Multidimensional Signal Analysis Part III – Lecture 04

November 6, 2003

Contents

References

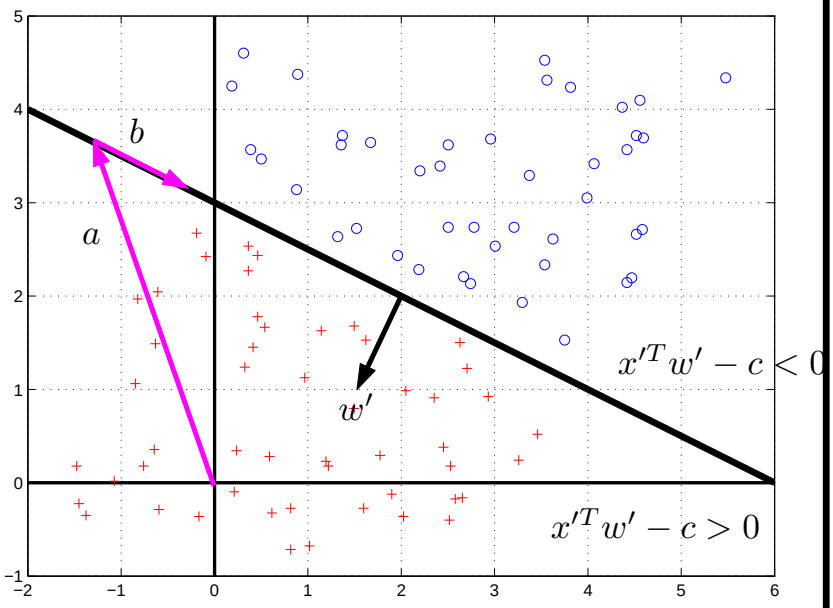
[Theodoridis and Koutroumbas, 1999] Theodoridis, S. and Koutroumbas, K. (1999). *Pattern Recognition*. Academic Press, San Diego.

1 The perceptron

[Theodoridis and Koutroumbas, 1999, 3.3]

Discrimination between two classes assumed linearly separable

$$
\begin{aligned}
x' &= a + \beta b \\
x'^T w' &= \underbrace{a^T w'}_{c} + \beta \underbrace{b^T w'}_{0} \\
x'^T w' - c &= 0
\end{aligned}
$$



Linear separability given if

$$
\begin{aligned}
w'^T x'_i - c > 0 \qquad & \forall x'_i \in \text{class 1} \\
w'^T x'_i - c < 0 \qquad & \forall x'_i \in \text{class 2}
\end{aligned}
$$

Equivalently,

$$
\begin{aligned}
w'^T x' - c &= [w_1 \quad \cdots \quad w_D] \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_D \end{bmatrix} - c = [w_1 \quad \cdots \quad w_D \quad -c] \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_D \\ 1 \end{bmatrix} \\
&= w^T x
\end{aligned}
$$

so that

$$w^T x_i > 0 \qquad \forall x_i \in \text{class } 1$$
$$w^T x_i < 0 \qquad \forall x_i \in \text{class } 2$$

To optimize the linear discriminant function, choose a loss function, e.g.

$$\begin{aligned} E(w) &= \sum_{i \in \text{misclassified}} |w^T x_i| \\ &= \sum_{i \in \text{misclassified}} k_i w^T x_i \end{aligned}$$

$$\text{with} \quad k_i = \begin{cases} +1 \forall x_i \in \text{class } 1 \\ -1 \forall x_i \in \text{class } 2 \end{cases}$$

The loss function is continuous but not differentiable everywhere.

Minimize the loss function using steepest descent

$$w(t+1) = w(t) - \rho_t \frac{\partial E(w)}{\partial w}\bigg|_{w=w(t)}$$

with t – iteration index and ρ – stepsize.

At all points at which the derivative exists,

$$\frac{\partial E(w)}{\partial w} = \sum_{i \in \text{misclassified}} k_i x_i$$

such that

$$w(t+1) = w(t) - \rho_t \sum_{i \in \text{misclassified}} k_i x_i$$

which is defined at all points.

It can be proven that if the sequence ρ_t is chosen such that

$$\lim_{t\to\infty}\sum_{k=0}^{t}\rho_k^2 < \infty, \quad \lim_{t\to\infty}\sum_{k=0}^{t}\rho_k \to \infty$$

$$\left(\text{e.g. } \rho_t = \frac{\text{const.}}{t}\right)$$

then the algorithm converges in a finite number of steps, but

- convergence can be slow
- and is proven to exist only for linearly separable classes: can fail for non-linearly separable data
  ($\rightarrow$ "pocket-algorithm" which stores the best previous solution)

The steepest descent algorithm introduced above processes the *entire* training set in *each* iteration: it is a **batch algorithm**.

Online or **pattern algorithms** adjust the parameters after each presentation of a single training vector.

Online algorithms are order-dependent. Random selection of training vectors for presentation can help find better local minima.

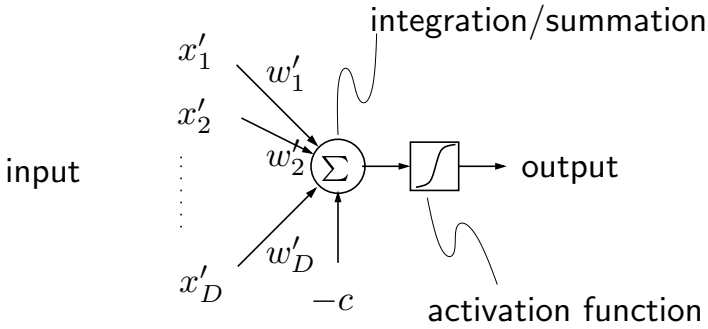
2 Multi-layer perceptron (MLP) / feed-forward neural network (NN)

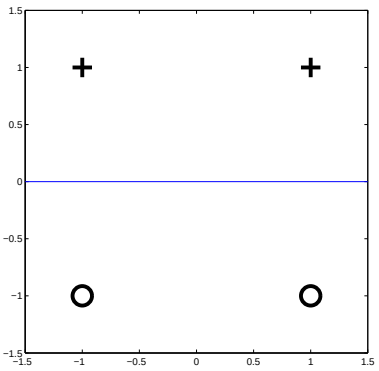
[Theodoridis and Koutroumbas, 1999, 4]

Perceptron discrimination surface

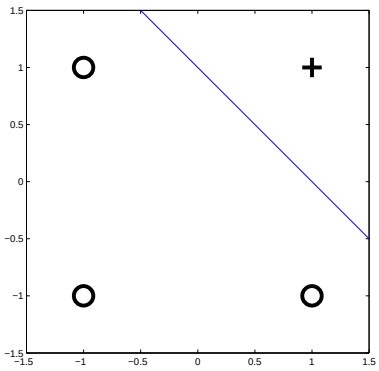
$$w'x' - c = 0$$

$$\sum_{k=1}^{D} x'_k w'_k - c = 0$$

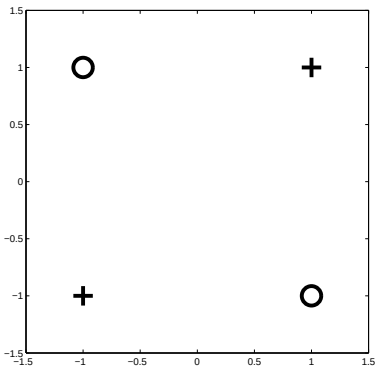




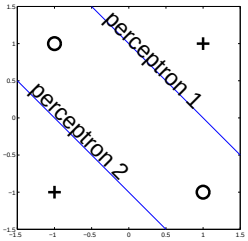
no problem



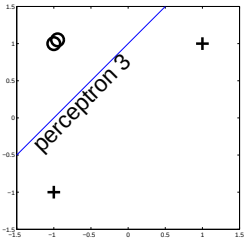
no problem

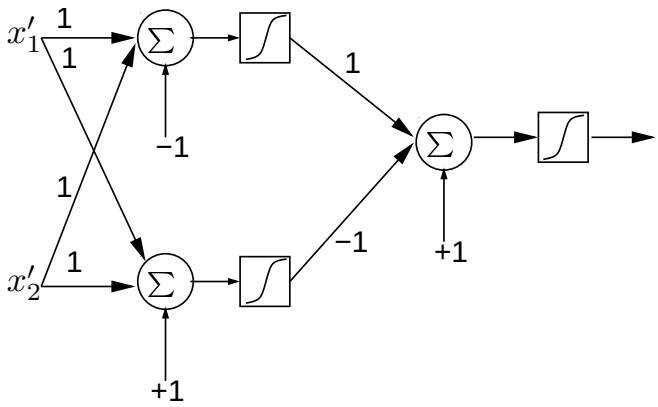


problem!



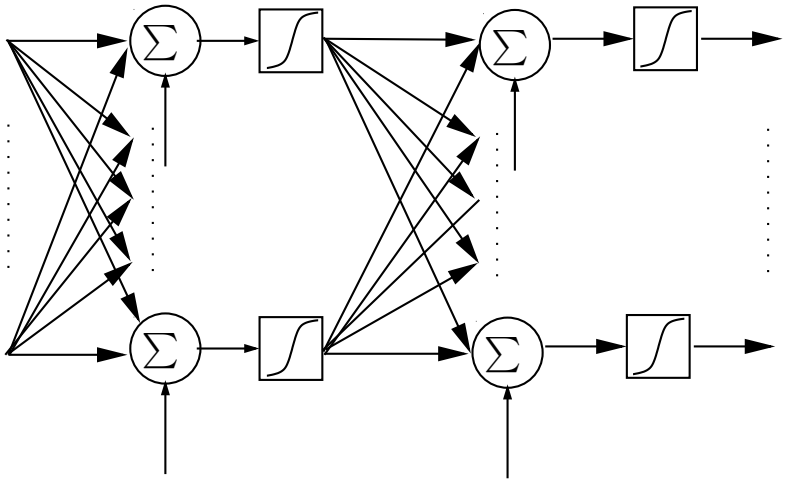
		above perceptron 1	above perceptron 2		
+	1,1	yes	yes	→	1,1
+	-1,-1	no	no	→	-1,-1
○	1,-1	no	yes	→	-1,1
○	-1,1	no	yes	→	-1,1





+	1,1	⟶	1
+	-1,-1	⟶	1
○	1,-1	⟶	-1
○	-1,1	⟶	-1

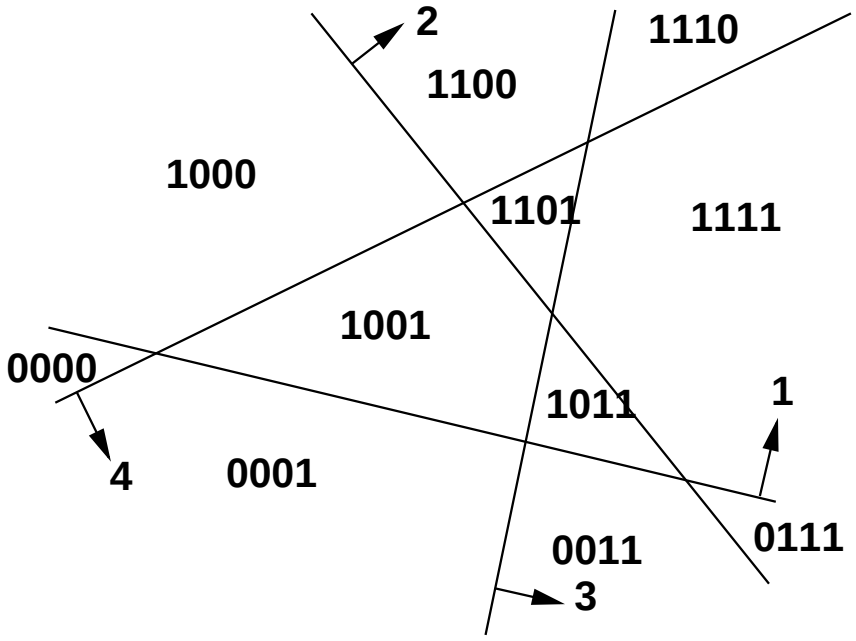
2.1 General two-layer perceptron



number of input layer nodes = dimension of input space
number of output layer nodes = dimension of output space

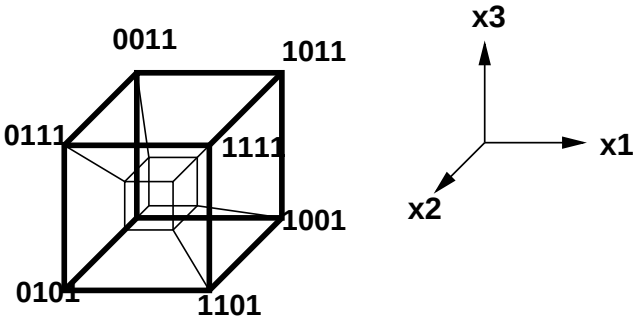
Each node in the hidden layer corresponds to a linear discriminant function.

Example: two input layer neurons, four hidden neurons

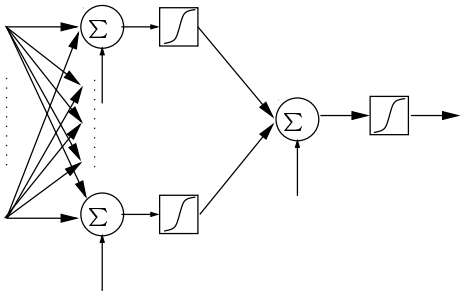


Only 11 out of the $2^4 = 16$ possible combinations are present. Missing is, for instance, 0010.

Each polyhedron $\{0,1\}^4$ corresponds to the corner of a 4-dimensional cube; not all corners of the cube have a corresponding polyhedron (e.g. 0010, a "virtual" polyhedron).



All unions of polyhedra whose corners can be cut off with a single (hyper-) plane can be represented as a two-layer perceptron with a single output node.



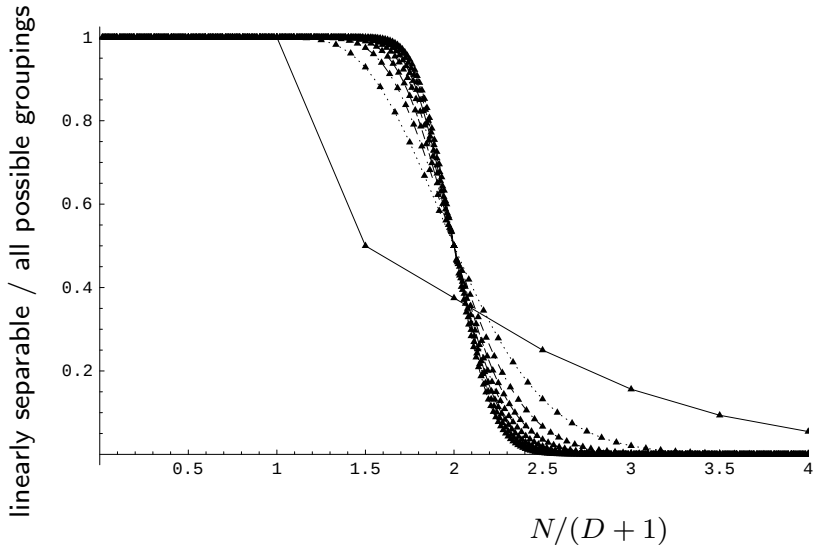
In essence, the data are mapped to a higher-dimensional space in which they become linearly separable ("projection trick", see following lecture)

$\Rightarrow$ "Generalized linear discrimination"

- MLP
- polynomial classifiers
- radial basis function networks
- support vector machines

Why does mapping to a higher-dimensional space help?

$\rightarrow$ given N points in D dimensions, there are 2^N possible groupings into two classes. If the points are in general position, a linear classifier can generate $2\sum_{d=1}^{D}\binom{N-1}{d}$ of these groupings [Theodoridis and Koutroumbas, 1999, 4.13].



In words, the multilayer perceptron maps non-linearly separable data to a higher-dimensional space and attempts a linear separation therein.

This works because the chance that any two groups of N points are linearly separable is almost 1 for $N < 2(D+1)$ and D large.

There is an ∞ number of possible mappings to higher-dimensional spaces.

Neural networks with a single hidden layer are universal approximators!
(Though MLPs with more than one hidden layer may require fewer neurons for a given task.)

Constructive proof that a three-layer perceptron (two hidden layers) can separate classes from arbitrary unions of polyhedral regions:

1. partition space using linear discriminators (from the first layer) such that each resultant polyhedron contains members of one class only.
2. identify those polyhedra containing samples from class 1 only
3. for each of these polyhedra, introduce a node in the second hidden layer that cuts off the corresponding corner of the hypercube only
4. use an OR-neuron (all weights=1) in the output layer.

2.2 Adjustable parameters

- activation function
  - McCulloch-Pitts 
  - logistic  $\frac{1}{\exp(-\alpha x)}$
- cost function for parametrization
  - least squares $\sum_{i=1}^{N}\sum_{k=1}^{D}(y_k(i)-\hat{y}_k(i))^2$
  - entropy fit $\sum_{i=1}^{N}\sum_{k=1}^{D} y_k(i)\log\frac{\hat{y}_k(i)}{y_k(i)}$
- network size

  $\rightarrow$ strategies:
  - build perfect network and prune
  - shrinkage: favor small weights in regularized cost function

2.3 Backpropagation

another word for steepest descent, and just one of the many conceivable optimization strategies

$$
\begin{aligned}
E(w) &= \sum_{i=1}^{N} \varepsilon_w(i) \qquad \varepsilon - \text{entropy or least squares} \\
w_j^r(t+1) &= w_j^r(t) - \rho \frac{\partial E(w)}{\partial w_j^r}\bigg|_{w=w(t)} \\
r &- \text{layer index} \\
j &- \text{node index in layer r}
\end{aligned}
$$

$$\downarrow$$

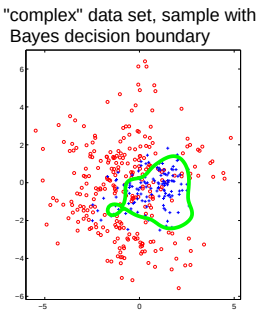
chain rule, chain rule, chain rule

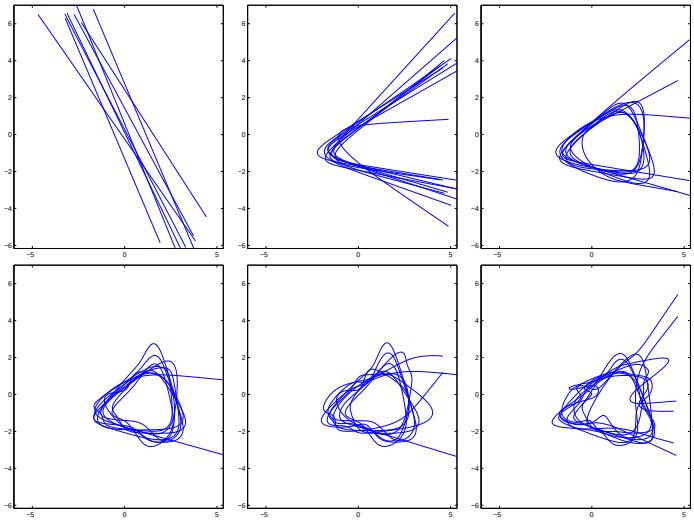
$$\vdots$$

recursively over activation unit, summation function, etc.

To a hidden-layer neuron in a MLP, all points on a parallel to the decision plane are indistinguishable $\rightarrow$ the influence region of a hidden-layer neuron in a MLP is global, while it is local in a radial basis function network. MLPs can have better generalization performance, but are harder to train.

Decision boundaries of MLPs with $1 \ldots 6$ hidden neurons for 10 different samples from the "complex" data set. Also shown is one of these samples along with the Bayes decision boundary.





2.4 Variations

- instead of steepest descent, use
  - momentum term
  - conjugate gradients
  - natural gradients
  - Newton-Raphson
  - Levenberg-Marquardt
  - Broyden-Fletcher-Goldfarb-Shanno
  - genetic algorithms
  - simulated annealing

    ⋮
- transformation invariance
  can be obtained through "weight sharing"

2.5 Initialization and Visualization of network structure

A MLP can be initialized as a linear model, e.g. as a principal components regression (PCR).

Weights of input layer specify axes of eigenspace in the original coordinate system.

Weights of single node in the hidden layer correspond to regression coefficients.

Coordinates of objects can be visualized in terms of the new coordinate system.

Training will drive the model away from linearity only if truly required by the data.