Inference and Decision

- ***Inference stage***: use the training data to learn a model for $p(C_k \mid \boldsymbol{x})$

- ***Decision stage***: use the given posterior probabilities to make optimal class assignments

Generative Methods

- Solve the inference problem of estimating the class-conditional densities $p(\boldsymbol{x} \mid C_k)$ for each class C_k

- Infer the prior class probabilities $p(C_k)$

- Use Bayes' theorem to find the class posterior probabilities:

$$p(C_k \mid \boldsymbol{x}) = \frac{p(\boldsymbol{x} \mid C_k)p(C_k)}{p(\boldsymbol{x})}$$

where

$$p(\boldsymbol{x}) = \sum_k p(\boldsymbol{x} \mid C_k)p(C_k)$$

- Use decision theory to determine class membership for each new input $\boldsymbol{x}$

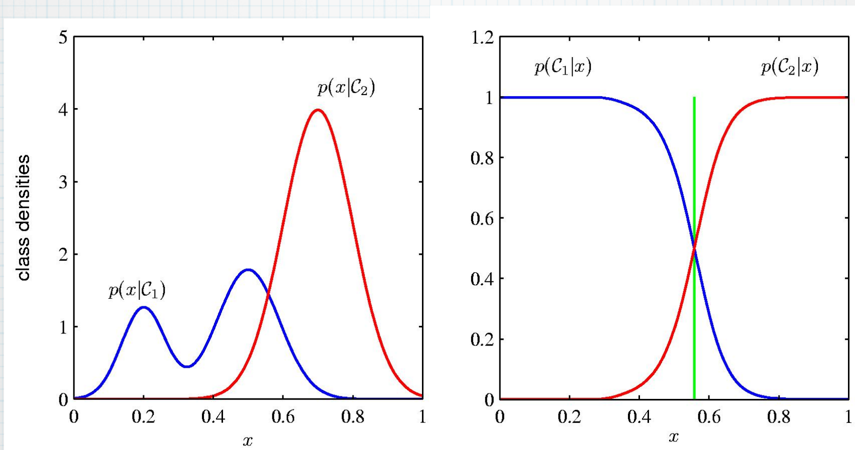
Discriminative Methods

- Solve directly the inference problem of estimating the class posterior probabilities $p(C_k \mid \boldsymbol{x})$
- Use decision theory to determine class membership for each new input $\boldsymbol{x}$

Discriminant Functions

- Find a function $f(\boldsymbol{x})$ which maps each input directly onto a class label. Probabilities play no role here.
- Use decision theory to determine class membership for each new input $\boldsymbol{x}$

Example



Linear Models for Classification

- ***Classification***: Given an input vector $\boldsymbol{x}$, assign it to one of K classes C_k where $k = 1,\ldots, K$

- The input space is divided in ***decision regions*** whose boundaries are called ***decision boundaries*** or ***decision surfaces***

- ***Linear models***: decision surfaces are linear functions of the input vector $\boldsymbol{x}$. They are defined by $(D-1)$-dimensional hyperplanes within the D-dimensional input space

Linear Models for Classification

- For regression: $y(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + w_0$

- For classification, we want to predict class labels, or more generally class posterior probabilities.

- We transform the linear function of $\boldsymbol{w}$ using a nonlinear function $f()$ so that

$$f\left(\boldsymbol{w}^T \boldsymbol{x} + w_0\right)$$

Generalized Linear Models

Linear Discriminant Functions

Two classes:

$$y(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + w_0$$

$$\begin{array}{ll} if \;\; y(\boldsymbol{x}) \geq 0 & assign \;\; \boldsymbol{x} \;\; to \;\; C_1 \\ otherwise & assign \;\; \boldsymbol{x} \;\; to \;\; C_2 \end{array}$$

Decision boundary: $y(\boldsymbol{x}) = 0$

Linear Discriminant Functions

Geometrical properties:

Decision boundary: $y(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + w_0 = 0$

Let $\boldsymbol{x}_1, \boldsymbol{x}_2$ be two points which lie on the decision boundary

$$y(\boldsymbol{x}_1) = \boldsymbol{w}^T \boldsymbol{x}_1 + w_0 = 0, y(\boldsymbol{x}_2) = \boldsymbol{w}^T \boldsymbol{x}_2 + w_0 = 0$$
$$\Rightarrow \boldsymbol{w}^T (\boldsymbol{x}_1 - \boldsymbol{x}_2) = 0$$

$\boldsymbol{w}$ represents the orthogonal direction to the decision boundary

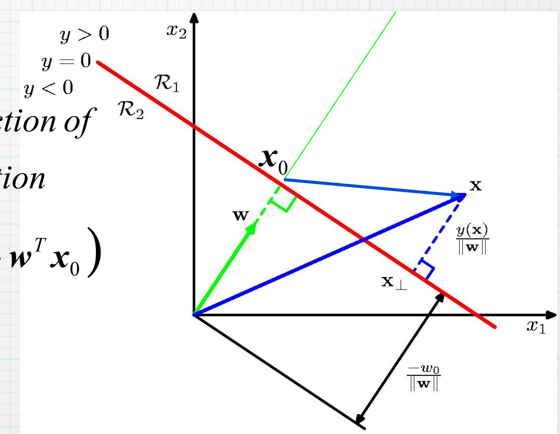
Geometrical properties (con' t)

$$\boldsymbol{w}^{*T} = \frac{\boldsymbol{w}^T}{\|\boldsymbol{w}\|}$$

$\boldsymbol{w}^{*T}(\boldsymbol{x} - \boldsymbol{x}_0)$ *is the projection of* $(\boldsymbol{x} - \boldsymbol{x}_0)$ *onto the* w^* *direction*

$$\frac{\boldsymbol{w}^T}{\|\boldsymbol{w}\|}(\boldsymbol{x} - \boldsymbol{x}_0) = \frac{1}{\|\boldsymbol{w}\|}\left(\boldsymbol{w}^T \boldsymbol{x} - \boldsymbol{w}^T \boldsymbol{x}_0\right)$$
$$= \frac{1}{\|\boldsymbol{w}\|}\left(\boldsymbol{w}^T \boldsymbol{x} + \boldsymbol{w}_0\right) = \frac{y(\boldsymbol{x})}{\|\boldsymbol{w}\|}$$

when $\boldsymbol{x} = \boldsymbol{0}$, $\frac{y(\boldsymbol{x})}{\|\boldsymbol{w}\|} = \frac{w_0}{\|\boldsymbol{w}\|}$

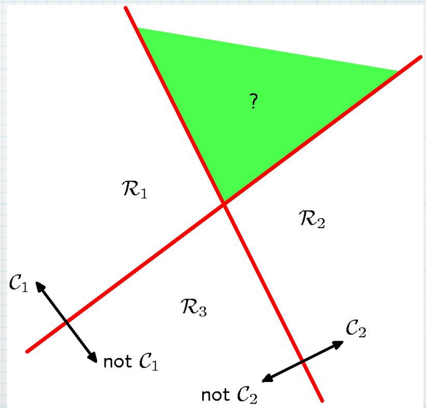


Signed orthogonal distance of the origin from the decision surface

Linear Discriminant Functions

Multiple classes

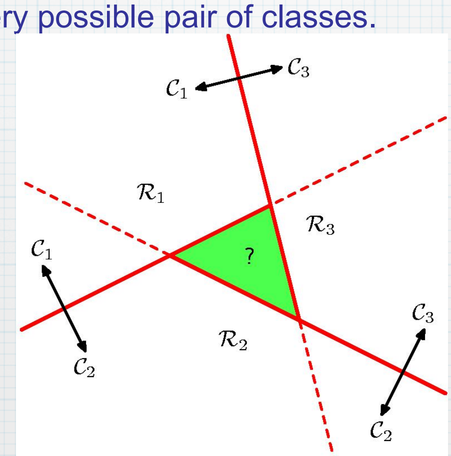
one-versus-the-rest: K-1 classifiers each of which solves a two-class problem of separating points of C_k from points not in that class



Linear Discriminant Functions

Multiple classes

one-versus-one: $K(K$-1)/2 binary discriminant functions, one for every possible pair of classes.



Linear Discriminant Functions

Multiple classes

Solution: consider a single *K-class* discriminant comprising *K* linear functions of the form

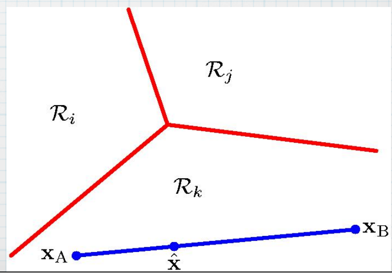
$$y_k(\boldsymbol{x}) = \boldsymbol{w}_k^T \boldsymbol{x} + w_{k0}$$

Assign a point $\boldsymbol{x}$ to class C_k if $y_k(x) > y_j(x)\ \forall j \neq k$

The decision boundary between class C_k and class C_j is given by

$$y_k(\boldsymbol{x}) = y_j(\boldsymbol{x})$$

$$\Rightarrow (\boldsymbol{w}_k - \boldsymbol{w}_j)^T \boldsymbol{x} + (w_{k0} - w_{j0}) = 0$$



Linear Discriminant Functions

Two approaches:

- ***Fisher's linear discriminant***
- ***Perceptron algorithm***

Fisher's Linear Discriminant

One way to view a linear classification model is in terms of *dimensionality reduction*.

Two class case:

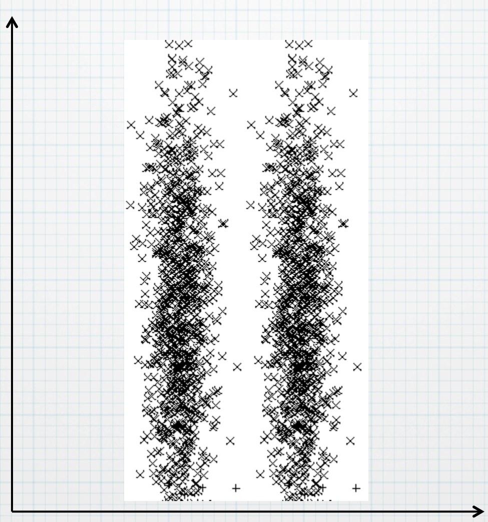
Suppose we project $\boldsymbol{x}$ onto one dimension:

$$y = \boldsymbol{w}^T \boldsymbol{x}$$

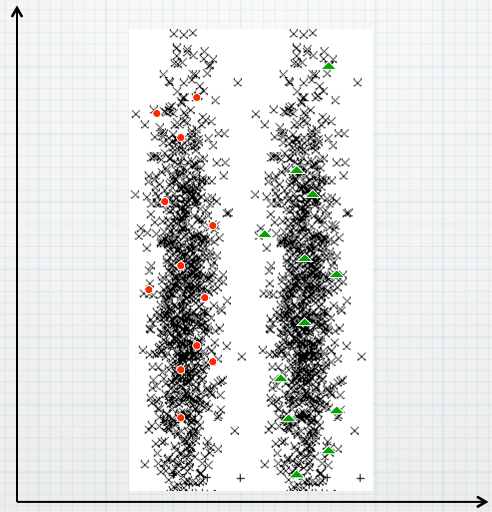
Set a threshold t

$$\begin{aligned} &if \;\; y \geq t \quad assign \;\; \boldsymbol{x} \;\; to \;\; C_1 \\ &otherwise \quad assign \;\; \boldsymbol{x} \;\; to \;\; C_2 \end{aligned}$$

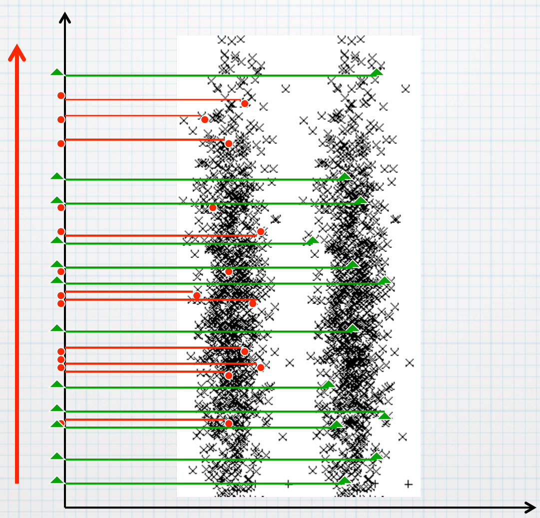
Example



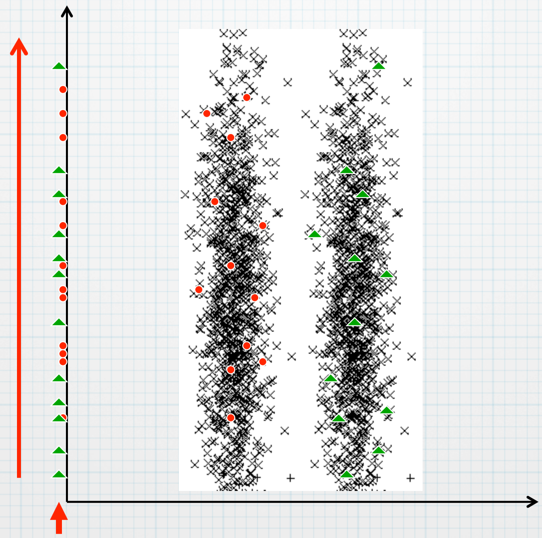
Example



Example

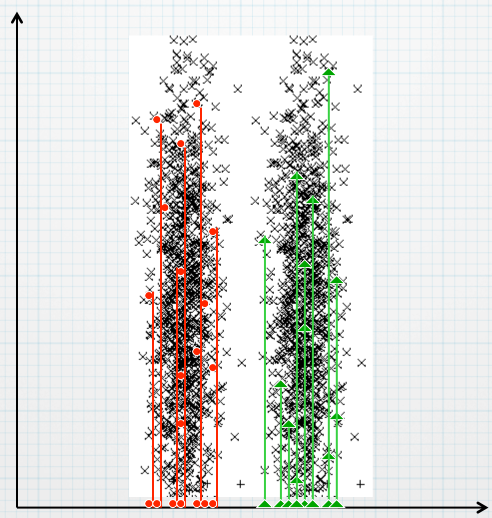


Example

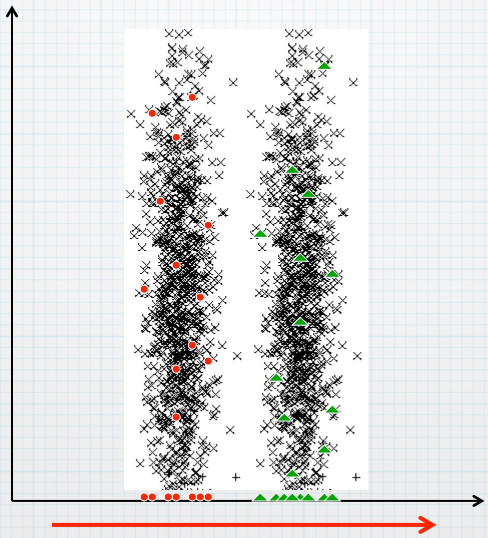


Confused mixture of samples from both classes

How about the horizontal dimension?



How about the horizontal dimension?



Projected samples are well separated now

Fisher's Linear Discriminant

- Find an orientation along which the projected samples are well separated;
- This is exactly the goal of linear discriminant analysis (LDA);
- In other words: we are after the linear projection that best separates the data, i.e. best discriminates data of different classes.

How can we find such discriminant direction?

LDA

$\{(\boldsymbol{x}_n, C_i)\}_{i=1}^{N} \quad \boldsymbol{x}_n \in \Re^q \quad C_i \in \{C_1, C_2\}$

- N_1 samples of class C_1
- N_2 samples of class C_2
- Consider $\boldsymbol{w} \in \Re^q$ with $\|\boldsymbol{w}\| = 1$
- Then: $\boldsymbol{w}^T\boldsymbol{x}$ is the projection of $\boldsymbol{x}$ along the direction of $\boldsymbol{w}$
- We want the projections $\boldsymbol{w}^T\boldsymbol{x}$ where $\boldsymbol{x} \in C_1$ separated from the projections $\boldsymbol{w}^T\boldsymbol{x}$ where $\boldsymbol{x} \in C_2$

LDA

- A measure of the separation between the projected points is the difference of the sample means:

$$\boldsymbol{m}_i = \frac{1}{N_i}\sum_{\boldsymbol{x}\in C_i}\boldsymbol{x}$$ Sample mean of class C_i

$$m_i = \frac{1}{N_i}\sum_{\boldsymbol{x}\in C_i}\boldsymbol{w}^T\boldsymbol{x} = \boldsymbol{w}^T\boldsymbol{m}_i$$ Sample mean for the projected points

$$\Rightarrow \quad |m_1 - m_2| = |\boldsymbol{w}^T(\boldsymbol{m}_1 - \boldsymbol{m}_2)|$$

We wish to make the above difference as large as we can. In addition…

LDA

- To obtain good separation of the projected data we really want the difference between the means to be large relative to some measure of the standard deviation of each class:

$$s_i^2 = \sum_{x \in C_i} \left(\boldsymbol{w}^T \boldsymbol{x} - m_i \right)^2$$

Scatter for the projected samples of class C_i

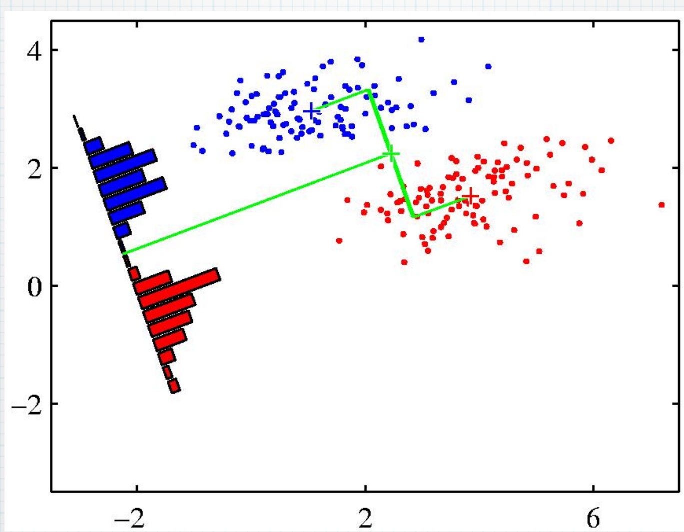
$$s_1^2 + s_2^2$$

Total **within-class scatter** of the projected samples

$$\arg\max_{w} \frac{|m_1 - m_2|^2}{s_1^2 + s_2^2}$$

Fisher linear discriminant analysis

LDA



LDA

$$J(w) = \frac{|m_1 - m_2|^2}{s_1^2 + s_2^2}$$

To obtain $J(\boldsymbol{w})$ as an explicit function of $\boldsymbol{w}$ we define the following matrices :

$$S_i = \sum_{x \in C_i} (\boldsymbol{x} - \boldsymbol{m}_i)(\boldsymbol{x} - \boldsymbol{m}_i)^T$$

$S_W = S_1 + S_2$ **Within-class scatter matrix**

Then:

$$s_i^2 = \sum_{x \in C_i} \left(\boldsymbol{w}^T \boldsymbol{x} - m_i\right)^2 = \sum_{x \in C_i} \left(\boldsymbol{w}^T \boldsymbol{x} - \boldsymbol{w}^T \boldsymbol{m}_i\right)^2$$

$$= \sum_{x \in C_i} \boldsymbol{w}^T (\boldsymbol{x} - \boldsymbol{m}_i)(\boldsymbol{x} - \boldsymbol{m}_i)^T \boldsymbol{w} = \boldsymbol{w}^T S_i \boldsymbol{w}$$

LDA

$$J(w) = \frac{|m_1 - m_2|^2}{s_1^2 + s_2^2}$$

So: $s_1^2 = \boldsymbol{w}^T S_1 \boldsymbol{w}$ and $s_2^2 = \boldsymbol{w}^T S_2 \boldsymbol{w}$

Thus: $s_1^2 + s_2^2 = \boldsymbol{w}^T S_1 \boldsymbol{w} + \boldsymbol{w}^T S_2 \boldsymbol{w} =$

$$\boldsymbol{w}^T (S_1 + S_2) \boldsymbol{w} = \boldsymbol{w}^T S_W \boldsymbol{w}$$

Similarly:

$$(m_1 - m_2)^2 = \left(\boldsymbol{w}^T \boldsymbol{m}_1 - \boldsymbol{w}^T \boldsymbol{m}_2\right)^2 =$$

$$\boldsymbol{w}^T (\boldsymbol{m}_1 - \boldsymbol{m}_2)(\boldsymbol{m}_1 - \boldsymbol{m}_2)^T \boldsymbol{w} =$$

$$\boldsymbol{w}^T S_B \boldsymbol{w}$$

where $S_B = (\boldsymbol{m}_1 - \boldsymbol{m}_2)(\boldsymbol{m}_1 - \boldsymbol{m}_2)^T$ **Between-class scatter matrix**

LDA

We have obtained:

$$s_1^2 + s_2^2 = \boldsymbol{w}^T S_W \boldsymbol{w}$$

$$(m_1 - m_2)^2 = \boldsymbol{w}^T S_B \boldsymbol{w}$$

$$\Rightarrow \quad J(\boldsymbol{w}) = \frac{|m_1 - m_2|^2}{s_1^2 + s_2^2} = \frac{\boldsymbol{w}^T S_B \boldsymbol{w}}{\boldsymbol{w}^T S_W \boldsymbol{w}}$$

$$\arg\max_{\boldsymbol{w}} \frac{\boldsymbol{w}^T S_B \boldsymbol{w}}{\boldsymbol{w}^T S_W \boldsymbol{w}}$$

LDA

$$J(\boldsymbol{w}) = \frac{\boldsymbol{w}^T S_B \boldsymbol{w}}{\boldsymbol{w}^T S_W \boldsymbol{w}}$$

$J(\boldsymbol{w})$ *is maximized when* $\left(\boldsymbol{w}^T S_B \boldsymbol{w}\right) S_W \boldsymbol{w} = \left(\boldsymbol{w}^T S_W \boldsymbol{w}\right) S_B \boldsymbol{w}$

We observe that:

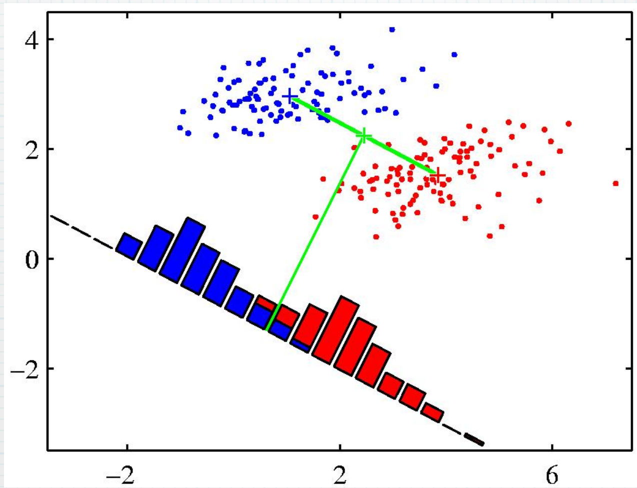
$$S_B \boldsymbol{w} = (\boldsymbol{m}_1 - \boldsymbol{m}_2)(\boldsymbol{m}_1 - \boldsymbol{m}_2)^T \boldsymbol{w}$$

scalar

Always in the direction of $(\boldsymbol{m}_1 - \boldsymbol{m}_2)$

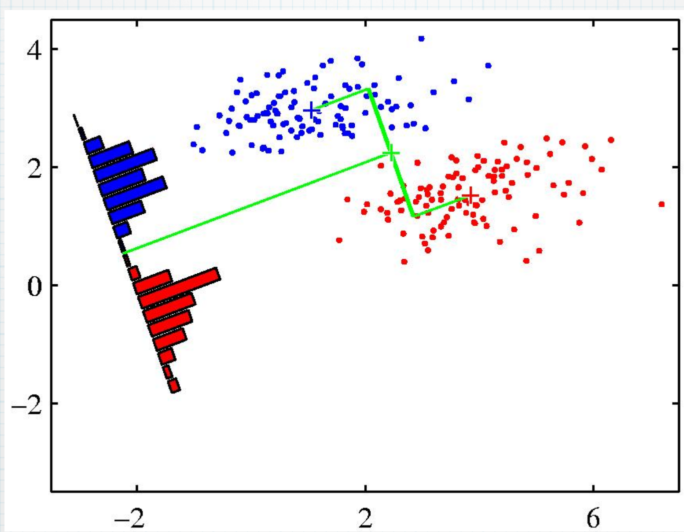
$$\Rightarrow \quad \boldsymbol{w} = S_W^{-1}(\boldsymbol{m}_1 - \boldsymbol{m}_2)$$

LDA



Projection onto the line joining the class means

LDA



Solution of LDA

LDA

$$w = S_W^{-1}(m_1 - m_2)$$

• Gives the linear function with the maximum ratio of between-class scatter to within-class scatter.

• The problem, e.g. classification, has been reduced from a *q*-dimensional problem to a more manageable one-dimensional problem.

• Optimal for multivariate normal class conditional densities.

LDA

• The analysis can be extended to multiple classes.

• LDA is a ***linear*** technique for dimensionality reduction: it projects the data along directions that can be expressed as ***linear combination*** of the input features.

• The "appropriate" transformation depends on the data and on the **task** we want to perform on the data. Note that LDA uses class labels.

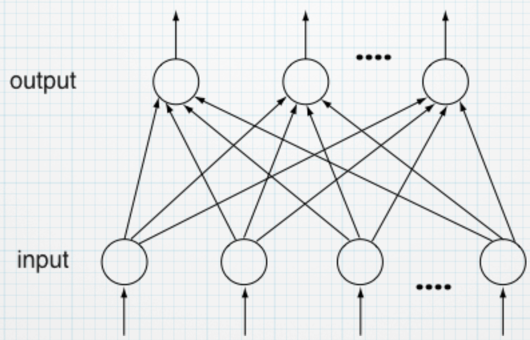
• Non-linear extensions of LDA exist (e.g., generalized LDA).

The Perceptron Algorithm

Perceptron (Frank Rosenblatt, 1957)

- First learning algorithm for neural networks;
- Originally introduced for character classification, where each character is represented as an image;

Perceptron (contd.)



Total input to output node: $\sum_{j=1}^{n} w_j x_j$

Output unit performs the function: (activation function):

$$H(x) = \begin{cases} 1 & \textbf{if } x \geq 0 \\ 0 & \textbf{if } x < 0 \end{cases}$$

Perceptron: Learning Algorithm

- **Goal**: we want to define a learning algorithm for the weights in order to compute a mapping from the inputs to the outputs;
- **Example**: two class character recognition problem.
  - **Training set**: set of images representing either the character 'a' or the character 'b' (supervised learning);
  - **Learning Task**: Learn the weights so that when a new unlabelled image comes in, the network can predict its label.
  - Settings:
    - Class 'a' ➔ 1 (class C1)
    - Class 'b' ➔ 0 (class C2)
    - n input units (intensity level of a pixel)
    - 1 output unit

The perceptron needs to learn $f : \Re^n \rightarrow \{0,1\}$

Perceptron: Learning Algorithm

The algorithm proceeds as follows:

- Initial random setting of weights;
- The input is a random sequence $\{\boldsymbol{x}_k\}_{k \in \aleph}$
- For each element of class C1, if output = 1 (correct) do nothing, otherwise update weights;
- For each element of class C2, if output = 0 (correct) do nothing, otherwise update weights.

Perceptron: Learning Algorithm

A bit more formally:

$\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ $\boldsymbol{w} = (w_1, w_2, \ldots, w_n)$

θ **:** Threshold of the output unit

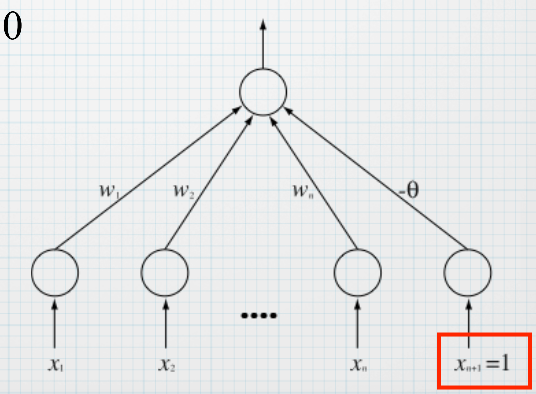
$\boldsymbol{w}\boldsymbol{x}^T = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n$

Output is 1 if $\boldsymbol{w}\boldsymbol{x}^T - \theta \geq 0$

To eliminate the explicit dependence on θ:

Output is 1 if:

$$\hat{\boldsymbol{w}}\hat{\boldsymbol{x}}^T = \sum_{i=1}^{n+1} w_i x_i \geq 0$$



Perceptron: Learning Algorithm

- We want to learn values of the weights so that the perceptron correctly discriminate elements of C1 from elements of C2:
- Given x in input, if x is classified correctly, weights are unchanged, otherwise:

$$w' = \begin{cases} w + x & \text{if an element of class } C_1 \text{ (1) was classified as in } C_2 \\ w - x & \text{if an element of class } C_2 \text{ (0) was classified as in } C_1 \end{cases}$$

Perceptron: Learning Algorithm

$$w' = \begin{cases} w + x & \text{if an element of class } C_1 \text{ (1) was classified as in } C_2 \\ w - x & \text{if an element of class } C_2 \text{ (0) was classified as in } C_1 \end{cases}$$

- 1st case: $x \in C_1$ and was classified in C_2

The correct answer is 1, which corresponds to: $\hat{w}\hat{x}^T \geq 0$

We have instead: $\hat{w}\hat{x}^T < 0$

We want to get closer to the correct answer: $wx^T < w'x^T$

$$wx^T < w'x^T \quad \text{iff} \quad wx^T < (w + x)x^T$$

$$(w + x)x^T = wx^T + xx^T = wx^T + \|x\|^2$$

$$\text{because } \|x\|^2 \geq 0, \text{ the condition is verified}$$

Perceptron: Learning Algorithm

$$w' = \begin{cases} w + x & \text{if an element of class } C_1\ (1) \text{ was classified as in } C_2 \\ w - x & \text{if an element of class } C_2\ (0) \text{ was classified as in } C_1 \end{cases}$$

- 2nd case: $x \in C_2$ *and was classified in* C_1

The correct answer is 0, which corresponds to: $\hat{w}\hat{x}^T < 0$

We have instead: $\hat{w}\hat{x}^T \geq 0$

We want to get closer to the correct answer: $wx^T > w'x^T$

$$wx^T > w'x^T \quad \text{iff} \quad wx^T > (w - x)x^T$$

$$(w - x)x^T = wx^T - xx^T = wx^T - \|x\|^2$$

because $\|x\|^2 \geq 0$, *the condition is verified*

The previous rule allows the network to get closer to the correct answer when it performs an error.

Perceptron: Learning Algorithm

- **In summary**:

1. A random sequence $x_1, x_2, \cdots, x_k, \cdots$ is generated such that $x_i \in C_1 \cup C_2$
2. If x_k is correctly classified, then $w_{k+1} = w_k$ otherwise

$$w_{k+1} = \begin{cases} w_k + x_k & \text{if } x_k \in C_1 \\ w_k - x_k & \text{if } x_k \in C_2 \end{cases}$$

Perceptron: Learning Algorithm

Does the learning algorithm converge?

Convergence theorem: Regardless of the initial choice of weights, if the two classes are linearly separable, i.e. there exist $\boldsymbol{w}$ s.t.

$$\begin{cases} \hat{\boldsymbol{w}}\hat{\boldsymbol{x}}^T \geq 0 \ \ \boldsymbol{if} \ \ \boldsymbol{x} \in C_1 \\ \hat{\boldsymbol{w}}\hat{\boldsymbol{x}}^T < 0 \ \ \boldsymbol{if} \ \ \boldsymbol{x} \in C_2 \end{cases}$$

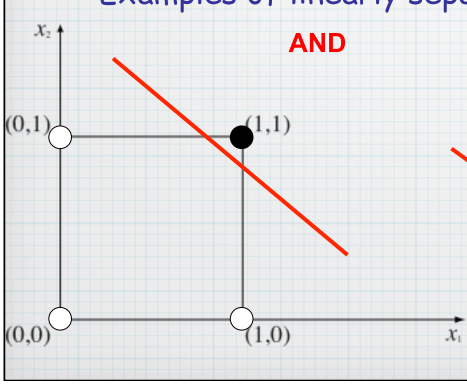
then the learning rule will find such solution after a finite number of steps.

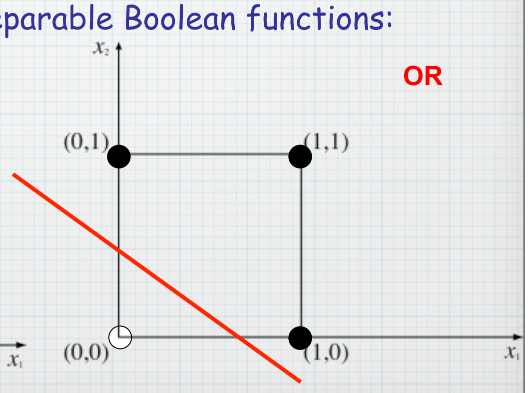
Representational Power of Perceptrons

- Marvin Minsky and Seymour Papert, "Perceptrons" 1969:

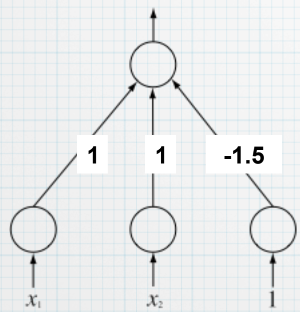
  "The perceptron can solve only problems with linearly separable classes."

- Examples of linearly separable Boolean functions:

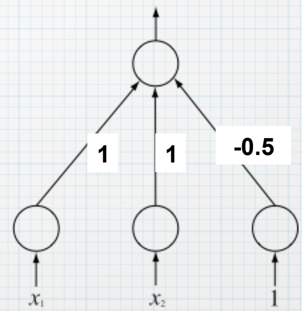




Representational Power of Perceptrons



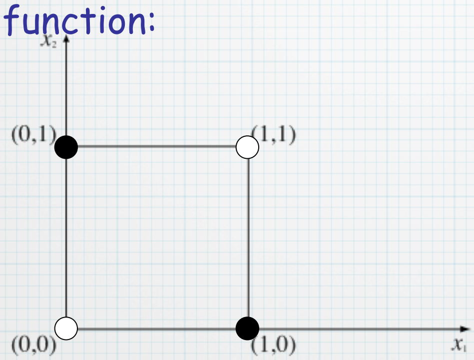
Perceptron that computes the AND function



Perceptron that computes the OR function

Representational Power of Perceptrons

- Example of a non linearly separable Boolean function:



EX-OR

The EX-OR function **cannot** be computed by a perceptron