j is either sufficiently low or zero.

Higher Order Learning

- higher order correlations can be effectively added to the encoding algorithm.
- second order connections are stored in the n-by-n-by-p matrix V and adjusted by
$$\Delta v_{hij} = \alpha a_h^k a_i^k d_j$$
and
$$b_j = f(\sum_{i=1}^{n} w_{ij} a_i + \sum_{h=1}^{n} \sum_{i=1}^{n} v_{hij} a_h a_i - \theta_j)$$

Recall

- employs the feedforward equation
$$b_j = f(\sum_{i=1}^{n} w_{ij} a_i - \theta_j)$$
if only first order correlations are used and
$$b_j = f(\sum_{i=1}^{n} w_{ij} a_i + \sum_{h=1}^{n} \sum_{i=1}^{n} v_{hij} a_h a_i - \theta_j)$$
if both first and second order correlations are used.

The Least Mean Square Solution

- a modification to the perceptron convergence procedure.
- minimizes the mean square error between the desired output of a perceptron-like net and the actual output.
- the algorithm is called the **Widrow-Hoff** or **LMS** algorithm.
- the LMS algorithm is identical to the perceptron convergence procedure except that the hard limiting nonlinearity is made linear or replaced by a threshold-logic nonlinearity.
- weights are corrected on every trial by an amount that depends on the difference between the desired and the actual output.
- a classifier could use desired outputs of 1 for class A and 0 for class B.
- during operation, the input would then be assigned to class A only if the output was above 0.5.

The Gaussian Classifier

- maximum likelihood Gaussian classifiers assume inputs are uncorrelated and distributions for different classes differ only in mean values.
- this type of Gaussian classifier and the associated Euclidean distance metric are often used in speech recognition.
- the decision regions formed by perceptrons are similar to those formed by maximum likelihood Gaussian classifiers.
- the weights and threshold in a perceptron can be selected such that the perceptron structure computes the difference between log likelihoods required by such a Gaussian classifier.

A Gaussian Classifier Implemented Using the Perceptron

- Class A
  - m_{A_i} – the mean of input x_i when the input is from class A.
  - $\sigma^2_{A_i}$ – the variance of input x_i.
- Class B
  - m_{B_i} – the mean of input x_i when the input is from class B.
  - $\sigma^2_{B_i}$ – the variance of input x_i and
  - $\sigma^2_{A_i} = \sigma^2_{B_i}$
- the likelihood values required by a maximum likelihood calssifier are monotonically related to

$$L_A = \sum_{i=0}^{n-1} \frac{(x_i - m_{A_i})^2}{\sigma_i^2}$$

$$= -\sum \frac{x_i^2}{\sigma_i^2} + 2\sum \frac{m_{A_i} x_i}{\sigma_i^2} - \sum \frac{m_{A_i}^2}{\sigma_i^2}$$

and

$$L_B = -\sum \frac{x_i^2}{\sigma_i^2} + 2\sum \frac{m_{B_i} x_i}{\sigma_i^2} - \sum \frac{m_{B_i}^2}{\sigma_i^2}$$

[L_B = (**Term 1**) (**Term 2**) (**Term 3**)]

- a maximum likelihood classifier must calculate L_A and L_B and select the class with the highest likelihood.
- since **Term 1** is identical for L_A and L_B, it can be dropped.
- **Term 2** is a product of the input times the weights and can be calculated by a perceptron.
- **Term 3** is a constant which can be obtained from the threshold in a perceptron node.
- a Gaussian classifier for two classes can thus be formed by using the perceptron to calculate $L_A - L_B$ by setting

  $$w_i = \frac{2(m_{A_i} - m_{B_i})}{\sigma_i^2}$$

  and

  $$\theta = \sum_{i=0}^{n-1} \frac{m_{A_i}^2 - m_{B_i}^2}{\sigma_i^2}$$

- the perceptron structure can be used to implement
  1. classifiers which use the perceptron training algorithm
     - makes no assumptions concerning the shape of underlying distributions; instead it focuses on errors that occur when distributions overlap.
     - more robust than classical techniques.
     - works well when inputs are generated by nonlinear processes and are heavily skewed and non-Gaussian.
  2. Gaussian maximum likelihood classifier
     - makes strong assumptions concerning underlying distributions and is more appropriate when distributions are known and match the Gaussian assumption.
- neither is appropriate when classes cannot be separated by a hyperplane.

In Conclusion

- the perceptron is limited by its linear separability condition.
- the perceptron's strengths:
  - its well understood behaviour
  - its adequate storage capacity

    - its immediate recall.

- the perceptron's limitations:
    - poor at generalization
    - requires lengthy supervised offline learning
    - cannot encode nonlinearly separable classifications.
- the perceptron is ideally suited to pattern matching applications that are inherently linear and only require a two-class response.

Adaline and Madaline

The Adaptive Linear Element (Adaline)

- introduced by Widrow and Hoff in 1960.
- two-layer feedforward perceptron with n F_A PEs and **one** F_B PE.
- the Adaline is a combinatorial logic circuit that accepts several inputs and produces one output.
- similar to Grossberg's **instar** – the minimal structure capable of recognizing a spatial pattern.

The Multiple Adaline (Madaline)

- a configuration of several Adalines in a two-layer feedforward topology.
- a heteroassociative, nearest-neighbour pattern matcher that stores pattern pairs using the least mean square (LMS) error-correction encoding procedure.
- the input pattern is represented by the analog valued vector $A_k = (a_1^k, \ldots, a_n^k)$.
- the output is the bipolar $[-1, +1]$ valued vector $B_k = (b_1^k, \ldots, b_p^k)$.
- Madaline learns offline and operates in discrete time.

Encoding - Least Mean Square Algorithm

1. Assign random values in the range $[-1, +1]$ to all w_{ij} and each θ_j.
2. For each pattern pair (A_k, B_k) do
   (a) Input A_k, filter F_A activations through W and calculate the new F_B activations
   $$b_j = \sum_{i=1}^{n} w_{ij} a_i + \theta_j$$
   (b) Compute the error
   $$d_j = b_j^k - b_j$$
   (c) Adjust the weights
   $$\Delta w_{ij} = \alpha a_i d_j$$

   - This correction procedure does gradient descent on the n-dimensional mean square error surface and was found by calculating the gradient of error with respect to the weights.

3. Repeat Step 2 until the error correction is sufficiently low or zero.

Madaline Recall

- uses a thresholded version of the encoding equation that produces bipolar values

$$b_j = f(\sum_{i=1}^{n} w_{ij} a_i + \theta_j)$$

  where the bipolar step function $f()$ is defined as

$$f(x) = \begin{cases} +1 & \text{if } x > 0 \\ -1 & \text{otherwise} \end{cases}$$

Convergence

- the LMS encoding procedure is proven to converge to the global error minimum through an analysis of its mean squared error.

Comparison of the Perceptron and Madaline

- in the encoding procedure, they are identical except for
  - the sign of the threshold
  - the omission of the threshold function.
- Perceptron

$$b_j = f(\sum_{i=1}^{n} w_{ij} a_i - \theta_j)$$

- Madaline

$$b_j = \sum_{i=1}^{n} w_{ij} a_i + \theta_j$$

- they are each associated with different mathematical foundations
  - placement of a hyperplane
  - minimization of the mean squared error between desired and computed outputs.

In Conclusion

- Adaline/Madaline Strengths
  - its capacity to hold twice as many patterns as there are A_k dimensions ($m = 2n$).
  - its well understood mathematics.
- Adaline/Madaline Limitations

  - lengthy encoding time.
  - inability to learn online.
  - the restriction to linear (A_k, B_k) mappings.
- other notes
  - LMS algorithm is the most pervasive of all encoding algorithms.
  - **adaptive signal processing** is based largely on the LMS algorithm
    * applications: system modelling, statistical prediction, noise cancelling, adaptive echo cancelling, and channel equalization.

Adaptive Signal Processing

- digital signals generally originate from sampling continuous input signals by A-to-D conversion.
- filtering digital signals is often done by means of a tapped-delay-line or transversal filter.
- the sampled input signal is applied to a string of delay boxes which each delay the signal by one sampling period.
- an adaptive linear combiner (ALC) is connected to the taps between delay boxes – the filtered output is a linear combination of the current and past input signal samples.
- by varying the weights in the ALC's, the impulse response from input to output is directly controllable – the weights are usually adjusted to cause the output signal to be a best least squares match over time to the desired-response input signal.

Adaptive Filters

- the simplest, most robust and widely used filter is the adaptive digital filter adapted by the LMS algorithm.
- an adaptive threshold element is a key component in adaptive pattern recognition systems.
- it is composed of
  - an adaptive linear combiner cascaded with
  - a quantizer to produce a binary 'decision'.
- the adaptive threshold element is trainable and capable of implementing binary logic functions.
- the LMS algorithm was originally developed to train the adaptive threshold element (Adaline).
- Adaline
  - adaptive linear neuron.
  - an early neuronal model
    * adaptive weights were analogous to synapses.

        * input vector components related to dendritic inputs.
        * quantized output was analogous to the axonal output.
        * the output decision was determined by a weighted sum of the inputs, much the way real neurons were believed to behave.

Example – Noise Cancelling

- a common problem in signal processing is that of separating a signal from additive noise.
- a classical approach uses optimal Wiener or Kalman filtering – attempts to pass the signal s without distortion while stopping the noise n_0. In general, this cannot be done perfectly.
- another approach uses adaptive filtering – only viable when an additional “reference input” is available containing noise n_1 which is correlated with the original noise n_0.
- adjusting or adapting the filter to minimize the total output power is tantamount to causing the output ϵ to be a best least squares estimate of the signal s for the given structure and adjustability of the adaptive filter, and for the given reference input;
    - therefore, little or no prior knowledge of s, n_0 or n_1 or of their interrelationships is required.

Cancelling Maternal Heartbeat in Fetal Electrocardiography

- need to cancel interference from the mother's heart when attempting to record clear fetal electrocardiograms.
- the abdominal leads provide the primary input (containing fetal ECG and interfering maternal ECG signals).
- chest leads provide the reference input (containing pure interference, the maternal ECG).
- the maternal ECG was adaptively filtered and subtracted from the abdominal signal leaving the fetal ECG.
- fetal and maternal ECG signals have spectral overlap – the two hearts are electronically isolated and work independently but the second harmonic frequency of the maternal ECG is close to the fundamental of the fetal ECG.

Back-Propagation

Learning by Example

- in a multilayer net, the information coming to the input nodes could be recoded into an **internal representation** and the outputs would be generated by the internal representation rather than by the original pattern.
- input patterns can always be encoded, if there are enough hidden units, in a form so that the appropriate output pattern can be generated from any input pattern.

The Learning Scenerio

- the network is provided with a set of example input-output pairs (a training set).
- the connections are to be modified so as to approximate the function from which the input/output pairs have been drawn.
- the net is then tested for the ability to generalize.
- error correction learning procedure
  - during training an input is put into the net and flows through the net generating a set of values on the output units.
  - the actual output is compared with the desired target and a match is computed.
    * if there is a match, no change is made to the net,
    * otherwise a change must be made to some of the connections.
  - the problem is to determine which connections in the entire network were at fault for the error – **credit assignment problem**.

Back to Rosenblatt's Perceptron

- the series-coupled perceptron
  - randomly preset feedforward connections between F_A and F_B.
  - adaptable connections between F_B and F_C.
- about the series-coupled perceptron from Rosenblatt's *The Principles of Neurodynamics* (1962):

> The procedure to be described here is called the 'back-propagating error correction procedure' since it takes its cue from the error of the R-units, propagating corrections back towards the sensory end of the network if it fails to make a satisfactory correction quickly at the response end. The actual correction procedure for the connections to a given unit, whether it is an A-unit or an R-unit, is perfectly identical to the correction procedure employed for an elementary perceptron, based on the error-indication assigned to the terminal unit.

- but Minsky and Papert in *Perceptrons* (1969) on multilayer machines:

> The perceptron has shown itself worthy of study despite (and even because of!) its severe limitations. It has many features that attract attention: its linearity; its intriguing learning theorem; its clear paradigmatic simplicity as a kind of parallel computation. There is no reason to suppose that any of these virtues carry over to the many-layered version. Nevertheless, we consider it to be an important research problem to elucidate (or reject) our intuitive judgement that the extension is sterile. Perhaps some powerful convergence theorem will be discovered, or some profound reason for the failure to produce an interesting "learning theorem" for the multilayered machine will be found.

A Brief History of Back-Propagation

- error back-propagation through nonlinear systems has existed in *variational calculus* for many years – more recently these methods have been widely used in *optimal control.*
- first gradient descent approach to training multilayered nets was by Amari in 1967 – used a single hidden layer PE to perform nonlinear classification.
- in 1974, Werbos discovered "dynamic feedback" in *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences* (P. Werbos, PhD thesis, Harvard University, August 1974).
- in 1982, Parker talked about "learning logic" and this appears again in an 1985 MIT report. Reference: *Learning-Logic* (Invention Report S81-64, File 1, Office of Technology Licensing, Stanford University, October 1982).
- in 1985, Rumelhart, Hinton and Williams published their version – most effect on dissemination of the algorithm in the neural net community. *Learning internal representations by error propagation*, ICS Report 8506, Institute for Cognitive Science, University of California at San Diego, September 1985.

The Need for Internal Representation

- a constraint on two-layer networks: similar inputs lead to similar outputs.
- problem: the case of XOR.

INPUT PATTERNS		OUTPUT PATTERNS
00	$\rightarrow$	0
01	$\rightarrow$	1
10	$\rightarrow$	1
11	$\rightarrow$	0

- Note: a two-layer net could solve:

INPUT PATTERNS		OUTPUT PATTERNS
000	$\rightarrow$	0
010	$\rightarrow$	1
100	$\rightarrow$	1
111	$\rightarrow$	0

- Problem:
  - there is a guaranteed learning rule for all problems that can be solved without hidden units but there is no equally powerful rule for nets with them.
- Three Responses:
  1. Competitive Learning
     - unsupervised learning rules are employed so that useful hidden units develop, but no external force is there to ensure that appropriate hidden units develop.
  2. Assume an internal representation that seems reasonable, usually based on a priori knowledge of the problem.
  3. Develop a learning procedure capable of learning an internal representation adequate for the task.

Back-Propagation

- three-layer perceptron with feedforward connections from the F_A PEs (input layer) to the F_B PEs (hidden layer) and feedforward connections from the F_B PEs to the F_C PEs (output layer).
- heteroassociative, function-estimating ANS that stores arbitrary analog spatial pattern pairs (A_k, C_k) using a multilayer gradient descent error-correction encoding algorithm.
- learns offline and operates in discrete time.

Encoding

- performs the input to output mapping by minimizing a *cost function.*
- cost function minimization
  - weight connection adjustments according to the error between computed and desired output (F_C) PE values.
  - usually it is the *squared error*

    * sqaured difference between computed and desired output values for each F_C PE across all patterns in the data set.
  - other cost functions
    1. entropic cost function
       * White (1988) and Baum & Wilczek (1988).
    2. linear error
       * Alystyne (1988).
    3. Minkowski-r back-propagation
       * r^{th} power of the absolute value of the error.
       * Hanson & Burr (1988).
       * Alystyne (1988).
  - weight adjustment procedure is derived by computing the change in the cost function with respect to the change in each weight.
  - this derivation is extended to find the equation for adapting the connections between the F_A and F_B layers
    * each F_B PE's error is a proportionally weighted sum of the errors produced at the F_C layer.
  - the basic *vanilla version back-propagation* algorithm minimizes the squared error cost function and uses the three-layer elementary back-propagation topology. Also known as the *generalized delta rule.*

Vanilla Back-Propagation Encoding Algorithm

1. Assign random values in the range $[+1, -1]$ to all F_A to F_B connections, v_{hi}, all F_B to F_C connections, w_{ij}, to each F_B PE threshold θ_i and to each F_C PE threshold, $, _j$.

2. For each pattern pair (A_k, C_k) do:

   (a) Process A_k's values to calculate new F_B PE activations using

   $$b_i = f(\sum_{h=1}^{n} a_h v_{hi} + \theta_i)$$

   where $f()$ is the logistic sigmoid threshold function

   $$f(x) = (1 + e^{-x})^{-1}$$

   (b) Filter the F_B activations through W to F_C using

   $$c_j = f(\sum_{i=1}^{n} b_i w_{ij} + , _j)$$

   (c) Compute the error between computed and desired F_C PE values using

   $$d_j = c_j(1 - c_j)(c_j^k - c_j)$$

(d) Calculate the error of each F_B PE relative to each d_j with

$$e_i = b_i(1 - b_i) \sum_{j=1}^{q} w_{ij} d_j$$

(e) Adjust the F_B to F_C connections

$$\Delta w_{ij} = \alpha b_i d_j$$

where Δw_{ij} is the amount of change made to the connection from the i^{th} F_B to the j^{th} F_C and α is a positive constant controlling the learning rate.

(f) Adjust the F_C thresholds

$$\Delta,_{j} = \alpha d_j$$

(g) Adjust the F_A to F_B connections

$$\Delta v_{hi} = \beta a_h e_i$$

where β is a positive constant controlling the learning rate.

(h) Adjust the F_B thresholds

$$\Delta \theta_i = \beta e_i$$

3. Repeat step 2 until d_j's are all either zero or sufficiently low.

The Recall Mechanism

- consists of two feedforward operations

  1. create F_B PE values

  $$b_i = f(\sum_{h=1}^{n} a_h v_{hi} + \theta_i)$$

  2. after all F_B PE activations have been calculated they are used to create new F_C PE values

  $$c_j = f(\sum_{i=1}^{n} b_i w_{ij} + ,_{j})$$

Convergence

- BP is not guaranteed to find the global error minimum during training, only the local error minimum.

  - simple gradient descent is very slow because we have information only about one point and no clear picture of how the surface may curve

    * small steps take forever
    * big steps cause divergent oscillations across "ravines".

- factors to consider

  1. number of hidden layer PEs

2. size of the learning rate parameters
3. amount of data necessary to create the proper mapping.

- a three-layer ANS using BP can approximate a wide range of functions to any desired degree of accuracy
  - if a mapping exists then BP can find it.

More on Back-Propagation

Advantages

- its ability to store many more patterns than the number of F_A dimensions ($m >> n$).
- its ability to acquire arbitrarily complex nonlinear mappings.

Limitations

- extremely long training time.
- offline encoding requirement.
- inability to know how to precisely generate any arbitrary mapping procedure.

Areas of Research and Analysis

1. optimizing the number of PEs in the hidden layer(s).
2. improving the rate of learning.
3. extending the topology to any arbitrary connection topology.
4. analyzing the scaling, generalization and fault tolerance properties.
5. employing higher-order correlations and arbitrary threshold functions.

Hidden Layer Optimization

- added cost functions that will minimize the number of connections and the number of hidden units as well as the overall squared error during training
  - Rumelhart, 1988
  - training time is greatly increased.
- heuristics for pruning away hidden units during training
  - Sietsma and Dow, 1988
  - Yu and Teh, 1988.
- 'dynamic node creation' technique grows hidden layer PEs as they are needed
  - Ash, 1988.

Learning Rate Improvement

1. Newton's Method
    - make precise weight changes without having to decide how large the learning parameters should be during encoding.
    - gives the gradient that will move from the current point on a curve (the error surface) in a straight line to the intersection on the curve.
    - the product of the gradient and the pseudoinverse of the Hessian gives the vector that intersects the curve.
    - can eliminate some of the local minima traps and facilitate a more efficient adaptation process but it is difficult and costly to compute.
2. Momentum
    - one way to increase the learning rate without leading to oscillation is to modify the generalized delta rule to include a *momentum* term.
    - $\Delta w_{ij} = \alpha b_i d_j$ changes to $\Delta w_{ij}(n+1) = \alpha b_i d_j + \eta \Delta w_{ij}(n)$ where n indexes the presentation number and η is a constant which determines the effect of past weight changes on the current direction of movement in the weight space.
    - this provides a kind of momentum in weight space that effectively filters out high-frequency variations of the error surface in the weight space.
    - useful in spaces with long ravines characterized by sharp curvature across the ravine and a gently sloping floor
        - sharp curvature tends to cause divergent oscillations across the ravine – to avoid this, small steps are necessary but this is too slow.
        - momentum filters out the high curvature and allows the effective weight steps to be bigger.
3. Other strategies
    - using 10 to 30 times the size of the calculated gradient (weight change) to speed the learning rate.
    - employment of line search routines on the calculated gradient in order to make the optimal size change during each training iteration.
    - performing a priori weight and threshold computations before applying the gradient descent training procedure.
    - sets of heuristics for escaping local minima.
    - addition of noise – decreased training time for small problems.
    - the use of shaping schedules – a training procedure that starts with the easier-to-learn mappings and gradually works to the more difficult.
    - constraining the weights during training to prevent oscillations.

Alternate Connection Topologies

- the most general BP topolgy possible is one that allows any PE to connect to any other PE.
- these topologies introduce feedback into the recall dynamics and are called *recurrent back-propagation* artificial neural networks.

Scaling, Generalization and Fault-Tolerance

- BP does not appear to scale well – large nets may have to be partitioned into separate modules that can be trained independently.
- fault-tolerance: the ability to withstand damage to connections and nodes and still process properly.

Higher Order Correlations and Alternate Threshold Functions

- Rumelhart, Hinton and Williams (1986) have proposed the use of higher order correlations
  - the PEs using higher order correlations are called *sigma-pi* units.
- other threshold functions
  - sine and cosine functions applied to the input layer PEs and used to augment the input layer.
  - clipped cosine function – BP was able to construct a Fourier series approximation to a given function as its output.

A Note About Symmetry

- if all the weights start out with equal values and the solution requires unequal weights to be developed, the system can never learn.
  - because error is proportional to the value of the weights.
  - the system is starting out at a local *maximum* (of the error function) which keeps the weights equal but once escaped it will never return there.
- solution: start the system with small random weights.

Examples

Once Again ... the XOR Problem

- one solution to the XOR problem developed by back-propagation:

From		To	Weight
$InputUnit_1$	$\rightarrow$	$HiddenUnit$	-6.4
$InputUnit_2$	$\rightarrow$	$HiddenUnit$	-6.4
$InputUnit_1$	$\rightarrow$	$OutputUnit$	-4.2
$InputUnit_2$	$\rightarrow$	$OutputUnit$	-4.2
$HiddenUnit$	$\rightarrow$	$OutputUnit$	-9.4

- this solution was reached after 558 sweeps through the four stimulus pairs with a learning rate $\alpha = 0.5$.
- another possible architecture: 2 hidden units and no direct connections from input to output. This topology can encounter local minima and thus not be able to solve the problem.

From		To	Weight
$InputUnit_1$	$\rightarrow$	$HiddenUnit_1$	-2.0
$InputUnit_1$	$\rightarrow$	$HiddenUnit_2$	4.3
$InputUnit_2$	$\rightarrow$	$HiddenUnit_1$	9.2
$InputUnit_2$	$\rightarrow$	$HiddenUnit_2$	8.8
$HiddenUnit_1$	$\rightarrow$	$OutputUnit$	-4.5
$HiddenUnit_2$	$\rightarrow$	$OutputUnit$	5.3

with the thresholds being: $HiddenUnit_1 = 2.0$, $HiddenUnit_2 = -0.1$ and $OutputUnit = -0.8$.

- this configuration was reached after 6,587 presentations of each pattern with $\alpha = 0.25$.
- with this configuration, two of the inputs (00 and 10) produced correct responses but $01 \rightarrow 0.5$ and $11 \rightarrow 0.5$.

Parity

- the output required is 1 if the input pattern contains an odd number of 1s and 0 otherwise.
- difficult problem because the most similar patterns require different answers.
- basic BP topology – at least N hidden units are required to solve parity with patterns of length N.
- "solution": N input units fully connected to N hidden units by connections with the weight of $+1$. The hidden units are connected to a single output unit – the weights on these connections alternate between $+1$ and -1. The thresholds on the hidden units (in order from the left-most unit): $-0.5, -1.5, -2.5, \ldots, 0.5 - n$ where n is the number of the hidden unit. The threshold on the output unit is -0.5.
- the hidden units "count" the number of inputs.
- the first m hidden units come on whenever m bits are on in the input pattern.
- the hidden units connect with alternately positive and negative weights to the output unit, therefore, the net output is zero for even numbers and $+1$ for odd numbers.
- for the problem of 4-bit parity:
  - topology: 4 inputs, 4 hidden units.
  - presentations: 2,825 of each of the 16 patterns.
  - learning rate: 0.5.
- the internal representation: the number of hidden units that come on is equal to the number of zeros in the input and is not dependent on which input units are on.

Symmetry

- classifying input strings as to whether or not they are symmetric about their centre.
- the problem can always be solved with only two hidden units, for example:

From		To	Weight
$InputUnit_1$	$\rightarrow$	$HiddenUnit_1$	-3.18
$InputUnit_1$	$\rightarrow$	$HiddenUnit_2$	3.16
$InputUnit_2$	$\rightarrow$	$HiddenUnit_1$	6.32
$InputUnit_2$	$\rightarrow$	$HiddenUnit_2$	-6.34
$InputUnit_3$	$\rightarrow$	$HiddenUnit_1$	-12.56
$InputUnit_3$	$\rightarrow$	$HiddenUnit_2$	12.52
$InputUnit_4$	$\rightarrow$	$HiddenUnit_1$	12.56
$InputUnit_4$	$\rightarrow$	$HiddenUnit_2$	-12.51
$InputUnit_5$	$\rightarrow$	$HiddenUnit_1$	-6.33
$InputUnit_5$	$\rightarrow$	$HiddenUnit_2$	6.31
$InputUnit_6$	$\rightarrow$	$HiddenUnit_1$	3.17
$InputUnit_6$	$\rightarrow$	$HiddenUnit_2$	-3.17
$HiddenUnit_1$	$\rightarrow$	$OutputUnit$	-9.44
$HiddenUnit_2$	$\rightarrow$	$OutputUnit$	-9.44

  with the thresholds on the hidden units being -1.0 and on the output unit it is 6.89.

- the solution for length size was arrived at after 1,208 presentations of each 6-bit pattern with $\alpha = 0.1$.
- for a given hidden unit, weights that are symmetric about the middle are equal in magnitude and opposite in sign.
- if a symmetric pattern is on, both hidden units will receive a net input of zero from the input units and therefore both will be off and the output unit will be on.
- the weights on each side of the midpoint are in the ratio 1:2:4.
- this ensures that each of the 8 patterns that can occur on each side of the midpoint sends a unique activation sum to the hidden unit – assures that there is no pattern on the left that will balance a non-mirror-image pattern on the right.
- the two hidden units have identical patterns of weights from the input units except for sign.
- this ensures that for every nonsymmetric pattern, at least one of the two hidden units will come on and turn off the output unit.
- in summary, the net is arranged so that both hidden units will receive exactly zero activation from the input units when the pattern is symmetric and at least one of them will receive positive input for every nonsymmetric pattern.

More Examples

- BP is capable of learning many different kinds of interesting representations in the hidden units.
    1. learning optimal codes for squeezing information through narrow bandwidth channels.
    2. learning sets of optimal filters for discriminating between very noisy signals.
    3. learning distributed representations of concepts (G. Hinton, 1986).
        - learn the semantic constraints that underlie a set of facts.
        - a 5-layer network was trained to generalize in a set of 100 triples such as *(Victoria has_father John)* derived from 2 family trees.
        - input vector represented the first 2 terms and the required output represented the third term.
        - after extensive training the net could generalize appropriately to triples on which it had not been trained like *(Victoria has_mother ??)*.
        - by recording the sets of triples for which each of the hidden units become active, it is possible to show that these units had learned to represent properties that were never mentioned in the input or output.
            - BP does gradient descent in the space of possible representations – hidden features and their interactions encode the underlying regularities of the domain.

NETtalk: A Parallel Network that Learns to Read Aloud

- T.J. Sejnowski and C.R. Rosenberg, 1986.
- English text is converted to speech by training by exposure to examples as opposed to entering phonological rules and handling exceptions with a look-up table.
    - a BP net is trained to transform an input vector that encodes a sequence of letters into an output vector that encodes the phonemic features of the sequence and which can be used to drive a speech synthesizer.
- NETtalk is a two-layer feedforward sigmoid network with 80 Adalines in the first layer and 26 Adalines in the second.
- pronunciation of most words follows general rules based upon spelling and word context, but there are many exceptionsand special cases.
- NETtalk learns to pronounce words in stages suggestive of the learning process in children.
    - the system makes babbling noises during the early stages of the training.
    - as the net learns, it next conquers the general rules and tends to make a lot of errors by using the rules even when not appropriate.
    - as training continues, the net abstracts the exceptions and special cases and produces intelligible speech with few errors.
- NETtalk operation
    - input is a vector of seven characters (including spaces) from a transcript of text.

  - output is phonetic information corresponding to the pronunciation of the centre (fourth) character in the seven-character input field – the other six characters provide context.
  - to read text, the 7-character window is scanned across a document and the net generates a sequence of phonetic symbols that can be used to control a speech synthesizer.
  - each of the 7 characters at the net's input is a 29-component binary vector, with each component representing a different alphabetic character or punctuation mark (a 1 is placed in the appropriate component and all others are set to 0).
  - the system's 26 outputs correspond to 23 articulatory features and 3 additional features which encode stress and syllable boundaries.
  - when training, the desired response vector has zeros in all components except those which correspond to the phonetic features associated with the centre character in the input field.

- NETtalk scanned a 1024-word transcript of phonetically transcribed continuous speech.
- with presentation of each window, the system's weights were trained by back-propagation.
- after roughly 50 presentations of the entire training set, the net was able to produce accurate speech from data the network had not been exposed to during training.

Back-Propagation Supplement

The Role of Hidden Layers

- **remap** the inputs and results of previous layers to achieve a more **separable** or **classifiable** representation of the data.
- may allow attachment of **semantics** to certain combinations of layer inputs.

Training Considerations

- training by pattern or epoch
- use of momentum
- sequential versus random ordering of the training vectors
- determining the occurrence of local minima
- choosing 'suitable' unit biases
- designing appropriate initial conditions on biases, weights, etc.

Bias

- it may be advantageous to modify the feedforward NN to include a **bias**.
- for the sigmoidal activation function

  $$f(net_j) = \frac{1}{1 + e^{-net_j}}$$

  we observe that

  $$0 \leq f(net_j) \leq 1$$

  and with no activation

  $$net_i = 0$$
  $$f(0) = \frac{1}{2}$$

- we may wish to bias this unit such that $f(0)$ has another value.
- a simple model for the unit with bias is to modify net_j such that

  $$net_j = \sum_i w_{ij} a_i + bias_j$$

Training by Epoch

- if we correct the network weights for each training pair for all connections then we are training **by sample**.
- in training **by epoch**, we would calculate the weight change by

$$\Delta \bar{w_{ij}} = \sum_p \Delta^p w_{ij}$$

- this represents an overall or accumulated correction to the weight set after each training **epoch**.

Hints for Using Back Propagation

- try different **threshold functions**
  - convergence is usually faster with sigmoid fucntions that range between -1 and $+1$ (hyperbolic tangent) instead of 0 to 1 (logistic).
- use **target** values within the sigmoid range instead of the asymptotes
  - use -0.7 and $+0.7$ instead of -1 and $+1$
  - the asymptotes tend to take a long time to train, worsen generalization and drive the weights to very large values.
- use reasonable **initial weights**
  - to keep the weighted sums within the range of the sigmoid,
    * for PEs with many input connections: small weights
    * for PEs with few input connections: larger weights.
  - if the initial weights are too small then the gradient will be small and training will begin slowly.
  - if the initial weights are too large then PEs may get **saturated** and again the gradients will be very small.

Some Guidelines for Hidden Units

- **E.B. Baum and D. Haussler**
  - "What size net gives valid generalization?", in Neural Computation, 1:151-160, 1989.
- Feedforward neural networks tend to **generalize** if
  1. The fraction of errors committed against the training set is $< \frac{\epsilon}{2}$ where ϵ is the allowed fraction of errors on the test set (assuming that $\epsilon < \frac{1}{8}$)
  2. $m > O(\frac{W}{\epsilon} log \frac{N}{\epsilon})$ where
     - N is the number of hidden units (one layer)
     - m is the number of patterns in the training set
     - W is the number of weights

Baum and Haussler on Hidden Units

- in practice, m patterns of training data are need for generalization where $m > \frac{W}{\epsilon}$ and
  - m is the number of patterns in the training set
  - W is the number of weights
  - ϵ is the allowed fraction of errors on the test set (assuming that $\epsilon < \frac{1}{8}$)

Appropriate Uses for Back Propagation

- applications with high-dimensional input vectors
- relatively sparse data
- noisy data with high within-class variability
- applications where training speed is not important

Non-Appropriate Uses for Back Propagation

- for very dense data in a low dimensional space (statistical methods are better)
- many classes with low variability (pattern matching is better)
- applications that need "explanations" with the answers

Detection of Explosives in Checked Airline Baggage

The System

- a system by Science Applications International Corporation (SAIC) has been developed for the FAA to detect explosives hidden in checked airline baggage (1985-1987).
- an artificial neural system has been applied to the problem of discriminating between suitcases with and without explosives.
- the performance of the ANS exceeded the performance of a standard statistical technique (discriminant analysis) over a substantial range.
- the input to the ANS was data gathered during field tests of a prototype explosive detection system.
- the detection system uses thermal neutron activation (TNA) to detect the presence of explosives.
- a suitcase on a conveyor belt moves past a source and an array of detectors.
- neutrons from the source penetrate the luggage and are absorbed by all of the materials present.
- different elements will emit different gamma radiation after absorbing these neutrons.

- the gamma rays have sufficient energy to penetrate the baggage and are then caught in the detector array which records the number of gamma rays observed at each energy level.
- in theory, the amount of each element and its location can be determined from the array of signals.

System Constraints

- must process 10 bags per minute continuously
- must not damage film or magnetic recording media
- must be reliable
- should be built from commercially available components wherever possible (delivery speed and cost)
- the output of the system (whether the baggage is a threat) must be signaled by the time the bag exits the system.

The Data

- commercial and military explosives have several characteristics which distinquish them from most objects in luggage, e.g. high density of nitrogen.
- the goal was to set the detection rate at a high value (90%) and then achieve the lowest false alarm rate practical.
- test data:
  - 3 tests of large numbers of passenger bags
  - simulated explosives (same instrument responses) were added to random bags.

Key System Performance Parameters

1. **Probability of Detection (PD)** for the minumum amount of explosives in a threat.
   - determined by the number of bags with threats that cause an alarm divided by the total number of bags with threats (modified slightly by a weighting factor for frequency of explosive type).
2. **Probability of False Alarms (PFA)** on bags without threats.
   - measured by the number of bags without threats which cause the system to alarm divided by the total number of bags without threats.

The Analysis by Neural Network

- a “feature” is any combination of signals from the detectors in the array.
- the object of feature analysis is to develop a set of features with different amounts of noise resistance which still does a suitable job of detecting regions of high nitrogen density.
- back propagation technology was used to analyse the data.
- the pattern recognition module is implemented by a three-layer, fully connected, feedforward BP neural network.
- the first layer receives a set of precomputed feature vectors from the detection system which are normalized to grey scale values between -0.5 and $+0.5$
- the hidden layer learns to encode features that are not explicitly present in the input patterns.
- the output layer produces a grey scale value which can be thresholded to obtain a decision
  - $+0.5$ if the baggage contains a threat
  - -0.5 if it does not contain a threat.
- the number of connection weights is about the same as the number of parameters in standard decision analysis.
- the data set was presented to the network about 2,000 times during training and relatively low learning rates were used.
- to measure performance, the PD and PFA rates were measured on a test set.
- the ANS outperformed the discrimant analysis for false alarms rates greater than 1.75%.
- at 95% detection – the ANS can reduce the false alarm rate by 1.5% to 2% – substantially fewer bags need to be scanned by secondary techniques.
- another advantage of the ANS is that it needs very little human supervision
  - the calibration of the discriminant analysis took several days of statistical analysis.
- the results took about 2 full days of unsupervised computer time on an IBM AT.
- a problem with the ANS model is that its peformance at low false alarm rates (less than 1.75%) is substantially lower than for the discriminant analysis
  - could be caused by the activiation (threshold) function used – the sigmoid function is steep in the middle and therefore clustering occurs around the extremes (-0.5 and $+0.5$)
  - a smoother function may solve some of these problems.
- another problem is that the output does not contain a “sureness” for the decision – due to the encoding of the threat values during training (no a priori information on how threatening a bag should be).
- other results:

- testing was done on raw detector data instead of precomputed features. This expanded the net to 200 input PEs from 20 and took much longer to train. It also appeared that two hidden layers were more appropriate for this data.
- **Counter Propagation** was also tested – it is faster in reaching its saturation point (no further improvement in performance) but this performance was not as good as for the BP net.