Discriminant Analysis

Topics:

- Linear discriminants
- Fisher's discriminant
- Mean square error discriminant
- Multiple discriminant analysis
- Bayesian discrimination
- Maximum likelihood discrimination
- k-Nearest neighbours
- Perceptron
- Multilayer perceptron

Basic Concepts

- Training set, test set.
- Test set is also known as: unseen, or out of sample.
- OC or ROC diagram for 2-class problems. (OC = operating characteristic; ROC = receiver operating characteristic.)
- Detection, false alarm.
- Multiple class problems.

Discriminant Analysis: Objectives and Properties

- Assess the adequacy of a classification, given the group memberships.
- Assign objects to one of a number of (known) groups of objects.
- Note: supervised classification (= discriminant analysis) vs. unsupervised classification (= cluster analysis). Sometimes, along these lines, classification is distinguished from clustering.
- Remark: discriminant analysis is “discrete prediction”, whereas regression analysis is “continuous prediction”.
- So a multilayer perceptron neural network used for classification purposes is discriminant analysis; but used for the purposes of continuous mapping it is nonlinear regression.

Practical Remarks

- We can evaluate *error rates* by means of a training sample (to construct the discrimination surface) and a test sample.
- An optimistic error rate is obtained by reclassifying the design set: this is known as the *apparent error rate.*
- If an independent test sample is used for classifying, we arrive at the *true error rate.*
- The *leaving one out* method attempts to use as much of the data as possible: for every subset of $n - 1$ objects from the given n objects, a classifier is designed, and the object omitted is assigned. This leads to the overhead of n discriminant analyses, and to n tests from which an error rate can be derived.
- Another approach to appraising the results of a discriminant analysis is to determine a *confusion matrix* which is a contingency table (a table of frequencies of co–occurrence) crossing the known groups with the obtained

groups.

- We may improve our discrimination by implementing a *reject option*: if for instance we find $P(\mathbf{x} \mid c) > P(\mathbf{x} \mid y)$ for all groups $y \neq c$, we may additionally require that $P(\mathbf{x} \mid c)$ be greater than some threshold for assignment of $\mathbf{x}$ to c. Such an approach will of course help to improve the error rate.

- Neyman-Pearson criterion: let p_d be the probability of detection (i.e. finding correct class), and let p_f be the probability of a false alarm. Then the Neyman-Pearson criterion is to choose the maximum p_d subject to $p_f = \alpha$ where $0 \leq \alpha \leq 1$.

- For alternative decision strategies, see J.L. Melsa and D.L. Cohn, Decision and Estimation Theory, McGraw-Hill, New York, 1978.

Statistical Pattern Recognition

```
            +-----------------+          +---------------+
            |                 |          |               |
Observation | Feature         | Feature  |  Classifier   |Decision
----------->| Extraction      |--------->|               |-------->
x (Vector)  |                 | Vector y |               |Vector w
            |                 |          |               |
            +-----------------+          +---------------+
```

Linear Discriminant – 1

Simplest possible transformation: linear discriminant.

$$y = a'x$$

i.e. x is transformed to a single scalar. Classification is then a matter of thresholding.

Restricting attention to the two-class case:

$$\begin{array}{llll} \text{y} & > & T & \Longrightarrow & \text{class 1} \\ & < & T & \Longrightarrow & \text{class 2} \\ & = & T & \Longrightarrow & \text{class chosen arbitrarily} \end{array}$$

Linear Discriminant – 2

(Simple) perceptron with a step transfer function

```
                                             T
              x0                             |
               \                             |
                \a1                          v
                 \                      +----+----+
                  \                     | 1|   +--|
                   +--+--+              |  |   |  |  output 1 => class 1
      x1    a2     |     |    y         | 0|   |  |         0 => class 2
      ------------+    +--------------->+--+---+--+-------->
            .      |     |              |  |   T  |
            .     /+--+--+              +---------+
                 /
                /ap-1       y = sum of        y>T?  output = 1 if y>T
               /            ai.xi for i=1                  = 0 otherwise
             xp-1           to i=p
                            (Weighted sum
                             of inputs)
```

Discriminant Function

A general discriminant function, g, is a function of the form:

$$y = g(x)$$

$$\begin{array}{lcccl} \mathrm{y} & > & T & \Longrightarrow & \text{class 1} \\ & < & T & \Longrightarrow & \text{class 2} \\ & = & T & \Longrightarrow & \text{class chosen arbitrarily} \end{array}$$

Examples of g: linear function given by scalar product; or non-linear function giving a multilayer preceptron.

Linear Discriminant as Projection – 1

```
      |       1        2 2 2 2
y1    |      1 1 1       2 2 2 2 2
      |    1 1 1 1 1 1   2 2 2 2 2
      |    1 1 1 1 1 1 1  2 2 2 2 2
      |    1 1 1 1 1 1          2 2
      |      1 1 1 1         class w2
      |          1
      |  class w1
      +--------------+---+--------------------------->
                       t1    t2                           y0
```

```
Feature space, two-class, two dimensions
```

Linear Discriminant as Projection – 2

- Simple linear discriminant, i.e. $y = a'x$
- Projecting the data onto the y_0 axis corresponds to a discriminant vector $a = (a_1, a_2)' = (1.0, 0.0)'$, i.e. y_0 is given a weight of 1.0, y_1 is given zero weight.
- This projection would result in a little overlap of the classes.
- Projecting data onto axis y_1, discriminant vector = (0.0,1.0), would result in much overlap.
- However, projecting the vectors onto the line shown below would be close to optimal – no class overlap.

Linear Discriminant as Projection – 3

```
                             *
      |       1        2 2 * 2
y1    |      1 1 1       * 2 2 2 2
      |    1 1 1 1 1 1*  2 2 2 2 2
      |    1 1 1 1 1* 1 1  2 2 2 2 2
      |    1 1 1 * 1 1          2 2
      |      1*1 1 1                  class w2
      |    *     1
      | *           class w1
      +-----------------------------------------------> 
                                                          y0

      * * * projection line
```

Fisher's Linear Discriminant

- Project the data onto a single axis, defined by the Fisher discriminant vector a
- $y = a'x$
- Simultaneously optimize two criteria:
- maximum between-class separation, expressed as separation of the class means m_1, m_2, and
- minimum within-class scatter, expressed as the within class variances, v_1, v_2
- Fisher criterion combines these two optimands as:

$$J = (m_1 - m_2)/(v_1 + v_2)$$

where the transformed means and variances are:
$m_j = a'm_j, v_j = aS_ja', S_j =$ covariance for class j, $m_j =$ mean vector for class j.

- The discriminant is computed using:

$$a = W^{-1}(m_1 - m_2)$$

where W is the pooled (overall, class-independent) covariance matrix, $W = p_1 S_1 + p_2 S_2$. p_1, p_2 are the prior probabilities.

The discriminant is the line joining m_1 and m_2, with the "correction factor", W^{-1}, which "corrects" for overall spread of points. ("Pooled": total, all points contributing.)

Fisher's Linear Discriminant: Algorithm

1. Estimate class means m_j and covariance matrices S_j, and prior probabilities, p_j.
2. Compute pooled covariance matrix, W (see equation above).
3. Invert matrix W (using some standard matrix inversion procedure).
4. Compute the discriminant vector, a (see equation above).
5. Apply the discriminant using equation $y = a'x$.

Summary

1. weighted sum of inputs
2. scalar (inner) product of vectors
3. projection
4. thresholding
5. perceptron
6. Fisher’s linear discrimant analysis

Mean Square Error Discriminant

- Fisher discriminant works only for two classes although it is possible to tackle multi-class problems using pairwise discriminants for each of the $\left(\begin{array}{c} c \\ 2 \end{array} \right)$ dichotomies.
- The class vector is a binary vector, with only one bit set at any time, in which a bit j set (= 1) denotes class j; i.e. $y = (1, 0, 0, ...0)'$ denotes class 0 $y = (0, 1, 0, ...0)'$ denotes class 1 etc.
-

```
                 +------------------+       y class vector
                 |                  +--->      class 0
      Observ-    | Feature          +--->      class 1
      ---------> | Extraction /     +--->      class j
      -ation x   | Classification   +--->
      Vector     |                  +--->      class c-1
                 +------------------+
```

- For, initially, just one component of y, y_j, the regression problem can be expressed compactly as:

  $y_{ji} = x_i' b + e_i$ where $b = (b_0, b_1, \ldots, b_{p-1})'$ is a $p \times 1$ vector of coefficients for class j, $x_i = (1, x_{i1}, \ldots x_{ip-1})'$ is the pattern vector, and e_i = error, $y_{ji} = 1$ if the pattern x belongs to class j, $= 0$ otherwise.

- Formulation in terms of the augmented vector x, which contains the bias element 1 is important; without it we would effectively be fitting a straight line through the origin – the bias (b_0) corresponds to a non-zero intercept of the y-axis; compared to using a separate bias element, the analysis is greatly simplified.

- The complete set of n observation equations can be expressed as:

$$y = Xb + e$$

  where $e = (e_1, e_2, \ldots e_i, \ldots e_n)'$, and $y = (y_1, y_2, \ldots y_i, \ldots y_n)'$, the $n \times 1$ vector of observations of the class variable (bit). X is the $n \times p$ matrix formed by n rows of p pattern components.

- The least square error fitting is given by:

$$b' = (X'X)^{-1}X'y$$

- Note: the jkth element of the $p \times p$ matrix $X'X$ is $\sum_i x_{ji}x_{ik}$, and the jth row of the $p \times 1$ vector $X'y$ is $\sum_i x_{ij}y_i$.
- Thus, $X'X$ differs from the autocorrelation matrix of x only by a multiplicative factor, n, so that the major requirement for the least square error fitting equation above to provide a valid result is that the autocorrelation matrix of x is non-singular.
- We can express the complete problem, where the vector y has c components by replacing the vector y in the least square error fitting equation with the matrix Y, the $n \times c$ matrix formed by n rows of c observations.
- Thus, the least square error fitting equation extends to the complete least square

error linear discriminant:

$$B' = (X'X)^{-1}X'Y$$

$X'Y$ is now a $p \times c$ matrix, and B' is a $p \times c$ matrix of parameters, i.e. one column of p parameters for each dependent variable.

- Applying the discriminant/transformation is simply a matter of premultiplying the (augmented) vector x by B':

$$y' = B'x$$

Multiple Discriminant Analysis

- Also a generalization of Fisher's linear discriminant analysis.
- Also termed Discriminant Factor Analysis and Canonical Discriminant Analysis.
- It adopts a similar perspective to PCA: the rows of the data matrix to be examined constitute points in a multidimensional space, as also do the group mean vectors. Discriminating axes are determined in this space, in such a way that optimal separation of the predefined groups is attained. As with PCA, the problem becomes mathematically the eigenreduction of a real, symmetric matrix.

Multiple Discriminant Analysis

- Consider the set of objects, $i \in I$; they are characterised by a finite set of parameters, $j \in J$.
- The vectors associated with the objects are given as the row vectors of the matrix $X = \{x_{ij}\}$.
- The grand mean of all vectors is given by

$$g_j = \frac{1}{n} \sum_{i \in I} x_{ij}$$

  (where n is the cardinality of I).

- Let y_j be the j^{th} coordinate of the mean of group y; i.e.

$$y_j = \frac{1}{n_y} \sum_{i \in y} x_{ij}$$

(where n_y is the cardinality of group y).

- We consider the case of mutually disjoint groups, y, whose union gives I.
- Let Y be the set of these groups, and let n_Y be the number of groups considered. Evidently, $n_Y \leq n$.
- We now define the following three variance–covariance matrices. T (of jk^{th} term, t_{jk}) is the total covariance matrix; W is the within classes covariance matrix; and B is the between classes covariance matrix:

  $\mathbf{T}\ : t_{jk} = \frac{1}{n} \sum_{i \in I} (x_{ij} - g_j)(x_{ik} - g_k)$

  $\mathbf{W}\ : w_{jk} = \frac{1}{n} \sum_{y \in Y} \sum_{i \in y} (x_{ij} - y_j)(x_{ik} - y_k)$

  $\mathbf{B}\ : b_{jk} = \sum_{y \in Y} \frac{n_y}{n} (y_j - g_j)(y_k - g_k).$
- The three matrices, T, W and B, are of dimensions $m \times m$ where m is the number of attributes (i.e. the vectors considered, their grand mean, and the group means are located in $\mathbb{R}^m$).
- Generalizing Huyghen's Theorem in classical mechanics, we have that $T = W + B$. This is proved as follows. We have, for the $(j,k)^{th}$ terms of these

matrices:

$$\frac{1}{n}\sum_{i\in I}(x_{ij}-g_j)(x_{ik}-g_k)$$

$$=\frac{1}{n}\sum_{y\in Y}\sum_{i\in y}(x_{ij}-y_j)(x_{ik}-y_k)+\sum_{y\in Y}\frac{n_y}{n}(y_j-g_j)(y_k-g_k).$$

Rewriting the first term on the right hand side of the equation as

$$\frac{1}{n}\sum_{y\in Y}\sum_{i\in y}((x_{ij}-g_j)-(y_j-g_j))((x_{ik}-g_k)-(y_k-g_k))$$

and expanding gives the required result.

- The sum of squared projections of the points in $\mathbb{R}^m$ along any given axis $\mathbf{u}$ is given by $\mathbf{u}'T\mathbf{u}$ (cf. the analogous situation in principal components analysis).
- For the class means along this axis we have $\mathbf{u}'B\mathbf{u}$.
- Finally, for the within class deviations along this axis, we have $\mathbf{u}'W\mathbf{u}$.
- Since $T = W + B$, we have that

$$\mathbf{u}'T\mathbf{u} = \mathbf{u}'B\mathbf{u} + \mathbf{u}'W\mathbf{u}.$$

- The optimal discrimination of the given groups is carried out as follows. We choose axis $\mathbf{u}$ to maximize the spread of class means, while restraining the compactness of the classes, i.e.

$$max \frac{\mathbf{u}'B\mathbf{u}}{\mathbf{u}'W\mathbf{u}}.$$

- This maximization problem is the same as

$$min \frac{\mathbf{u}'W\mathbf{u}}{\mathbf{u}'B\mathbf{u}} = min \frac{\mathbf{u}'W\mathbf{u}}{\mathbf{u}'B\mathbf{u}} + 1 = min \frac{\mathbf{u}'W\mathbf{u} + \mathbf{u}'B\mathbf{u}}{\mathbf{u}'B\mathbf{u}}$$

$$= min \frac{\mathbf{u}'T\mathbf{u}}{\mathbf{u}'B\mathbf{u}} = max \frac{\mathbf{u}'B\mathbf{u}}{\mathbf{u}'T\mathbf{u}}.$$

- As in PCA, we use λ as a Lagrangian multiplier, and differentiate the expression $\mathbf{u}'B\mathbf{u} - \lambda(\mathbf{u}'T\mathbf{u})$ with respect to $\mathbf{u}$.

- This yields $\mathbf{u}$ as the eigenvector of $T^{-1}B$ associated with the largest eigenvalue, λ.

- Eigenvectors associated with successively large eigenvalues define discriminating factors or axes which are orthogonal to those previously obtained.
- We may therefore say that MDA is the PCA of a set of centred vectors (the group means) in the T^{-1}-metric.
- A difficulty has not been mentioned in the foregoing: the matrix product, $T^{-1}B$ is not necessarily symmetric, and so presents a problem for diagonalization. This difficulty is circumvented as follows. We have that $B\mathbf{u} = \lambda T\mathbf{u}$. Writing B as the product of its square roots CC' (which we can always do because of the fact that B is necessarily positive definite and symmetric) gives: $CC'\mathbf{u} = \lambda T\mathbf{u}$. Next, define a new vector $\mathbf{a}$ as follows: $\mathbf{u} = T^{-1}C\mathbf{a}$. This gives:

$$CC'T^{-1}C\mathbf{a} = \lambda TT^{-1}C\mathbf{a} \Rightarrow C(C'T^{-1}C)\mathbf{a} = \lambda C\mathbf{a} \Rightarrow (C'T^{-1}C)\mathbf{a} = \lambda\mathbf{a}$$

We now have an eigenvalue equation, which has a matrix which is necessarily real and symmetric. This is solved for $\mathbf{a}$, and substituted back to yield $\mathbf{u}$.

- Since the largest eigenvalue is

$$\frac{\mathbf{u}'B\mathbf{u}}{\mathbf{u}'T\mathbf{u}} = \frac{\mathbf{u}'T\mathbf{u} - \mathbf{u}'W\mathbf{u}}{\mathbf{u}'T\mathbf{u}},$$

  it is seen that the right side here, and hence all eigenvalues, are necessarily positive and less than 1.

- The eigenvalues represent the discriminating power of the associated eigenvectors. Unlike in PCA, the percentage variance explained by a factor has no sense in MDA, since the sum of eigenvalues has no meaning.

- The n_Y classes lie in a space of dimension at most $n_Y - 1$. This will be the number of discriminant axes or factors obtainable in the most common practical case when $n > m > n_Y$.

Linear or Fisher Discriminant Analysis

- 2-group case of MDA.
- We will look at assigning a new object (rather than confirming the separation between given groups).
- The distance, in this new T^{-1}-metric, between some new vector **a** and the barycentre (or centre of gravity) $\mathbf{y}$ of class y is defined by the *Mahalanobis* or *generalized distance*:

$$d(a, y) = (\mathbf{a} - \mathbf{y})' T^{-1} (\mathbf{a} - \mathbf{y})$$

- Vector **a** is assigned to the class y such that $d(a, y)$ is minimal over all groups.
- In the two-group case, we have that **a** is assigned to group y_1 if $d(a, y_1) < d(a, y_2)$.
- Writing out explicitly the Euclidean distances associated with the matrix T^{-1},

and following some simplifications, we find that vector $\mathbf{a}$ is assigned to group y_1 if $(\mathbf{y}_1 - \mathbf{y}_2)' T^{-1} \mathbf{a} > \frac{1}{2} (\mathbf{y}_1 - \mathbf{y}_2)' T^{-1} (\mathbf{y}_1 + \mathbf{y}_2)$
and to group y_2 if $(\mathbf{y}_1 - \mathbf{y}_2)' T^{-1} \mathbf{a} < \frac{1}{2} (\mathbf{y}_1 - \mathbf{y}_2)' T^{-1} (\mathbf{y}_1 + \mathbf{y}_2)$

- The left hand side is the T^{-1}–projection of $\mathbf{a}$ onto $\mathbf{y}_1 - \mathbf{y}_2$ (i.e. the vector connecting $\mathbf{y}_2$ to $\mathbf{y}_1$; and the right hand side is the T^{-1}–projection of $(\mathbf{y}_1 + \mathbf{y}_2)/2$ onto $\mathbf{y}_1 - \mathbf{y}_2$.

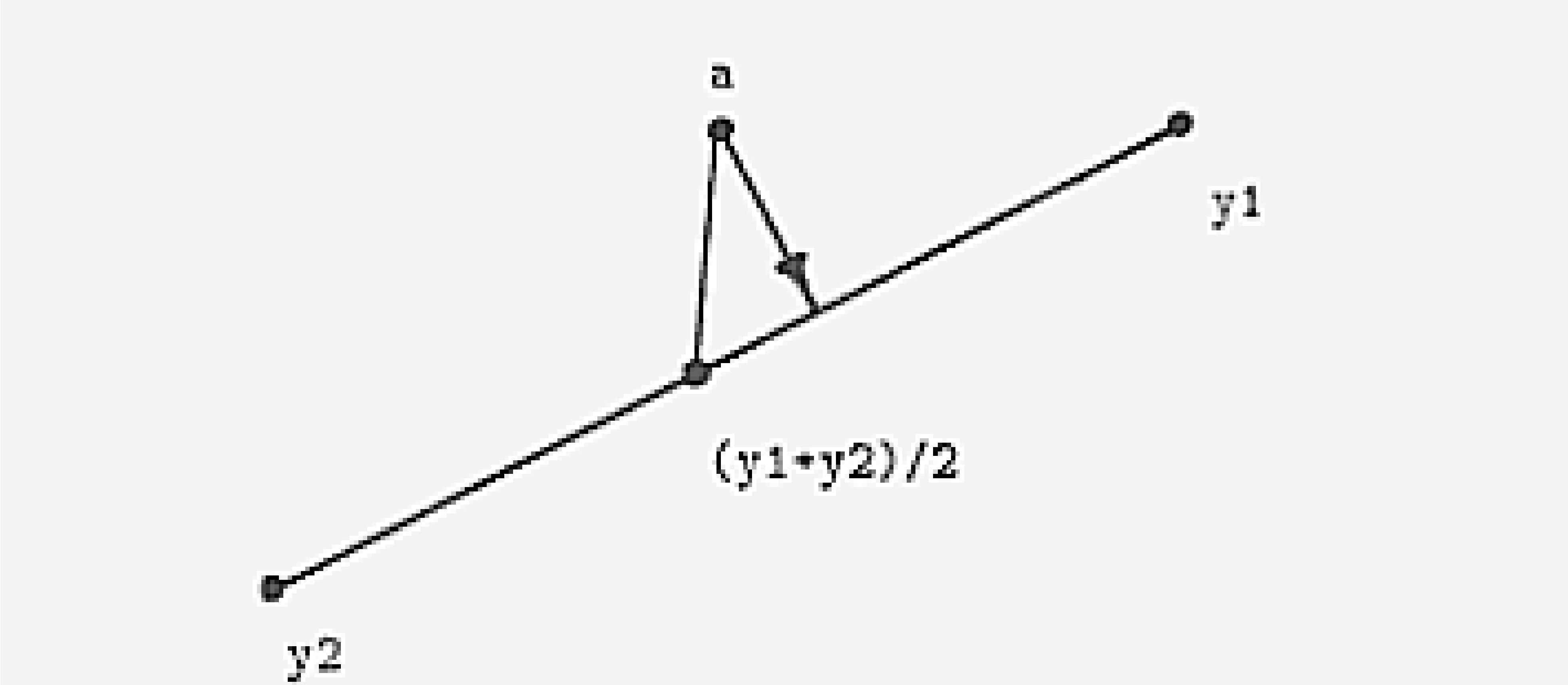
- This allocation rule may be rewritten as

$$(\mathbf{y}_1 - \mathbf{y}_2)' T^{-1} (\mathbf{a} - \frac{\mathbf{y}_1 + \mathbf{y}_2}{2}) > 0 \Longrightarrow \mathbf{a} \rightarrow y_1$$

$$(\mathbf{y}_1 - \mathbf{y}_2)' T^{-1} (\mathbf{a} - \frac{\mathbf{y}_1 + \mathbf{y}_2}{2}) < 0 \Longrightarrow \mathbf{a} \rightarrow y_2.$$

The left hand side here is known as *Fisher's linear discriminant function.*

Fisher’s Linear Discriminant



The assignment of a new sample $\mathbf{a}$ to one of two groups of centres $\mathbf{y}_1$ and $\mathbf{y}_2$.

Bayesian Discrimination: Quadratic Case

- Consider a vector of measured parameters, $\mathbf{x}$, relating to attributes of galaxies.
- Next consider that a sample of galaxies which is being studied consists of 75% spirals and 25% ellipticals.
- That is, $P(spiral) = 0.75, P(elliptical) = 0.25$
- In the absence of any other information, we would therefore assign any unknown galaxy to the class of spirals. In the long run, we would be correct in 75% of cases, but we have obviously derived a very crude assignment rule.
- Consider now that we are given also the conditional probabilities: for a particular set of parameter values, $\mathbf{x}_0$, we have $P(spiral \mid \mathbf{x}_0) = 0.3, \qquad P(elliptical \mid \mathbf{x}_0) = 0.7$
- In this case, we are led to choose the class of ellipticals for our unknown galaxy, for which we have measured the parameter values $\mathbf{x}_0$.

- This leads to Bayes' rule for the assignment of an unknown object to group c rather than to any other group, y: $P(c \mid \mathbf{x}_0) > P(y \mid \mathbf{x}_0) \quad for\ all\ y \neq c$
- A difficulty arises with Bayes' rule as defined above: although we could attempt to determine $P(c \mid \mathbf{x})$ for all possible values of $\mathbf{x}$ (or, perhaps, for a discrete set of such values), this is cumbersome.
- In fact, it is usually simpler to derive values for $P(\mathbf{x}_0 \mid c)$, i.e. the probability of having a given set of measurements, $\mathbf{x}_0$, given that we are dealing with a given class, c.
- Bayes' theorem relates priors and posteriors.

$$P(c \mid \mathbf{x}_0) = \frac{P(\mathbf{x}_0 \mid c)P(c)}{\sum_{all\ y} P(\mathbf{x}_0 \mid y)P(y)}$$

- All terms on the right hand side can be sampled: $P(c)$ is determined straightforwardly; $P(\mathbf{x}_0 \mid c)$ may be sampled by looking at each parameter in turn among the vector $\mathbf{x}_0$, and deriving estimates for the members of class c.
- Assignment rule: Choose class c over all classes y, if

$$P(\mathbf{x}_0 \mid c)\ P(c) > P(\mathbf{x}_0 \mid y)\ P(y) \quad for\ all\ y \neq c$$

- But again a difficulty arises: a great deal of sampling is required to estimate the terms of the above expression.

- Hence it is convenient to make distributional assumptions about the data.

- The multivariate normal density function (defining a multidimensional bell-shaped curve) is taken to better represent the distribution of $\mathbf{x}$ than the single point as heretofore. This is defined as $(2\pi)^{-\frac{n}{2}} \mid V \mid^{-\frac{1}{2}}\ exp\ (-\frac{1}{2}(\mathbf{x} - \mathbf{g})' V^{-1} (\mathbf{x} - \mathbf{g}))$

- Where V is the variance-covariance matrix. It is of dimensions $m \times m$, if m is the dimensionality of the space. If equal to the identity matrix, it would indicate that $\mathbf{x}$ is distributed in a perfectly symmetric fashion with no privileged direction of elongation. $\mid V \mid$ is the determinant of the matrix V.

- Assuming that each group, c, is a Gaussian, we have $P(\mathbf{x} \mid c) = (2\pi)^{-\frac{n}{2}} \mid V_c \mid^{-\frac{1}{2}}\ exp\ (-\frac{1}{2}(\mathbf{x} - \mathbf{g}_c)' V_c^{-1} (\mathbf{x} - \mathbf{g}_c))$ where $\mathbf{g}_c$ is the centre of class c, and V_c is its variance-covariance matrix.

- Substituting, taking natural logs of both sides of the inequality, and cancelling common terms on both sides, gives the following assignment rule: Assign $\mathbf{x}$ to class c if

$$ln \mid V_c \mid + (\mathbf{x} - \mathbf{g}_c)' V_c^{-1} (\mathbf{x} - \mathbf{g}_c) - ln\ P(c)$$
$$< ln \mid V_y \mid + (\mathbf{x} - \mathbf{g}_y)' V_y^{-1} (\mathbf{x} - \mathbf{g}_y) - ln\ P(y) \quad for\ all\ y \neq c.$$

- This expression is simplified by defining a “discriminant score” as

$$\delta_c(\mathbf{x}) = ln \mid V_c \mid + (\mathbf{x} - \mathbf{g_c})' V_c^{-1} (\mathbf{x} - \mathbf{g_c})$$

- The assignment rule then becomes: Assign $\mathbf{x}$ to class c if

$$\delta_c(\mathbf{x}) - ln\ P(c) < \delta_y(\mathbf{x}) - ln\ P(y) \quad for\ all\ y \neq c.$$

- The dividing curve between any two classes immediately follows from this. It is defined by: $\delta_c(\mathbf{x}) - ln\ P(c) = \delta_y(\mathbf{x}) - ln\ P(y) \quad for\ all\ y \neq c$

- The shape of a curve defined by this equation is quadratic. Hence this general form of Bayesian discrimination is also referred to as *quadratic discrimination.*

Maximum Likelihood Discrimination

- In a practical context we must estimate the mean vectors ($\mathbf{g}_y$) and the variance-covariance matrices (V_y) from the data which may be taken to constitute a sample from an underlying population.
- We have used a multivariate normal density function for $P(\mathbf{x} \mid y)$.
- If all n objects $\mathbf{x}_i$ have been independently sampled, then their joint distribution is

$$\mathcal{L} = \Pi_{i=1}^{n} P(\mathbf{x}_i \mid y).$$

- Considering $\mathcal{L}$ as a function of the unknown parameters $\mathbf{g}$ and V, it is termed a *likelihood function.*
- The *principle of maximum likelihood* then states that we should choose the unknown parameters such that $\mathcal{L}$ is maximized.

- The classical approach for optimizing $\mathcal{L}$ is to differentiate it with respect to $\mathbf{g}$ and then with respect to V, and to set the results equal to zero.
- Doing this for the multivariate normal expression used previously allows us to derive estimates for the mean and covariances as follows.

$$\hat{\mathbf{g}} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i$$

$$\hat{V} = \frac{1}{n} \sum_{i=1}^{n} (\mathbf{x}_i - \mathbf{g})(\mathbf{x}_i - \mathbf{g})'.$$

- These are used to provide *maximum likelihood estimates* for the Bayesian classifier.
- In a more general setting, we could wish to consider a multivariate normal mixture of the following form:

$$P(\mathbf{x} \mid y) = \sum_k w_k f_k(\mathbf{x} \mid \mathbf{g}_k, V_k)$$

where k ranges over the set of mixture members, w is a weighting factor, and the function f depends on the mean and the covariance structure of the mixture members.

- For such density functions, an iterative rather than an analytic approach is used

Bayesian Equal Covariances Case

- The groups we study will not ordinarily have the same covariance structure.
- However it may be possible to assume that this is the case, and here we study what this implies in Bayesian discrimination.

  The discriminant score, when expanded, is

$$\delta_c(\mathbf{x}) = ln \mid V_c \mid + \mathbf{x}' V_c^{-1} \mathbf{x} - \mathbf{g}_c' V_c^{-1} \mathbf{x} - \mathbf{x}' V_c^{-1} \mathbf{g}_c + \mathbf{g}_c' V_c^{-1} \mathbf{g}_c .$$

- The first two terms on the right hand side can be ignored since they will feature on both sides of the assignment rule (by virtue of our assumption of equal covariances); and the third and fourth terms are equal.
- If we write $\phi_c(\mathbf{x}) = 2\mathbf{g}_c' V_c^{-1} \mathbf{x} - \mathbf{g}_c' V_c^{-1} \mathbf{g}_c$ then the assignment rule is: Assign $\mathbf{x}$ to class c if

$$\phi_c(\mathbf{x}) + ln\ P(c) > \phi_y(\mathbf{x}) + ln\ P(y) \quad for\ all\ y \neq c.$$

- However, ϕ can be further simplified. Its second term is a constant for a given group, a_0; and its first term can be regarded as a vector of constant coefficients (for a given group), $\mathbf{a}$.
- Hence ϕ may be written as: $\phi_c(\mathbf{x}) = a_{c0} + \sum_{j=1}^{m} a_{cj} x_j$
- Assuming $P(c) = P(y)$, for all y, the assignment rule in the case of equal covariances thus involves a linear decision surface.
- We have a result which is particularly pleasing from the mathematical point of view: Bayesian discrimination in the equal covariances case, when the group cardinalities are equal, gives *exactly* the same decision rule (i.e. a linear decision surface) as linear discriminant analysis discussed from a geometric standpoint.

Non-Parametric Discrimination

- Non-parametric (distribution-free) methods dispense with the need for assumptions regarding the probability density function.
- Given a vector of parameter values, $\mathbf{x}_0$, the probability that any unknown point will fall in a local neighbourhood of $\mathbf{x}_0$ may be defined in terms of the relative volume of this neighbourhood.
- If n' points fall in this region, out of a set of n points in total, and if v is the volume taken up by the region, then the probability that any unknown point falls in the local neighbourhood of $\mathbf{x}_0$ is n'/nv.
- In the k-*NN* (k nearest neighbours) approach, we specify that the volume is to be defined by the k NNs of the unclassified point.
- Consider n_c of these k NNs to be members of class c, and n_y to be members of class y (with $n_c + n_y = k$).
- The conditional probabilities of membership in classes c and y are then

$$P(\mathbf{x}_0 \mid c) = \frac{n_c}{nv}$$

$$P(\mathbf{x}_0 \mid y) = \frac{n_y}{nv}.$$

- Hence the decision rule is: Assign to group c if

$$\frac{n_c}{nv} > \frac{n_y}{nv}$$

$$i.e. \quad n_c > n_y$$

- The determining of NNs of course requires the definition of distance: the Euclidean distance is usually used.
- An interesting theoretical property of the NN–rule relates it to the Bayesian misclassification rate.
- The latter is defined as $1 - \max_y P(y \mid \mathbf{x}_0)$ or, using notation introduced previously, $1 - P(c \mid \mathbf{x}_0)$
- This is the probability that $\mathbf{x}_0$ will be misclassified, given that it should be classified into group c.

- In the 1-NN approach, the misclassification rate is the product of: the conditional probability of class y given the measurement vector $\mathbf{x}$, and one minus the conditional probability of class y given the NN of $\mathbf{x}$ as the measurement vector:

$$\sum_{all\ y} P(y \mid \mathbf{x})(1 - P(y \mid NN(\mathbf{x}))$$

- This is the probability that we assign to class y given that the NN is not in this class.
- It may be shown that the misclassification rate in the 1–NN approach is not larger than twice the Bayesian misclassification rate.

Multilayer Perceptron

- The multilayer perceptron (MLP) is an example of a *supervised* method, in that it a training set of samples or items of known properties is used.
- In dealing with the MLP, the single perceptron is first described, and subsequently the networking of perceptrons in a set of interconnected, multiple layers to form the MLP.
- The influential generalized delta rule, used in training the network, is introduced via the simpler case of the delta rule.
- The perceptron algorithm is due to Rosenblatt in the late 1950s. The perceptron, a simple computing engine which has been dubbed a “linear machine” for reasons which will become clear.
- Let **x** be an input vector of binary values; o an output scalar; and bf w a vector of weights (or learning coefficients; initially containing arbitrary values). The perceptron calculates $o = \sum_j w_j x_j$. Let θ be some threshold.

- If $o \geq \theta$, when we would have wished $o < \theta$ for the given input, then i is incorrectly categorized. We therefore seek to modify the weights and the threshold.
- Set $\theta \longleftarrow \theta + 1$ to make it less likely that wrong categorization will take place again.
- If $x_j = 0$ then no change is made to w_j. If $x_j = 1$ then $w_j \longleftarrow w_j - 1$ to lessen the influence of this weight.
- If the output was found to be less than the threshold, when it should have been greater for the given input, then the reciprocal updating schedule is implemented.
- The updates to weights and thresholds may be denoted as follows::
  $o = \sum_j w_j x_j$
  $\Delta\theta = -(t_p - o_p) = -\delta_p$ (change in threshold for pattern p)
  $\Delta w_i = (t_p - o_p)x_{pi} = \delta_p x_{pi}$ (change in weights for pattern p).
- If a set of weights exist, then the perceptron algorithm will find them. A

counter-example is the exclusive-or, XOR, problem.

- The line (hyperplane) separating T from F is defined by $\sum_j w_j x_j = \theta$. In the XOR case, linear separability does not hold. Perceptron learning fails for non-separable problems.

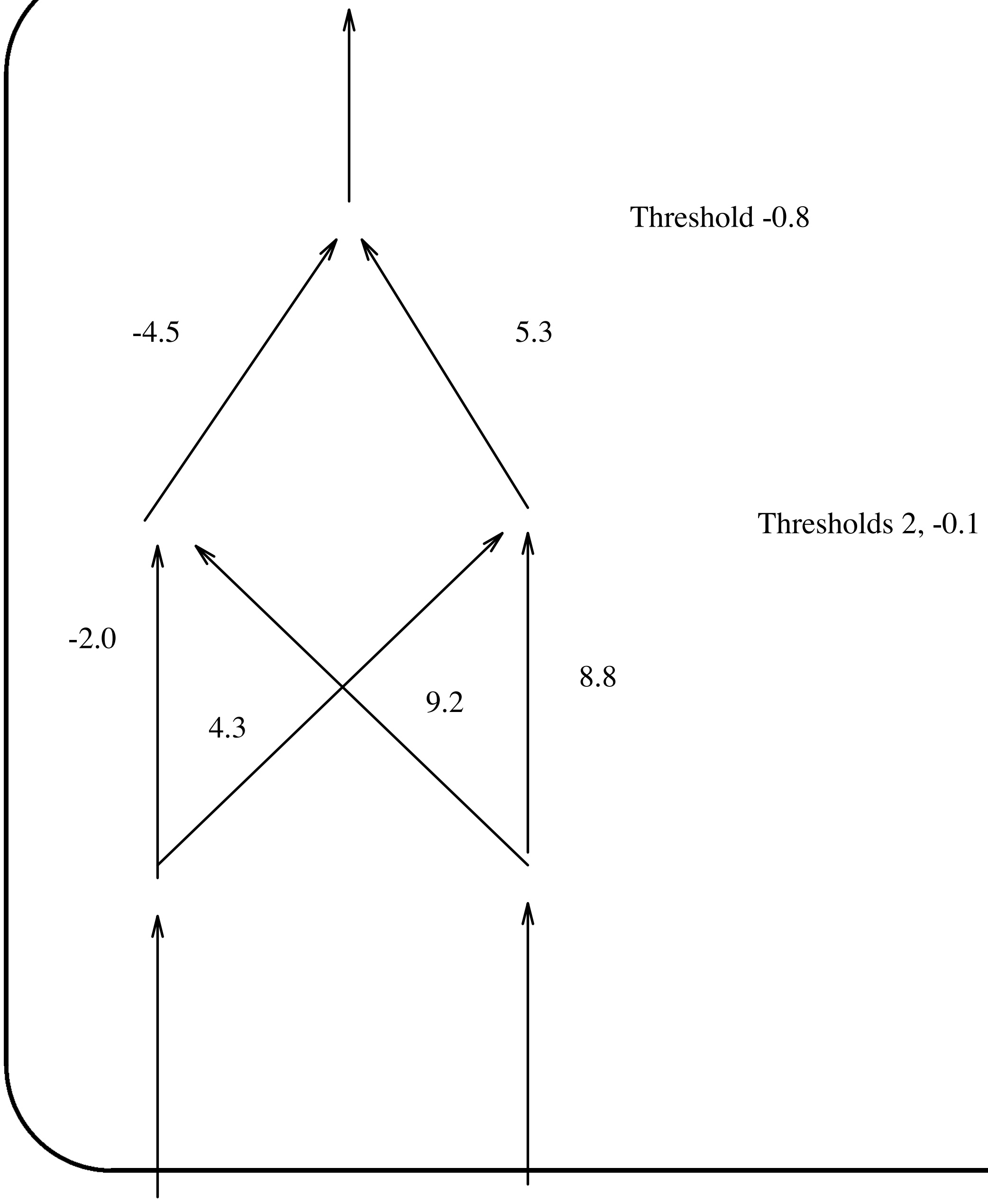
Multilayer Perceptron

AND		
0	0	F
0	1	F
1	0	F
1	1	T

OR		
0	0	F
0	1	T
1	0	T
1	1	T

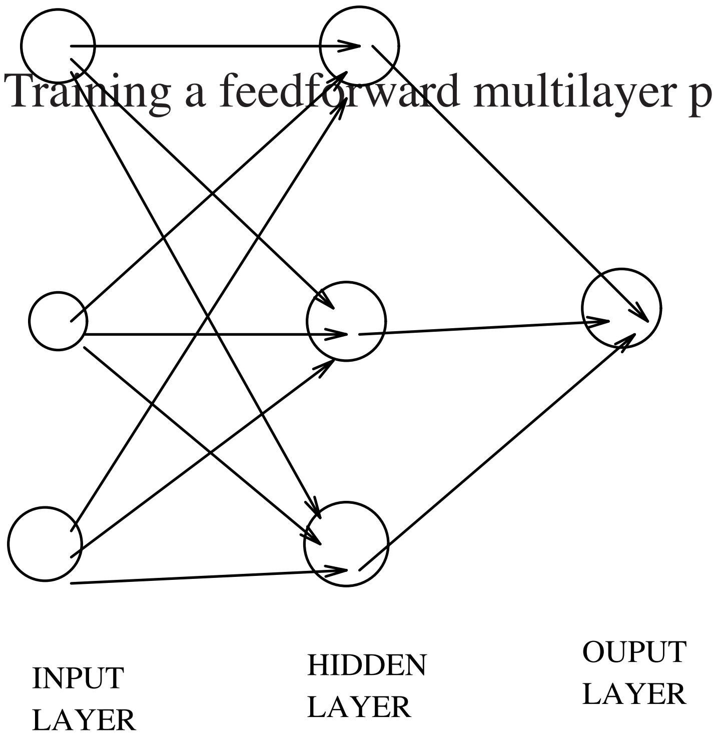
XOR		
0	0	F
0	1	T
1	0	T
1	1	F

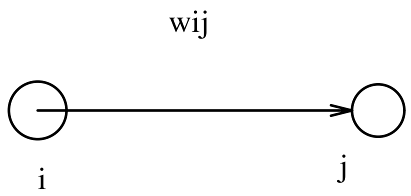
Multilayer Perceptron Solution for XOR Problem


Threshold -0.8
-4.5
5.3
Thresholds 2, -0.1
-2.0
8.8
9.2
4.3

The Generalized Delta Rule

Training a feedforward multilayer perceptron.





The Generalized Delta Rule

- Initially we consider linear units only, i.e. $o_{pj} = \sum_i w_{ij} x_{pi}$
- The input at the ith neuron, x_{pi}, is occasioned by the pth pattern.
- The weight connecting the ith neuron in a given layer, to the jth neuron in the subsequent layer, is denoted w_{ij}.
- Consider an error term which we seek to minimize at each output neuron j; and let p by be the pattern which is currently being presented to the net.
- Then $E_p = \frac{1}{2} \sum_j (t_{pj} - o_{pj})^2$ where o is the output obtained when using the current set of weights.
- The multiplicative constant of a half is purely conventional, to make this error term look like an energy expression.
- The target output, t, is what we wish the network to replicate on being presented with the pth pattern.

- Consider $E = \sum_p E_p$
- We may write the expression for E_p as $\frac{1}{2} \sum_j \delta_{pj}^2$
- The rate of change of E_p with respect to w_{ij} is given by the chain rule:

$$\frac{\partial E_p}{\partial w_{ij}} = \frac{\partial E_p}{\partial o_{pj}} \frac{\partial o_{pj}}{\partial w_{ij}}$$

- Now, since $E_p = \frac{1}{2} \sum_j (t_{pj} - o_{pj})^2$, the first term here is $-(t_{pj} - o_{pj})$.
- Given that linear units are being considered, i.e. $o_{pj} = \sum_i w_{ij} x_{pi}$, the second term equals x_{pi}.
- The gradient of E_p is thus

$$\frac{\partial E_p}{\partial w_{ij}} = -\delta_{pj} x_{pi}$$

- If $\partial E / \partial w_{ij} = \sum_p \partial E_p / \partial w_{ij}$ and if updates do not take place after every training pattern, p, then we may consider the updating of weights as a classical gradient descent minimization of E.

- Each weight is updated according to

$$\Delta_p w_{ij} = \eta(t_{pj} - o_{pj})x_{pi}$$

  where η is a small constant.

- This corresponds to steepest descent optimization: we vary the weights in accordance with the downwards slope.
- So far, with linear units, we have done the following.
- Given a vector of inputs, x_p, the values at the hidden layer are given by $x_p W_1$. The values at the output layer are then $x_p W_1 W_2$. Note that we can “collapse” our network to one layer by seeking the weights matrix $W = W_1 W_2$. If t_p is the target vector, then we are seeking a solution of the equation $x_p W = t_p$. This is linear regression.
- Backpropagation assumes greater relevance when nonlinear transfer functions are used at neurons.

The Generalized Delta Rule for Nonlinear Units

- Nonlinear transformations are less tractable mathematically but may offer more sensitive modeling of real data.
- They provide a more faithful modeling of electronic gates or biological neurons.
- Consider the accumulation of weighted values of a neuron

$$\mathrm{net}_{pj} = \sum_i w_{ij} o_{pi}$$

  where $o_i = x_i$ if unit i is an input one.

- This is passed through a differentiable and nondecreasing transformation, f,

$$o_{pj} = f_j(\mathrm{net}_{pj})$$

- Normally this transfer function is a sigmoidal one.
- If it were a step function, this would violate our requirement for a differentiable

function.

- One possibility is the function $y = (1 + e^{-x})^{-1}$. Another choice is the hyperbolic tangent or tanh function: for $x > 20.0, y = +1.0$; for $y < -20.0, y = -1.0$; otherwise $y = (e^x - e^{-x})/(e^x + e^{-x})$
- Both of these functions are invertible and continuously differentiable.
- Both have semilinear zones which allow good (linear) fidelity to input data.
- Both can make “soft” or fuzzy decisions.
- Finally, they are similar to the response curve of a biological neuron.
- As before, the change in weights will be defined to be proportional to the energy (or error function) slope, and the chain rule yeilds:

$$\Delta_p w_{ij} \propto -\frac{\partial E_p}{\partial w_{ij}} = -\frac{\partial E_p}{\partial \mathrm{net}_{pj}} \frac{\partial \mathrm{net}_{pj}}{\partial w_{ij}}$$

- From the definition of net_{pj}, the last term is o_{pi}. Let $\delta_{pj} = -\partial E_p / \partial \mathrm{net}_{pj}$.

- Hence

  $$-\frac{\partial E_p}{\partial w_{ij}} = \delta_{pj} o_{pj}$$

  or

  $$\Delta_p w_{ij} = \eta \delta_{pj} o_{pi}$$

  where η is the learning constant, usually a small fraction which prevents rebounding from side to side in ravines of the energy surface.

- Note that for a linear output unit, by definition $o_{pj} = \mathrm{net}_{pj}$ and so we have $-\partial E_p / \partial o_{pj} = \delta_{pj}$ as was seen above when considering such units.

- It must now be determined how to define δ_{pj}.

- We have

  $$\delta_{pj} = -\frac{\partial E_p}{\partial \mathrm{net}_{pj}} = -\frac{\partial E_p}{\partial \mathrm{o}_{pj}} \frac{\partial o_{pj}}{\partial \mathrm{net}_{pj}}$$

  and the last term is equal to $f'_j(\mathrm{net}_p j)$, i.e. the derivative of the transfer function.

- Two cases will be distinguished depending on whether the unit is an output one or a hidden layer one.

Case 1: Unit j is an output one, and it is found that

$$\delta_{pj} = (t_{pj} - o_{pj}) f'_j(\text{net}_{pj})$$

Case 2: Unit j is a hidden layer one and it is found that

$$\delta_{pj} = f'_j(\text{net}_{pj}) \sum_k \delta_{pk} w_{kj}$$

- Hence the deltas at an internal node can be derived from the deltas at a subsequent (closer to output) node.
- The overall algorithm is as follows: present pattern; feed forward, through successive layers; backpropagate – updating weight; repeat.
- An alternative is to determine the changes in weights as a result of presenting all patterns: $\Delta w_{ij} = \sum_p \Delta_p w_{ij}$. This so-called “off-line” updating of weights is computationally more efficient but loses on the adaptivity of the overall approach.

- Some further notes on the multilayer perceptron using the generalized delta rule follow.
- A local optimum set of weights may be arrived at. There is no guarantee of arriving at a global optimum in weight space.
- For the logistic activation function defined by

$$f_j(\mathrm{net}_{pj}) = 1/(1 + \exp^{-\mathrm{net}_{pj}})$$

where

$$\mathrm{net}_{pj} = \sum_i w_{ij} o_{pi} + \theta_j$$

and the last term is a bias (or threshold, we have the following result:

$$f_j'(\mathrm{net}_{pj}) = o_{pj}(1 - o_{pj})$$

- For this function:

$$\delta_{pj} = (t_{pj} - o_{pj}) o_{pj} (1 - o_{pj})$$

for an output unit,

$$\delta_{pj} = o_{pj}(1 - o_{pj}) \sum_k \delta_{pk} w_{kj}$$

for a hidden layer unit.

- In practice one must use approximations to hard-limiting values of 0, 1; e.g. 0.1, 0.9 can be used. Otherwise, infinite weight values would be needed in the above expression for $f_j(\text{net}_{pj})$.
- It is found that symmetry breaking is necessary in order to get the backpropagation algorithm started. This involves randomizing the initial arbitrary weight values.
- The presence of an additional momentum term, $\Delta w_{ij}^{(n+1)} = \eta \delta_{pj} o_{pi} + \alpha \Delta w_{ij}^{(n)}$ often helps the convergence properties of the steepest descent iterative optimization.
- This term takes the effect of previous steps into account. If the change in the previous step was large, the momentum tends to continue to keep the delta large.

- The learning rate, η, should be small (0.7 or smaller).
- The MLP architecture using the generalized delta rule can be very slow.
- The number of hidden layers in an MLP and the number of nodes in each layer can vary for a given problem. In general, more nodes offer greater sensitivity to the problem being solved, but also the risk of overfitting.

Examples

- Faint star/galaxy discrimination
- Cosmic ray hit/point source discrimination