# Artificial Neural Networks

## Lecture 2: Linear Discriminants


Tudor Berariu

*tudor.berariu@gmail.com*

Faculty of Automatic Control and Computers

University Politehnica of Bucharest



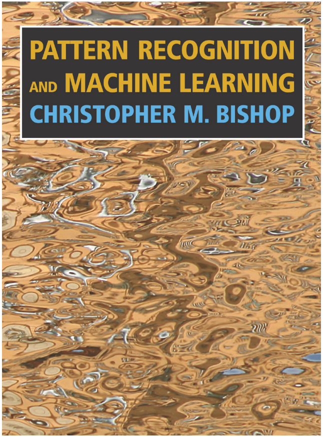
Lecture : $14^{th}$ of October, 2015

Last Updated: $14^{th}$ of October, 2015

# Today's Outline

1. The Classification Problem
2. Linear Discriminant Functions
3. The Perceptron

# Resources




Christopher M. Bishop, *Pattern recognition and machine learning (information science and statistics)*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006

Section 4.1: *Discriminant functions*

# Today's Outline

# Today's Outline

1. The Classification Problem
   - Linear classifiers
   - 2 classes vs. $K$ classes
2. Linear Discriminant Functions
3. The Perceptron

# Classification

**Definition**

Given a data set $\mathbf{X}$ containing $N$ examples $\mathbf{x}^{(i)}, 1 \leq i \leq N$ in a $D$-dimensional space, each labeled with one of the $K$ discrete classes $\mathcal{C}_1, \mathcal{C}_2, \ldots \mathcal{C}_K$, build a model that can classify new examples.

# Classification

**Definition**

Given a data set $\mathbf{X}$ containing $N$ examples $\mathbf{x}^{(i)}, 1 \leq i \leq N$ in a $D$-dimensional space, each labeled with one of the $K$ discrete classes $\mathcal{C}_1, \mathcal{C}_2, \ldots \mathcal{C}_K$, build a model that can classify new examples.

- Notations used:

$$
\begin{aligned}
\mathbf{X} &\text{ - the data set (collection of } N \text{ examples);} \\
\mathbf{x}^{(i)} &\text{ - the } i^{\text{th}} \text{ example in the data set;} \\
&\quad \mathbf{X} = \{(\mathbf{x}^{(1)}, t^{(1)}), \ldots, (\mathbf{x}^{(N)}, t^{(N)})\} \\
x_j^{(i)} &\text{ - } j^{\text{th}} \text{ attribute of the } i^{\text{th}} \text{ example: } \mathbf{x}^{(i)} = [x_1^{(i)}, \ldots, x_D^{(i)}] \\
t^{(i)} &\text{ - the scalar label of the } i^{\text{th}} \text{ example} \\
&\quad (t^{(i)} \in \{\mathcal{C}_1, \ldots, \mathcal{C}_K\}); \\
\mathbf{t}^{(i)} &\text{ - the label vector of the } i^{\text{th}} \text{ example}
\end{aligned}
$$

# Classification

## Definition

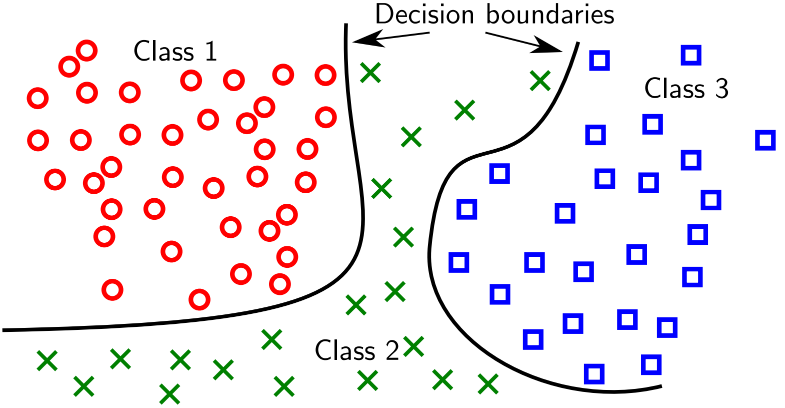
Given a data set $\mathbf{X}$ containing $N$ examples $\mathbf{x}^{(i)}, 1 \leq i \leq N$ in a $D$-dimensional space, each labeled with one of the $K$ discrete classes $\mathcal{C}_1, \mathcal{C}_2, \ldots \mathcal{C}_K$, build a model that can classify new examples.

- Approaches to the classification problem:
  - construct discriminant functions
  - compute the conditional probability $p(\mathcal{C}_k|\mathbf{x})$
    - model them directly (e.g. using parametric models)
    - learn a generative model $P(\mathbf{x}|\mathcal{C}_k)$ and use Bayes:

$$p(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{p(\mathbf{x})} \tag{1}$$

# Decision boundaries

- The goal is to divide the input space into **decision regions**.
- Classes are separated by **decision boundaries** or **decision surfaces**.



# Linear classification models

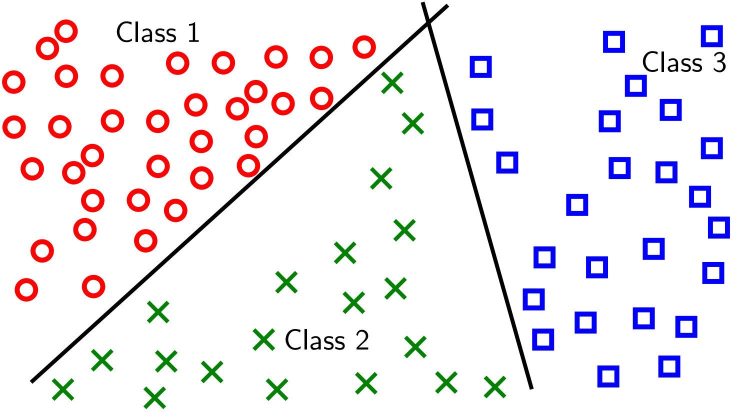
**Definition**

Models that create decision boundaries that are linear functions of the input vector $\mathbf{x}$ (therefore, they are $(D-1)$-dimensional hyperplanes in the input space) are called **linear models**.

# Linear separability

**Definition**

A data set which can be separated perfectly using linear decision boundaries is called **linearly separable**.



# Generalized linear models

- Generalized linear models:

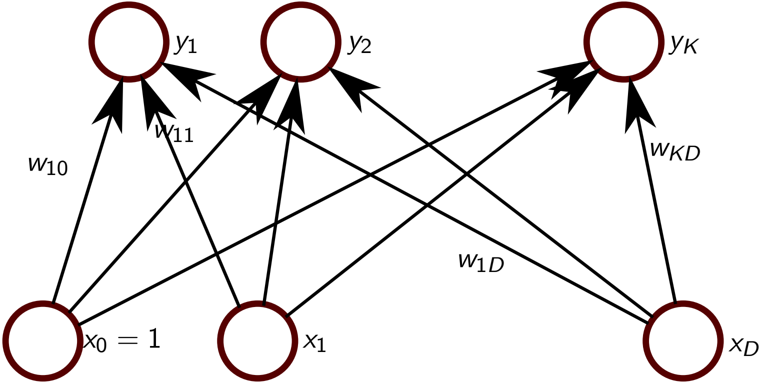
$$y(\mathbf{x}) = f(\mathbf{w}^{\mathsf{T}}\mathbf{x} + w_0) \tag{2}$$

  $f(\cdot)$ nonlinear activation function

- Decision boundaries are linear functions of $\mathbf{x}$ even if the function $f(\cdot)$ is nonlinear.
- In today's lecture: linear discriminants (decision surfaces are hyperplanes).

[Bis06, pag. 180]

# Single Layer Network



# Today's Outline

# The 2-class problem

- A discriminant function:

$$y(\mathbf{x}) = \sum_{j=1}^{D} w_j x_j + w_0 = \mathbf{w}^{\mathsf{T}}\mathbf{x} + w_0 \tag{3}$$

  - $\mathbf{x} \in \mathcal{C}_1$ if $y(\mathbf{x}) \geq 0$
  - $\mathbf{x} \in \mathcal{C}_2$ if $y(\mathbf{x}) < 0$
- $\mathbf{w}$ determines the orientation of the decision boundary
- Notations:

$$\begin{aligned} \tilde{\mathbf{w}} &= (w_0, \mathbf{w}^{\mathsf{T}})^{\mathsf{T}} \\ \tilde{\mathbf{x}} &= (1, \mathbf{x}^{\mathsf{T}})^{\mathsf{T}} \\ &\text{so... } y(\mathbf{x}) = \tilde{\mathbf{w}}^{\mathsf{T}}\tilde{\mathbf{x}} \end{aligned}$$

# A linear discriminant function

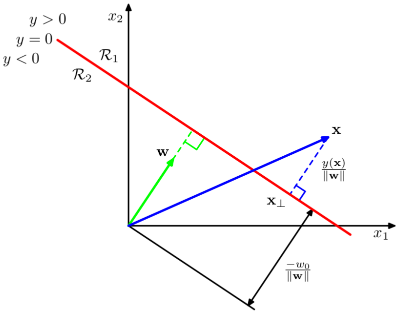
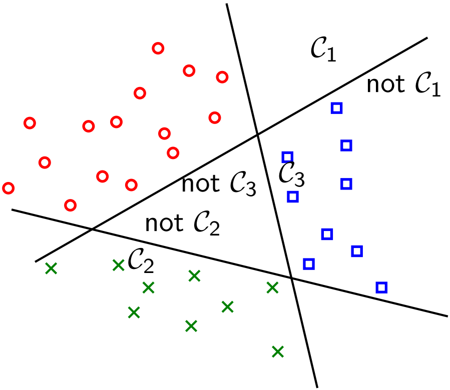



Figure: Illustration of the geometry of a linear discriminant function in two dimensions. The decision surface, shown in red, is perpendicular to **x**, and its displacement from the origin is controlled by the bias parameter $w_0$. Also, the signed orthogonal distance of a general point **x** from the decision surface is given by $y(\mathbf{x})/||\mathbf{w}||$. (Taken from [Bis06, pag. 182])

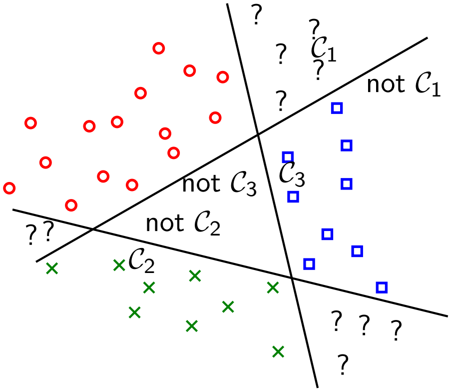
# $K$ classes

- $K$ one-versus-all classifiers



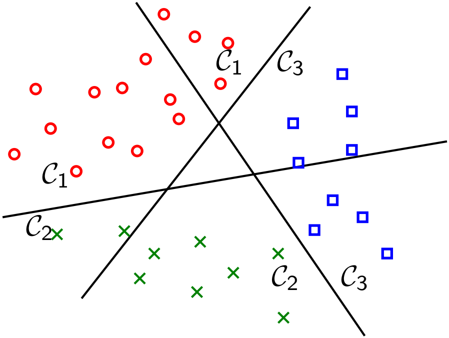
# $K$ classes

- $K$ one-versus-all classifiers



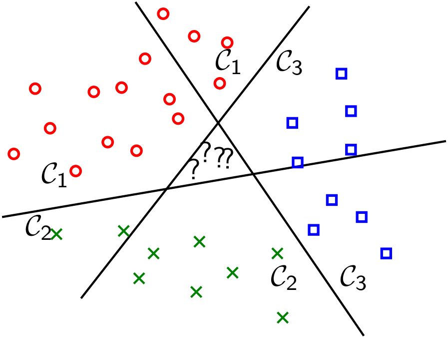
# $K$ classes

- $K$ one-versus-all classifiers
- $K \cdot (K-1)/2$ one-versus-one classifiers



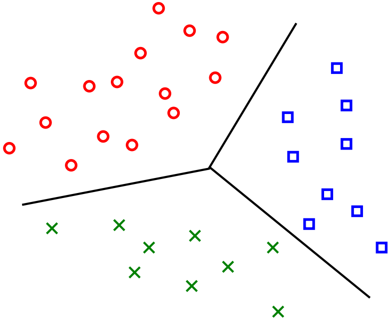
# $K$ classes

- $K$ one-versus-all classifiers
- $K \cdot (K-1)/2$ one-versus-one classifiers



# $K$ classes

- $K$ one-versus-all classifiers
- $K \cdot (K-1)/2$ one-versus-one classifiers
- single classifier with $K$ linear functions



# $K$ classes

- $K$ one-versus-all classifiers
- $K \cdot (K-1)/2$ one-versus-one classifiers
- single classifier with $K$ linear functions

$$y_k(\mathbf{x}) = \mathbf{w}_k^{\mathsf{T}} \mathbf{x} + w_{k0} \tag{4}$$

  - $\mathbf{x}$ in class $k$ if $y_k(\mathbf{x}) > y_j(\mathbf{x}) \quad \forall j \in \{1, \cdots, K\}, j \neq k$
  - decision regions are always **connected** and **convex** (proof in [Bis06, pag. 184])

# Let's write some code...

- Demo: generating a synthetic linearly separable data set

# Today's Outline

# Today's Outline

1. The Classification Problem
2. Linear Discriminant Functions
   - Least Squares
   - Fisher's linear discriminant
3. The Perceptron

# Linear model

- Each class $(1 \leq k \leq K)$ is described by a separate linear model:

$$y_k(\mathbf{x}) = \mathbf{w}_k^\top \mathbf{x} + w_{k0} \tag{5}$$

- Using the matrix notation:

$$\underbrace{\mathbf{Y}}_{N \times K} = \underbrace{\tilde{\mathbf{X}}}_{N \times (D+1)} \underbrace{\tilde{\mathbf{W}}}_{(D+1) \times K} \tag{6}$$

where $\tilde{\mathbf{W}}$ is a $(D+1) \times K$ matrix:

$$\begin{matrix} w_{10} & w_{20} & \dots & w_{K0} \\ w_{11} & a_{21} & \dots & a_{K1} \\ \vdots & \vdots & \ddots & \vdots \\ w_{1D} & w_{2D} & \dots & w_{KD} \end{matrix}$$

# Minimizing the sum-of-squares error

- The sum-of-squares error function:

$$E(\tilde{\mathbf{W}}) = \frac{1}{2} \sum_{n=1}^{N} \sum_{k=1}^{K} (y_k(\tilde{\mathbf{x}}^{(n)}, \tilde{\mathbf{w}}_k) - t_k^{(n)})^2 \tag{7}$$

# Minimizing the sum-of-squares error

- The sum-of-squares error function:

$$E(\tilde{\mathbf{W}}) = \frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{K}(y_k(\tilde{\mathbf{x}}^{(n)}, \tilde{\mathbf{w}}_k) - t_k^{(n)})^2 \tag{7}$$

- The error is a quadratic function of the weights.
- Its derivates w.r.t the weights will be linear functions of the weights.

# Minimizing the sum-of-squares error

- The sum-of-squares error function:

$$E(\tilde{\mathbf{W}}) = \frac{1}{2} \sum_{n=1}^{N} \sum_{k=1}^{K} (y_k(\tilde{\mathbf{x}}^{(n)}, \tilde{\mathbf{w}}_k) - t_k^{(n)})^2 \tag{7}$$

- The normal equations:

$$\sum_{n=1}^{N} \left( \sum_{l=0}^{D} w_{kl} x_l^{(n)} - t_k^{(n)} \right) x_j^{(n)} = 0, \qquad \forall 1 \leq j \leq K \tag{8}$$

# SSE in matrix form

- The sum-of-squares error function:

$$E(\tilde{\mathbf{W}}) = \frac{1}{2}\mathbf{Tr}\left\{\left(\tilde{\mathbf{X}}\tilde{\mathbf{W}} - \mathbf{T}\right)\left(\tilde{\mathbf{X}}\tilde{\mathbf{W}} - \mathbf{T}\right)^{\mathsf{T}}\right\} =$$

# SSE in matrix form

- The sum-of-squares error function:

$$E(\tilde{\mathbf{W}}) = \frac{1}{2}\mathbf{Tr}\left\{\left(\tilde{\mathbf{X}}\tilde{\mathbf{W}} - \mathbf{T}\right)\left(\tilde{\mathbf{X}}\tilde{\mathbf{W}} - \mathbf{T}\right)^{\mathsf{T}}\right\} =$$

$$\frac{1}{2}\left(\mathrm{Tr}\left\{\tilde{\mathbf{X}}\tilde{\mathbf{W}}\tilde{\mathbf{W}}^{\mathsf{T}}\tilde{\mathbf{X}}^{\mathsf{T}}\right\} - \mathrm{Tr}\left\{\tilde{\mathbf{X}}\tilde{\mathbf{W}}\mathbf{T}^{\mathsf{T}}\right\} - \mathrm{Tr}\left\{\mathbf{T}\tilde{\mathbf{W}}^{\mathsf{T}}\tilde{\mathbf{X}}^{\mathsf{T}}\right\} + \mathrm{Tr}\left\{\mathbf{T}\mathbf{T}^{\mathsf{T}}\right\}\right)$$

# SSE in matrix form

- The sum-of-squares error function:

$$E(\tilde{\mathbf{W}}) = \frac{1}{2}\mathbf{Tr}\left\{\left(\tilde{\mathbf{X}}\tilde{\mathbf{W}} - \mathbf{T}\right)\left(\tilde{\mathbf{X}}\tilde{\mathbf{W}} - \mathbf{T}\right)^{\mathsf{T}}\right\} =$$

$$\frac{1}{2}\left(\mathrm{Tr}\left\{\tilde{\mathbf{X}}\tilde{\mathbf{W}}\tilde{\mathbf{W}}^{\mathsf{T}}\tilde{\mathbf{X}}^{\mathsf{T}}\right\} - \mathrm{Tr}\left\{\tilde{\mathbf{X}}\tilde{\mathbf{W}}\mathbf{T}^{\mathsf{T}}\right\} - \mathrm{Tr}\left\{\mathbf{T}\tilde{\mathbf{W}}^{\mathsf{T}}\tilde{\mathbf{X}}^{\mathsf{T}}\right\} + \mathrm{Tr}\left\{\mathbf{T}\mathbf{T}^{\mathsf{T}}\right\}\right)$$

- The derivate w.r.t. $\tilde{\mathbf{W}}$:

$$\begin{aligned}\frac{\partial E(\tilde{\mathbf{W}})}{\partial \tilde{\mathbf{W}}} &= \frac{1}{2}\left(\tilde{\mathbf{X}}^{\mathsf{T}}\tilde{\mathbf{X}}\tilde{\mathbf{W}} + \tilde{\mathbf{X}}^{\mathsf{T}}\tilde{\mathbf{X}}\tilde{\mathbf{W}} - \tilde{\mathbf{X}}^{\mathsf{T}}\mathbf{T} - \tilde{\mathbf{X}}^{\mathsf{T}}\mathbf{T}\right) \\ &= \tilde{\mathbf{X}}^{\mathsf{T}}\tilde{\mathbf{X}}\tilde{\mathbf{W}} - \tilde{\mathbf{X}}^{\mathsf{T}}\mathbf{T}\end{aligned}$$

See matrix derivation rules here: `http://cal.cs.illinois.edu/~johannes/research/matrix%20calculus.pdf`

# The pseudoinverse solution

- We minimize $E(\tilde{\mathbf{W}})$ by setting $\frac{\partial E(\tilde{\mathbf{W}})}{\partial \tilde{\mathbf{W}}}$ to zero:

$$
\begin{aligned}
\frac{\partial E(\tilde{\mathbf{W}})}{\partial \tilde{\mathbf{W}}} = 0 \quad &\iff \quad \tilde{\mathbf{X}}^\top \tilde{\mathbf{X}} \tilde{\mathbf{W}} - \tilde{\mathbf{X}}^\top \mathbf{T} = 0 \\
&\iff \quad \tilde{\mathbf{X}}^\top \tilde{\mathbf{X}} \tilde{\mathbf{W}} = \tilde{\mathbf{X}}^\top \mathbf{T} \\
&\iff \quad \tilde{\mathbf{W}} = \left(\tilde{\mathbf{X}}^\top \tilde{\mathbf{X}}\right)^{-1} \tilde{\mathbf{X}}^\top \mathbf{T} \\
&\iff \quad \tilde{\mathbf{W}} = \tilde{\mathbf{X}}^\dagger \mathbf{T}
\end{aligned}
$$

# Problems

- lack of robustness for outliers
- penalizes *too correct* examples
- poor results when target vectors don't have a normal distribution

# Let's write some code...

- Demo: computing the weights using least-squares

# Today's Outline

1. The Classification Problem
2. Linear Discriminant Functions
   - Least Squares
   - Fisher's linear discriminant
3. The Perceptron

# The 2-class problem

- The mean vectors:

$$
\begin{aligned}
m_1 &= \frac{1}{N_1} \sum_{n \in \mathcal{C}_1} \mathbf{x}_n \\
m_2 &= \frac{1}{N_2} \sum_{n \in \mathcal{C}_2} \mathbf{x}_n
\end{aligned}
$$

- **Separate the projected means**, i.e. maximize $m_2 - m_1 = \mathbf{w}^\mathsf{T}(m_2 - m_1)$

# The 2-class problem

- The mean vectors:

$$\begin{aligned} m_1 &= \frac{1}{N_1} \sum_{n \in \mathcal{C}_1} \mathbf{x}_n \\ m_2 &= \frac{1}{N_2} \sum_{n \in \mathcal{C}_2} \mathbf{x}_n \end{aligned}$$

- **Separate the projected means**, i.e. maximize $m_2 - m_1 = \mathbf{w}^\mathsf{T}(m_2 - m_1)$
- Problem: this could be achieved by increasing $\mathbf{w}$

# An example

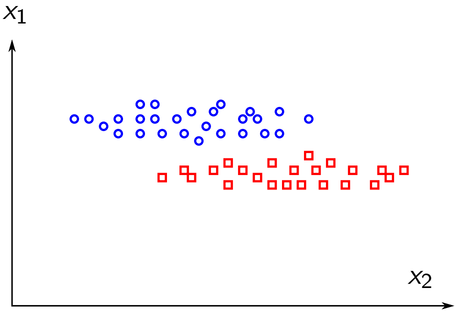



Figure: Projecting means

# An example

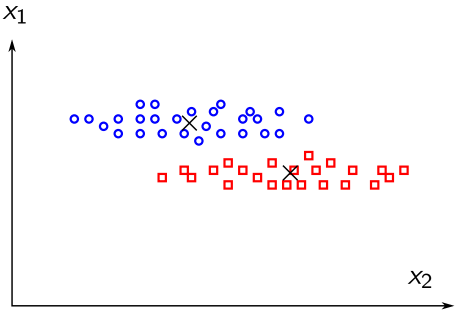



Figure: Projecting means

# An example

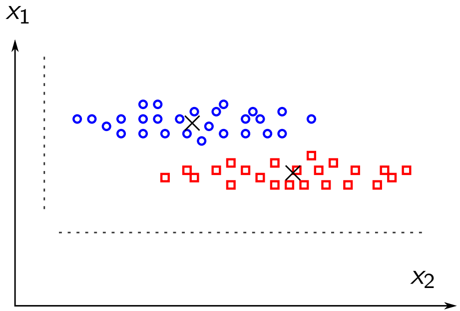



Figure: Projecting means

# An example

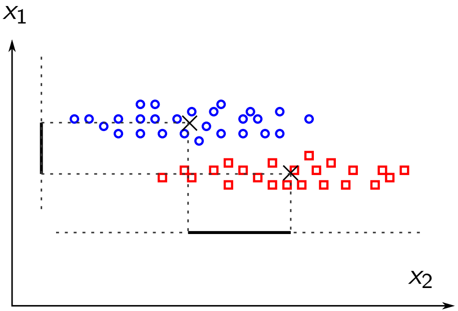



Figure: Projecting means

# An example

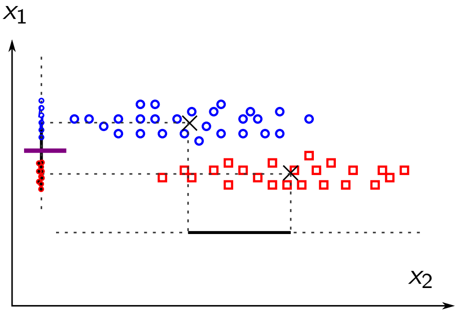



Figure: Projecting means

# Intuition on Fisher's criterion

**Definition**

Maximize a function that will give a large separation between the projected class means while als giving a small variance within each class.

- the within-class variance of the projected data:

$$s_k^2 = \sum_{n \in \mathcal{C}_k} (y_n - m_k)^2 \tag{9}$$

**Definition**

Fisher's criterion is given by:

$$J(w) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2} \tag{10}$$

# Finding $\mathbf{w}$

- Rewriting Fisher's criterion:

$$J(\mathbf{w}) = \frac{\mathbf{w}^\mathsf{T}\mathbf{S}_B\mathbf{w}}{\mathbf{w}^\mathsf{T}\mathbf{S}_W\mathbf{w}}$$

- The between-class covariance:

$$\mathbf{S}_B = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^\mathsf{T}$$

- The within-class covariance:

$$\mathbf{S}_W = \sum_{n \in \mathcal{C}_1} (\mathbf{x}^{(n)} - \mathbf{m}_1)(\mathbf{x}^{(n)} - \mathbf{m}_1)^\mathsf{T} + \sum_{n \in \mathcal{C}_2} (\mathbf{x}^{(n)} - \mathbf{m}_2)(\mathbf{x}^{(n)} - \mathbf{m}_2)^\mathsf{T}$$

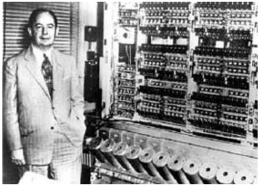
- Hence, the direction of $\mathbf{w}$

$$\mathbf{w} \propto \mathbf{S}_W^{-1}(\mathbf{m}_2 - \mathbf{m}_1)$$

# Today's Outline

# History

- Rosenblatt's *perceptron* (1958) represents ...
  - ... the first implemented neural network;
  - ... the first model for supervised learning.



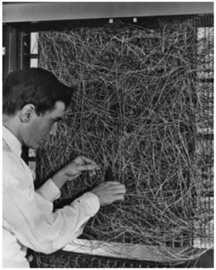


Figure: Rosenblatt

# Perceptron

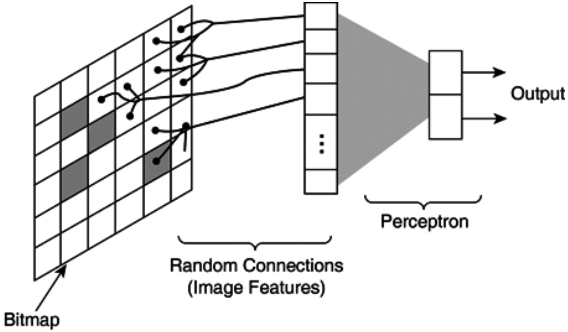



Figure: Rosenblatt's *Perceptron*

# General form

- two-class problems
- input space is transformed using fixed nonlinear functions $\phi(\mathbf{x})$
- uses labels $\{-1, 1\}$ for the two classes

$$y(\mathbf{x}) = f(\mathbf{w}^{\mathsf{T}}\phi(\mathbf{x}))$$

$$f(a) = \begin{cases} +1, & a \geq 0 \\ -1, & a < 0 \end{cases}$$

# The Perceptron Algorithm

- Goal: learn a vector **w** such that:
  - $\mathbf{w}^\mathsf{T}\phi(\mathbf{x}^{(n)}) > 0 \forall \mathbf{x}^{(n)} \in \mathcal{C}_1$
  - $\mathbf{w}^\mathsf{T}\phi(\mathbf{x}^{(n)}) < 0 \forall \mathbf{x}^{(n)} \in \mathcal{C}_2$
- The *perceptron criterion* minimizes:

$$E_p(\mathbf{w}) = -\sum_{\mathcal{M}} \mathbf{w}^\mathsf{T}\phi(\mathbf{x}^{(n)})t^{(n)}$$

# The Perceptron Algorithm

**Algorithm 1** Perceptron Learning Rule

**procedure** THE PERCEPTRON ALGORITHM($\mathbf{X}, \mathbf{T}, \eta$)
    $\mathbf{w} \leftarrow \mathbf{0}$
    **while** $E_p(\mathbf{X}, \mathbf{w}) > 0$ **do**
        choose $\mathbf{x}^{(n)} \in \mathbf{X}$
        compute $y(\mathbf{x}^{(n)}) = \text{sgn}(\mathbf{w}^{(t)\mathsf{T}}\mathbf{x}^{(n)})$
        **if** $t^{(n)} \neq y(\mathbf{x}^{(n)})$ **then**
            $\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t+1)} + \mathbf{x}^{(n)} t^{(n)}$
    **return** $\mathbf{w}$

# The Perceptron Convergence Theorem

**Definition**

If the data set is linearly separable, the perceptron algorithm always converges.

# Demo

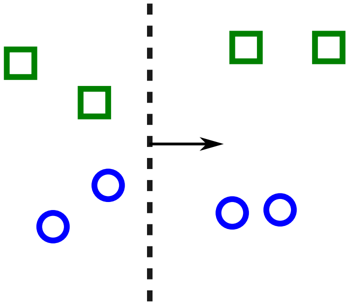


Figure: One update step for the perceptron

## Demo

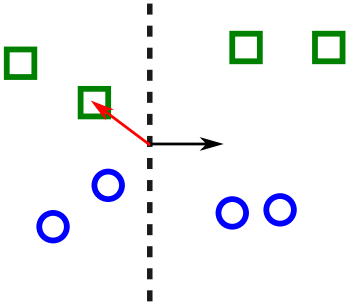


Figure: One update step for the perceptron

# Demo

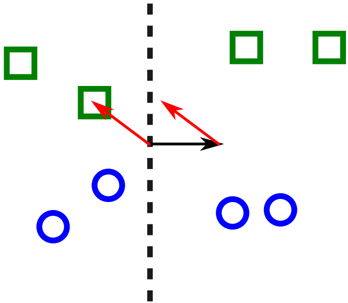


Figure: One update step for the perceptron

# Demo

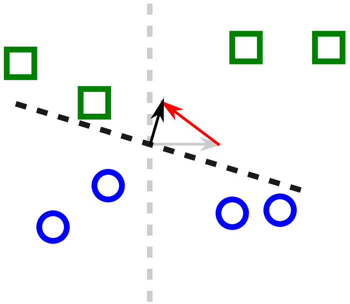


Figure: One update step for the perceptron

## Demo

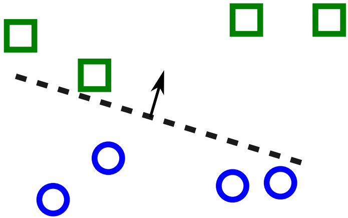


Figure: One update step for the perceptron

# Let's write some code...

- Demo: The Perceptron

# Summary

- One approach to classification is to build *discriminants*.
- The perceptron, Fisher's criterion and the LMS algorithm are all *linear discriminants* (they create *decision boundaries* that are *hyperplans*)
- The perceptron learning algorithm always converges if the data set is linearly separable.
- The perceptron learning algorithm never converges if the data set is not linearly separable.
- Minsky and Papert stated that the perceptron is incapable of generalization and its limitations also hold for the multi-layer case.
- We shall see if that is true in the following lectures.

# For the exam

For the exam you should be able to ...

- ... explain what *classification*, *decision boundaries*, and *linearly separable* data sets are;
- ... explain how least squares technique, Fisher's criterion and the perceptron work (description, not formulas);

Read ...

- *Discriminant Functions* [Bis06, Section 4.1]
- *Rosenblatt's Perceptron* [Hay09, Chapter 1]

# Today's Outline

# References I

Christopher M. Bishop, *Pattern recognition and machine learning (information science and statistics)*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

Simon S. Haykin, *Neural networks and learning machines*, Prentice Hall, 2009.