

# Machine Learning - Lecture 5

## Linear Discriminant Functions

28.04.2015

**Bastian Leibe**

**RWTH Aachen**

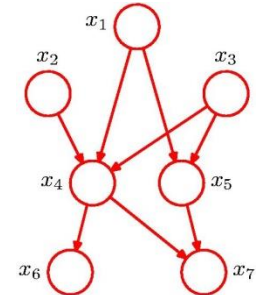
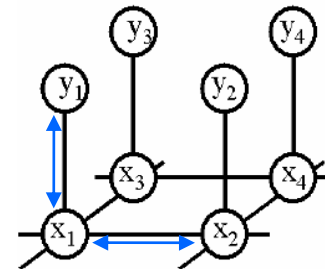
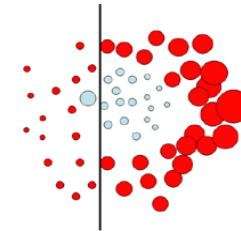
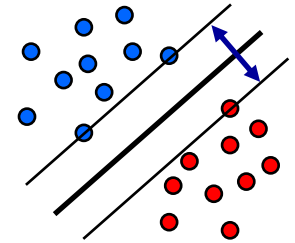
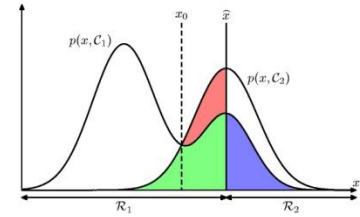
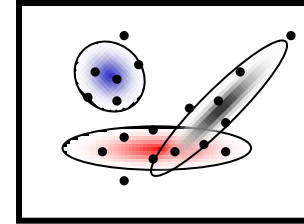
<http://www.vision.rwth-aachen.de>

[leibe@vision.rwth-aachen.de](mailto:leibe@vision.rwth-aachen.de)

Many slides adapted from B. Schiele

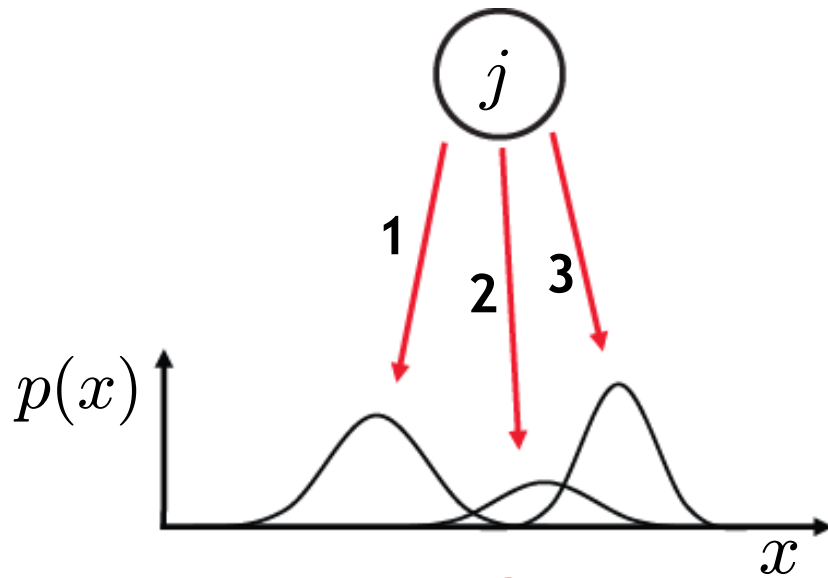
# Course Outline

- **Fundamentals (2 weeks)**
  - Bayes Decision Theory
  - Probability Density Estimation
- **Discriminative Approaches (5 weeks)**
  - **Linear Discriminant Functions**
  - Support Vector Machines
  - Ensemble Methods & Boosting
  - Randomized Trees, Forests & Ferns
- **Generative Models (4 weeks)**
  - Bayesian Networks
  - Markov Random Fields



# Recap: Mixture of Gaussians (MoG)

- “Generative model”

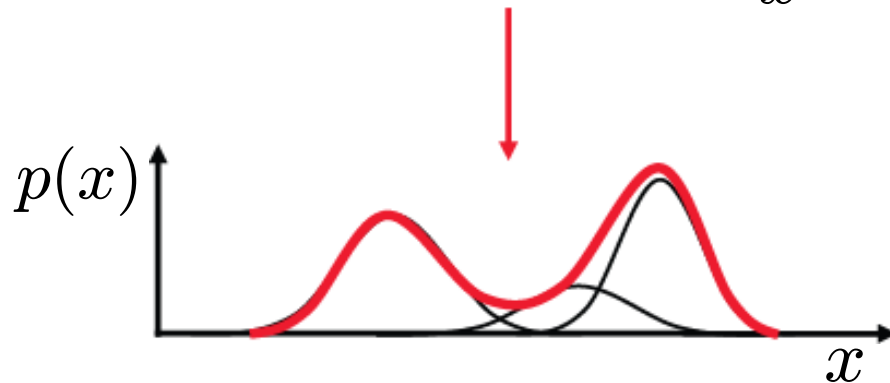


$$p(j) = \pi_j$$

“Weight” of mixture component

$$p(x|\theta_j)$$

Mixture component

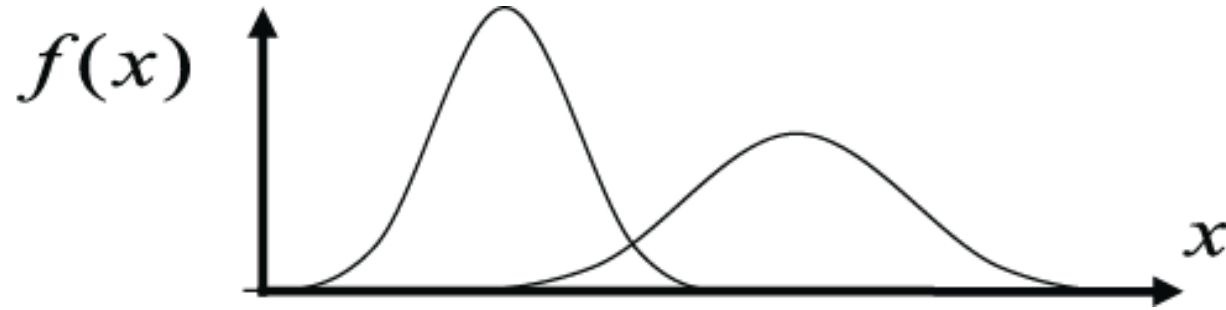


$$p(x|\theta) = \sum_{j=1}^M p(x|\theta_j)p(j)$$

Mixture density

# Recap: Estimating MoGs - Iterative Strategy

- Assuming we knew the values of the hidden variable...



ML for Gaussian #1

ML for Gaussian #2

assumed known  $\longrightarrow$  1 111                      22 2 2                       $j$

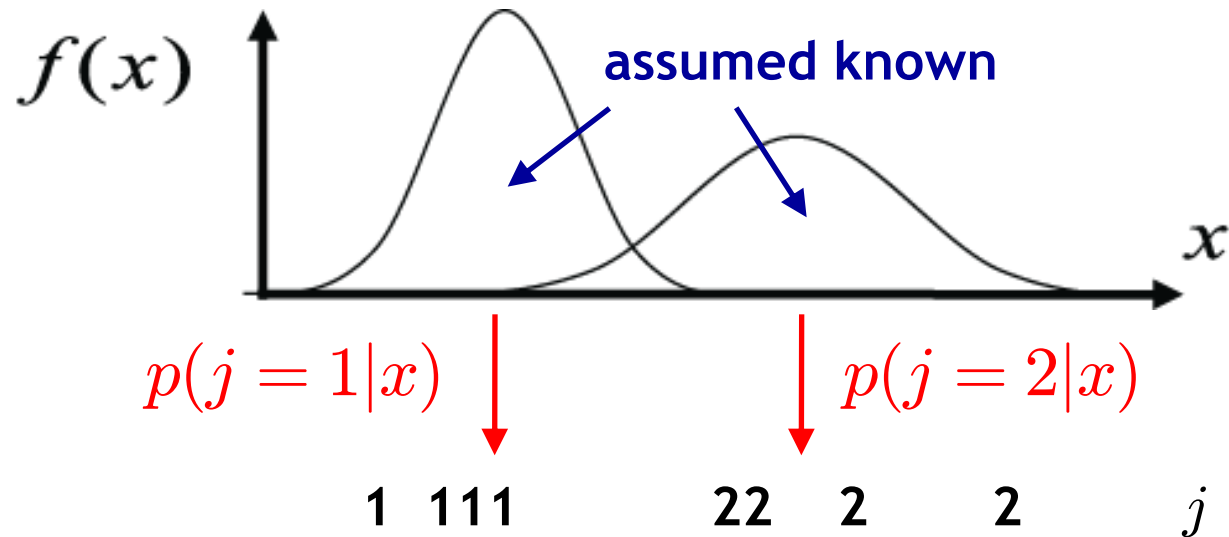
$$h(j = 1|x_n) = \begin{matrix} 1 & 111 \\ 00 & 0 & 0 \end{matrix}$$

$$h(j = 2|x_n) = \begin{matrix} 0 & 000 \\ 11 & 1 & 1 \end{matrix}$$

$$\mu_1 = \frac{\sum_{n=1}^N h(j = 1|x_n)x_n}{\sum_{i=1}^N h(j = 1|x_n)} \qquad \mu_2 = \frac{\sum_{n=1}^N h(j = 2|x_n)x_n}{\sum_{i=1}^N h(j = 2|x_n)}$$

# Recap: Estimating MoGs - Iterative Strategy

- Assuming we knew the mixture components...

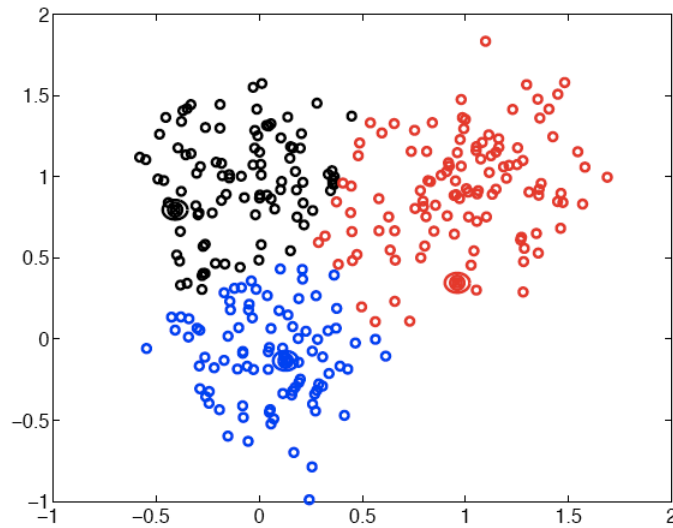
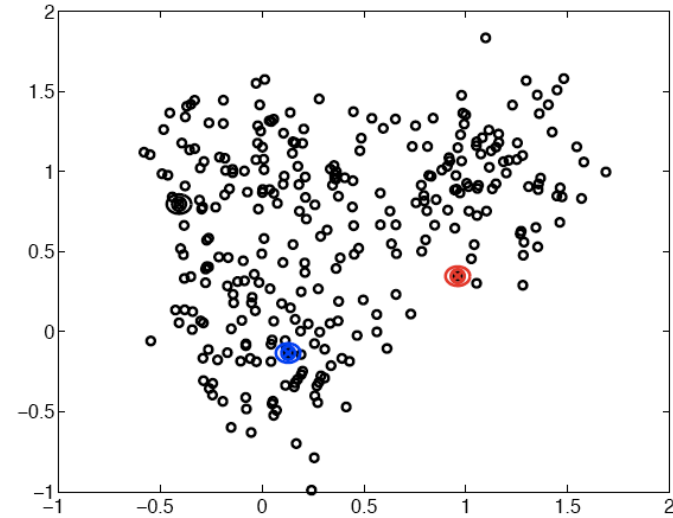


- Bayes decision rule: Decide  $j = 1$  if

$$p(j = 1|x_n) > p(j = 2|x_n)$$

# Recap: K-Means Clustering

- Iterative procedure
  1. Initialization: pick  $K$  arbitrary centroids (cluster means)
  2. Assign each sample to the closest centroid.
  3. Adjust the centroids to be the means of the samples assigned to them.
  4. Go to step 2 (until no change)
- Algorithm is guaranteed to converge after finite #iterations.
  - Local optimum
  - Final result depends on initialization.



# Recap: EM Algorithm

- **Expectation-Maximization (EM) Algorithm**

- **E-Step:** softly assign samples to mixture components

$$\gamma_j(\mathbf{x}_n) \leftarrow \frac{\pi_j \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}{\sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)} \quad \forall j = 1, \dots, K, \quad n = 1, \dots, N$$

- **M-Step:** re-estimate the parameters (separately for each mixture component) based on the soft assignments

$$\hat{N}_j \leftarrow \sum_{n=1}^N \gamma_j(\mathbf{x}_n) = \text{soft number of samples labeled } j$$

$$\hat{\pi}_j^{\text{new}} \leftarrow \frac{\hat{N}_j}{N}$$

$$\hat{\boldsymbol{\mu}}_j^{\text{new}} \leftarrow \frac{1}{\hat{N}_j} \sum_{n=1}^N \gamma_j(\mathbf{x}_n) \mathbf{x}_n$$

$$\hat{\boldsymbol{\Sigma}}_j^{\text{new}} \leftarrow \frac{1}{\hat{N}_j} \sum_{n=1}^N \gamma_j(\mathbf{x}_n) (\mathbf{x}_n - \hat{\boldsymbol{\mu}}_j^{\text{new}})(\mathbf{x}_n - \hat{\boldsymbol{\mu}}_j^{\text{new}})^{\text{T}}$$

# Topics of This Lecture

- **Linear discriminant functions**
  - Definition
  - Extension to multiple classes
- **Least-squares classification**
  - Derivation
  - Shortcomings
- **Generalized linear models**
  - Connection to neural networks
  - Generalized linear discriminants & gradient descent



# Discriminant Functions

- **Bayesian Decision Theory** 
$$p(\mathcal{C}_k|x) = \frac{p(x|\mathcal{C}_k)p(\mathcal{C}_k)}{p(x)}$$
  - Model conditional probability densities  $p(x|\mathcal{C}_k)$  and priors  $p(\mathcal{C}_k)$
  - Compute posteriors  $p(\mathcal{C}_k|x)$  (using Bayes' rule)
  - Minimize probability of misclassification by maximizing  $p(\mathcal{C}|x)$ .
- **New approach**
  - Directly encode decision boundary
  - Without explicit modeling of probability densities
  - Minimize misclassification probability directly.

# Recap: Discriminant Functions

- Formulate classification in terms of comparisons

- Discriminant functions

$$y_1(x), \dots, y_K(x)$$

- Classify  $x$  as class  $C_k$  if

$$y_k(x) > y_j(x) \quad \forall j \neq k$$

- Examples (Bayes Decision Theory)

$$y_k(x) = p(C_k|x)$$

$$y_k(x) = p(x|C_k)p(C_k)$$

$$y_k(x) = \log p(x|C_k) + \log p(C_k)$$

# Discriminant Functions

- **Example: 2 classes**

$$y_1(x) > y_2(x)$$

$$\Leftrightarrow y_1(x) - y_2(x) > 0$$

$$\Leftrightarrow \mathbf{y}(x) > 0$$

- **Decision functions (from Bayes Decision Theory)**

$$y(x) = p(\mathcal{C}_1|x) - p(\mathcal{C}_2|x)$$

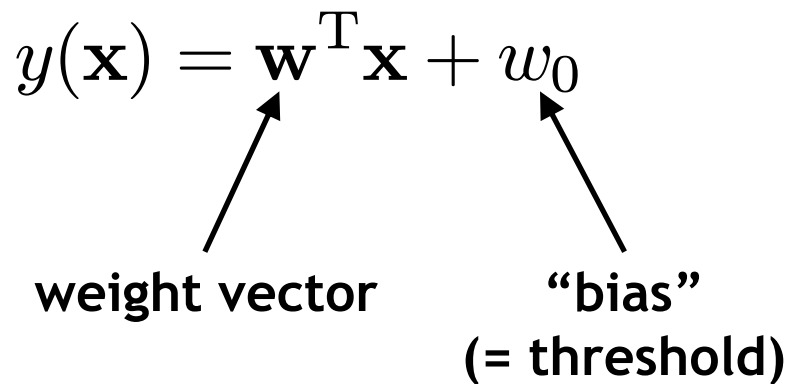
$$y(x) = \ln \frac{p(x|\mathcal{C}_1)}{p(x|\mathcal{C}_2)} + \ln \frac{p(\mathcal{C}_1)}{p(\mathcal{C}_2)}$$

# Learning Discriminant Functions

- **General classification problem**
  - Goal: take a new input  $\mathbf{x}$  and assign it to one of  $K$  classes  $C_k$ .
  - Given: training set  $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$   
with target values  $\mathbf{T} = \{\mathbf{t}_1, \dots, \mathbf{t}_N\}$ .
  - ⇒ Learn a discriminant function  $y(\mathbf{x})$  to perform the classification.
- **2-class problem**
  - Binary target values:  $t_n \in \{0, 1\}$
- **K-class problem**
  - 1-of-K coding scheme, e.g.  $\mathbf{t}_n = (0, 1, 0, 0, 0)^T$

# Linear Discriminant Functions

- 2-class problem
  - $y(x) > 0$  : Decide for class  $C_1$ , else for class  $C_2$
- In the following, we focus on **linear discriminant functions**

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


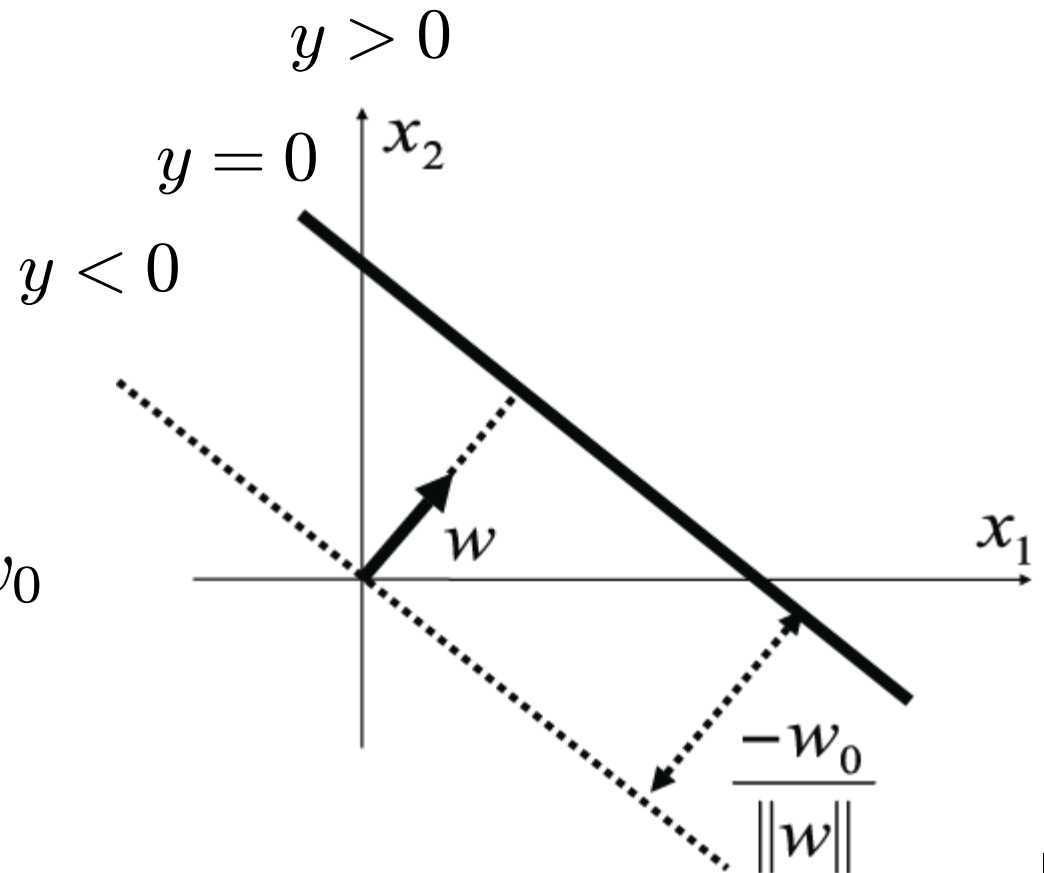
weight vector      “bias”  
(= threshold)

- If a data set can be perfectly classified by a linear discriminant, then we call it **linearly separable**.

# Linear Discriminant Functions

- Decision boundary  $y(\mathbf{x}) = 0$  defines a hyperplane
  - Normal vector:  $\mathbf{w}$
  - Offset:  $\frac{-w_0}{\|\mathbf{w}\|}$

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



# Linear Discriminant Functions

- **Notation**

- $D$  : Number of dimensions

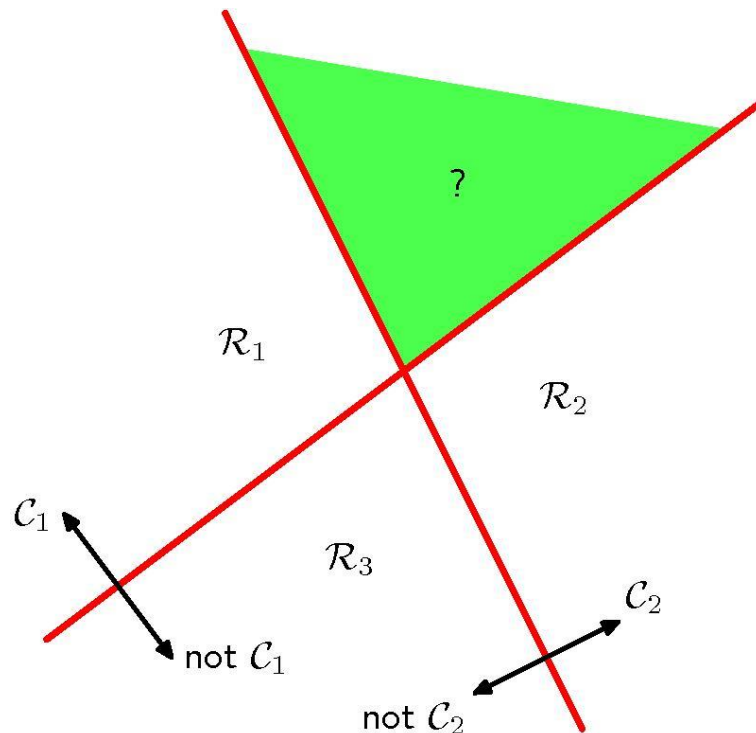
$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \end{bmatrix} \quad \mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_D \end{bmatrix}$$

$$\begin{aligned} y(\mathbf{x}) &= \mathbf{w}^T \mathbf{x} + w_0 \\ &= \sum_{i=1}^D w_i x_i + w_0 \\ &= \sum_{i=0}^D w_i x_i \quad \text{with } x_0 = 1 \text{ constant} \end{aligned}$$

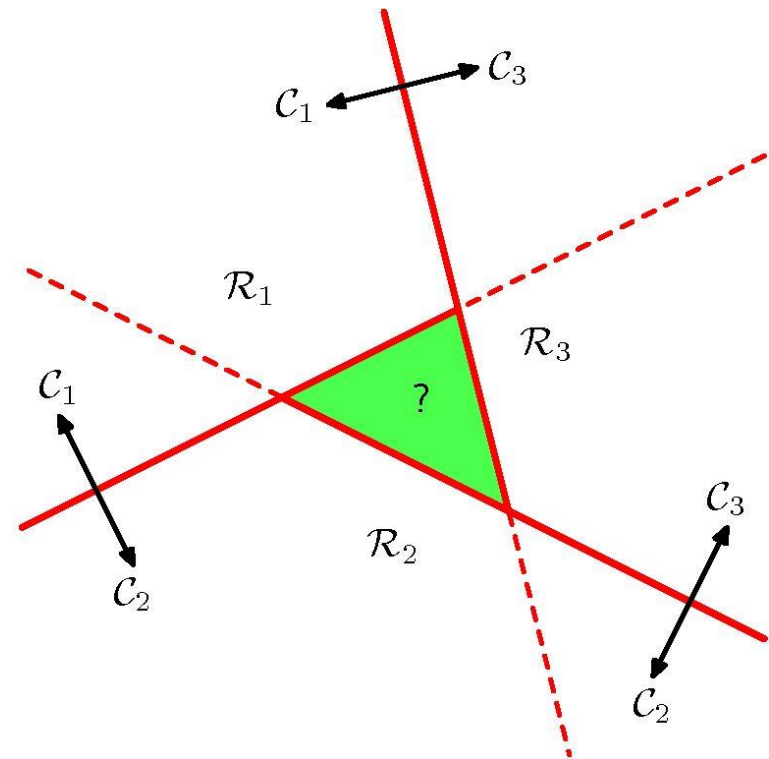
# Extension to Multiple Classes

- Two simple strategies

*One-vs-all* classifiers



*One-vs-one* classifiers



- How many classifiers do we need in both cases?
- What difficulties do you see for those strategies?



# Extension to Multiple Classes

## • Problem

- Both strategies result in regions for which the pure classification result ( $y_k > 0$ ) is ambiguous.
- In the *one-vs-all* case, it is still possible to classify those inputs based on the continuous classifier outputs  $y_k > y_j \forall j \neq k$ .

## • Solution

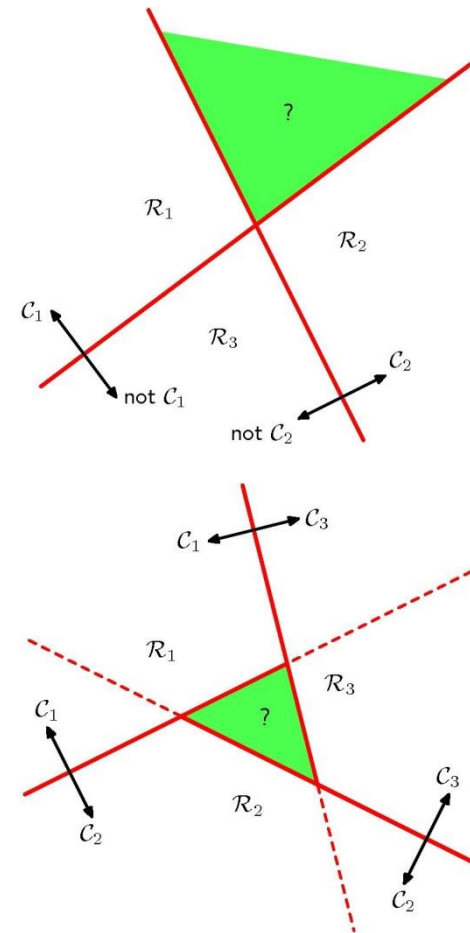
- We can avoid those difficulties by taking  $K$  linear functions of the form

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

and defining the decision boundaries directly by deciding for  $C_k$  iff  $y_k > y_j \forall j \neq k$ .

- This corresponds to a 1-of-K coding scheme

$$\mathbf{t}_n = (0, 1, 0, \dots, 0, 0)^T$$



# Extension to Multiple Classes

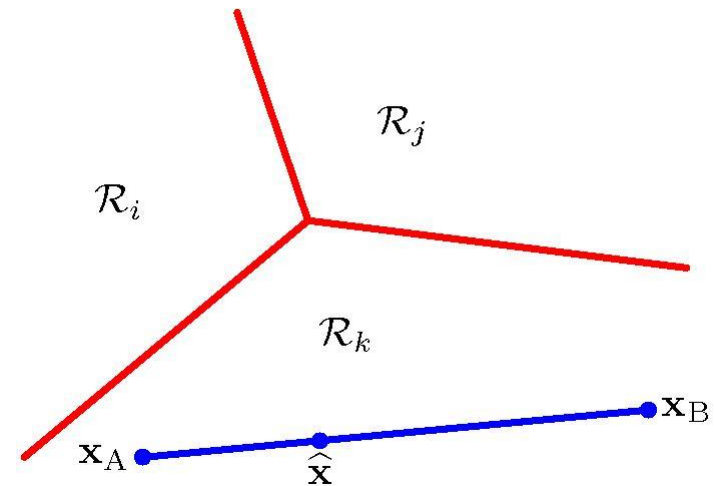
- K-class discriminant

- Combination of K linear functions

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

- Resulting decision hyperplanes:

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



- It can be shown that the decision regions of such a discriminant are always singly connected and convex.
- This makes linear discriminant models particularly suitable for problems for which the conditional densities  $p(\mathbf{x} | w_i)$  are unimodal.

# Topics of This Lecture

- Linear discriminant functions
  - Definition
  - Extension to multiple classes
- **Least-squares classification**
  - **Derivation**
  - **Shortcomings**
- Generalized linear models
  - Connection to neural networks
  - Generalized linear discriminants & gradient descent

# General Classification Problem

- **Classification problem**

- Let's consider  $K$  classes described by linear models

$$y_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{x} + w_{k0}, \quad k = 1, \dots, K$$

- We can group those together using vector notation

$$\mathbf{y}(\mathbf{x}) = \widetilde{\mathbf{W}}^T \widetilde{\mathbf{x}}$$

where

$$\widetilde{\mathbf{W}} = [\widetilde{\mathbf{w}}_1, \dots, \widetilde{\mathbf{w}}_K] = \begin{bmatrix} w_{10} & \dots & w_{K0} \\ w_{11} & \dots & w_{K1} \\ \vdots & \ddots & \vdots \\ w_{1D} & \dots & w_{KD} \end{bmatrix}$$

- The output will again be in 1-of-K notation.

⇒ We can directly compare it to the target value  $\mathbf{t} = [t_1, \dots, t_k]^T$ .

# General Classification Problem

- Classification problem
  - For the entire dataset, we can write

$$\mathbf{Y}(\tilde{\mathbf{X}}) = \tilde{\mathbf{X}}\tilde{\mathbf{W}}$$

and compare this to the target matrix  $\mathbf{T}$  where

$$\tilde{\mathbf{W}} = [\tilde{\mathbf{w}}_1, \dots, \tilde{\mathbf{w}}_K]$$
$$\tilde{\mathbf{X}} = \begin{bmatrix} \mathbf{x}_1^T \\ \vdots \\ \mathbf{x}_N^T \end{bmatrix} \quad \mathbf{T} = \begin{bmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_N^T \end{bmatrix}$$

- Result of the comparison:

$$\tilde{\mathbf{X}}\tilde{\mathbf{W}} - \mathbf{T}$$

**Goal: Choose  $\tilde{\mathbf{W}}$  such that this is minimal!**

# Least-Squares Classification

- Simplest approach
  - Directly try to minimize the **sum-of-squares error**
  - We could write this as

$$\begin{aligned} E(\mathbf{w}) &= \frac{1}{2} \sum_{n=1}^N \sum_{k=1}^K (y_k(\mathbf{x}_n; \mathbf{w}) - t_{kn})^2 \\ &= \frac{1}{2} \sum_{n=1}^N \sum_{k=1}^K (\mathbf{w}_k^T \mathbf{x}_n - t_{kn})^2 \end{aligned}$$

- But let's stick with the matrix notation for now...
- (The result will be simpler to express and we'll learn some nice matrix algebra rules along the way...)

# Least-Squares Classification

- Multi-class case

- Let's formulate the **sum-of-squares error** in matrix notation

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

- Taking the derivative yields

$$\begin{aligned} \frac{\partial}{\partial \tilde{\mathbf{W}}} E_D(\tilde{\mathbf{W}}) &= \frac{1}{2} \frac{\partial}{\partial \tilde{\mathbf{W}}} \text{Tr} \left\{ (\tilde{\mathbf{X}}\tilde{\mathbf{W}} - \mathbf{T})^T (\tilde{\mathbf{X}}\tilde{\mathbf{W}} - \mathbf{T}) \right\} \\ &= \frac{1}{2} \frac{\partial}{\partial (\tilde{\mathbf{X}}\tilde{\mathbf{W}} - \mathbf{T})^T (\tilde{\mathbf{X}}\tilde{\mathbf{W}} - \mathbf{T})} \text{Tr} \left\{ (\tilde{\mathbf{X}}\tilde{\mathbf{W}} - \mathbf{T})^T (\tilde{\mathbf{X}}\tilde{\mathbf{W}} - \mathbf{T}) \right\} \\ &\quad \cdot \frac{\partial}{\partial \tilde{\mathbf{W}}} (\tilde{\mathbf{X}}\tilde{\mathbf{W}} - \mathbf{T})^T (\tilde{\mathbf{X}}\tilde{\mathbf{W}} - \mathbf{T}) \\ &= \tilde{\mathbf{X}}^T (\tilde{\mathbf{X}}\tilde{\mathbf{W}} - \mathbf{T}) \end{aligned}$$

using:

$$\sum_{i,j} a_{ij}^2 = \text{Tr} \{ \mathbf{A}^T \mathbf{A} \}$$

chain rule:

$$\frac{\partial \mathbf{Z}}{\partial \mathbf{X}} = \frac{\partial \mathbf{Z}}{\partial \mathbf{Y}} \frac{\partial \mathbf{Y}}{\partial \mathbf{X}}$$

using:

$$\frac{\partial}{\partial \mathbf{A}} \text{Tr} \{ \mathbf{A} \} = \mathbf{I}$$

# Least-Squares Classification

- Minimizing the sum-of-squares error

$$\frac{\partial}{\partial \widetilde{\mathbf{W}}} E_D(\widetilde{\mathbf{W}}) = \widetilde{\mathbf{X}}^T (\widetilde{\mathbf{X}} \widetilde{\mathbf{W}} - \mathbf{T}) \stackrel{!}{=} 0$$

$$\widetilde{\mathbf{X}} \widetilde{\mathbf{W}} = \mathbf{T}$$

$$\widetilde{\mathbf{W}} = (\widetilde{\mathbf{X}}^T \widetilde{\mathbf{X}})^{-1} \widetilde{\mathbf{X}}^T \mathbf{T}$$

$$= \widetilde{\mathbf{X}}^\dagger \mathbf{T} \quad \text{“pseudo-inverse”}$$

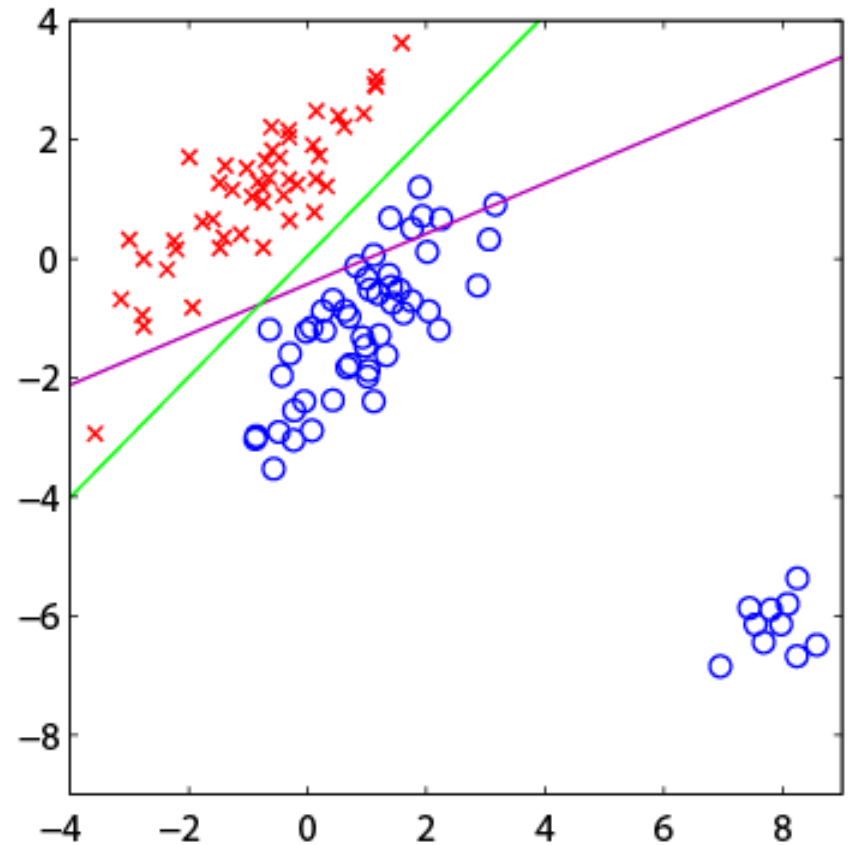
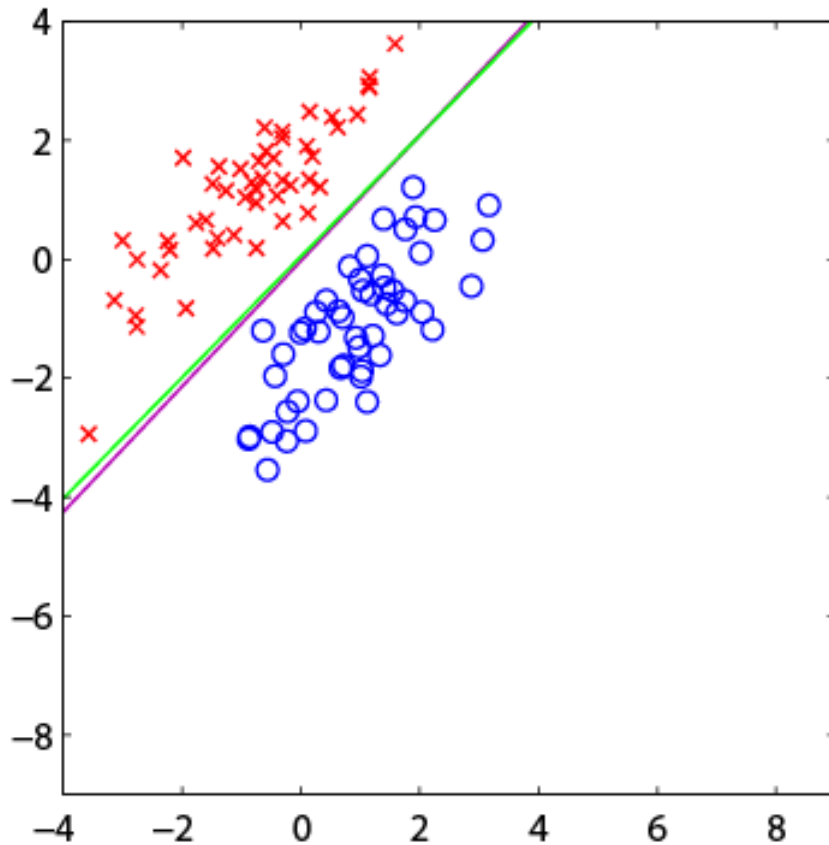
- We then obtain the discriminant function as

$$\mathbf{y}(\mathbf{x}) = \widetilde{\mathbf{W}}^T \widetilde{\mathbf{x}} = \mathbf{T}^T (\widetilde{\mathbf{X}}^\dagger)^T \widetilde{\mathbf{x}}$$

- ⇒ Exact, closed-form solution for the discriminant function parameters.



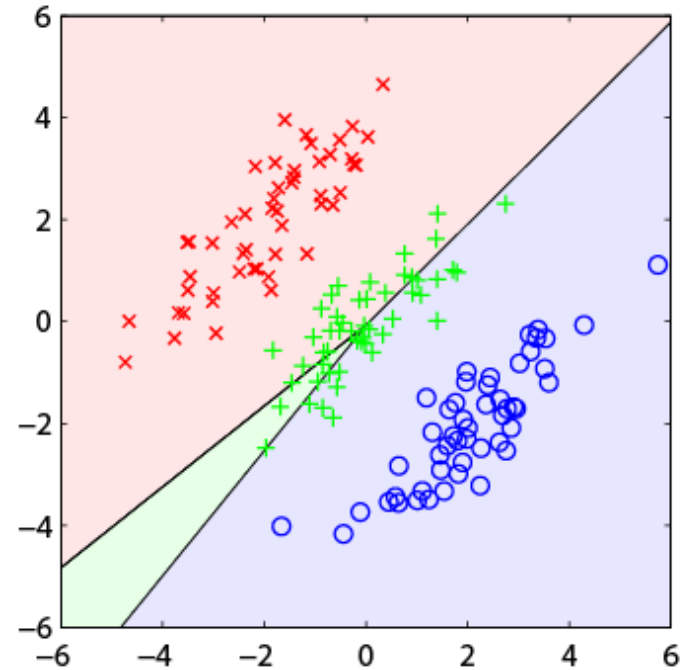
# Problems with Least Squares



- **Least-squares is very sensitive to outliers!**
  - The error function penalizes predictions that are “too correct”.

# Problems with Least-Squares

- Another example:
    - 3 classes (red, green, blue)
    - Linearly separable problem
    - Least-squares solution:  
Most green points are misclassified!
  - Deeper reason for the failure
    - Least-squares corresponds to Maximum Likelihood under the assumption of a Gaussian conditional distribution.
    - However, our binary target vectors have a distribution that is clearly non-Gaussian!
- ⇒ Least-squares is the wrong probabilistic tool in this case!



# Topics of This Lecture

- Linear discriminant functions
  - Definition
  - Extension to multiple classes
- Least-squares classification
  - Derivation
  - Shortcomings
- **Generalized linear models**
  - **Connection to neural networks**
  - **Generalized linear discriminants & gradient descent**

# Generalized Linear Models

- Linear model

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

- Generalized linear model

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

- $g(\cdot)$  is called an **activation function** and may be nonlinear.
- The decision surfaces correspond to
$$y(\mathbf{x}) = \text{const.} \quad \Leftrightarrow \quad \mathbf{w}^T \mathbf{x} + w_0 = \text{const.}$$
- If  $g$  is monotonous (which is typically the case), the resulting decision boundaries are still linear functions of  $\mathbf{x}$ .

# Generalized Linear Models

- Consider 2 classes:

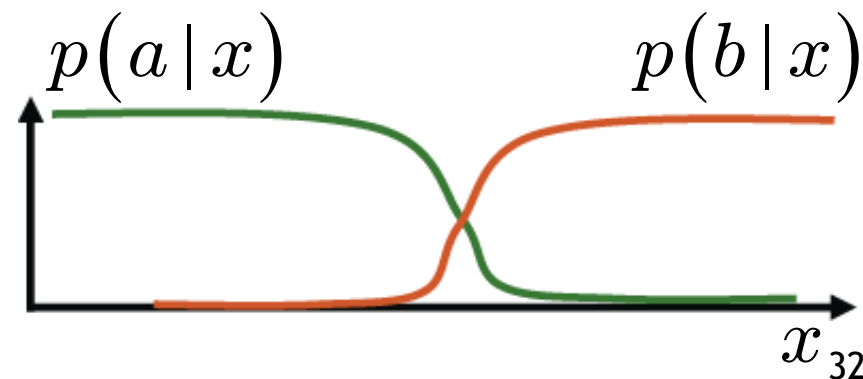
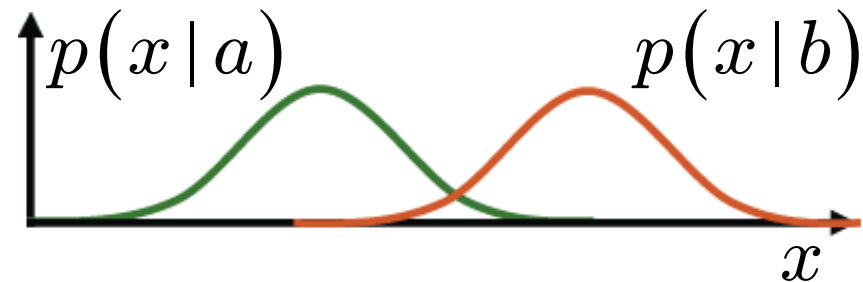
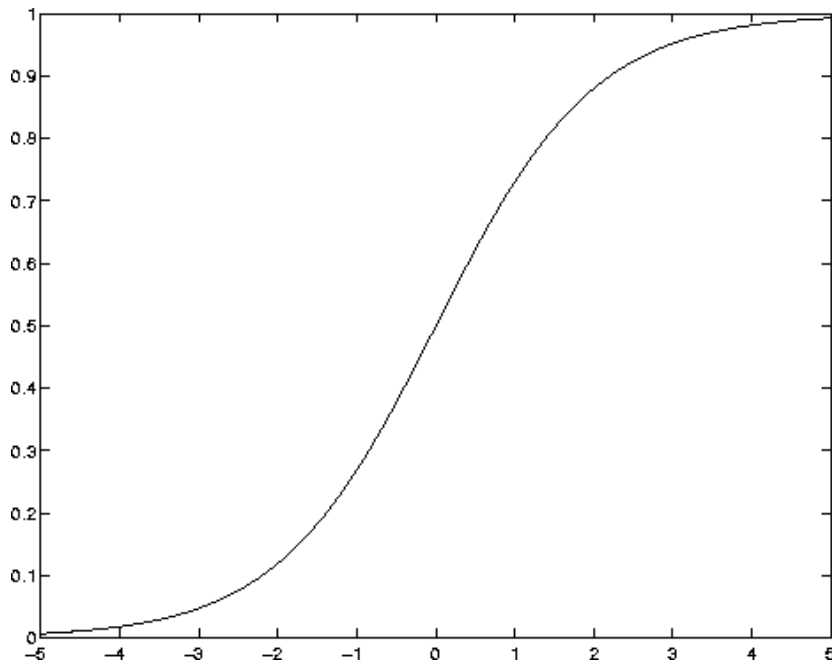
$$\begin{aligned} p(\mathcal{C}_1|\mathbf{x}) &= \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \\ &= \frac{1}{1 + \frac{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}} \\ &= \frac{1}{1 + \exp(-a)} \equiv g(a) \end{aligned}$$

$$\text{with } a = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}$$

# Logistic Sigmoid Activation Function

$$g(a) \equiv \frac{1}{1 + \exp(-a)}$$

Example: Normal distributions with identical covariance



# Normalized Exponential

- General case of  $K > 2$  classes:

$$\begin{aligned} p(\mathcal{C}_k | \mathbf{x}) &= \frac{p(\mathbf{x} | \mathcal{C}_k) p(\mathcal{C}_k)}{\sum_j p(\mathbf{x} | \mathcal{C}_j) p(\mathcal{C}_j)} \\ &= \frac{\exp(a_k)}{\sum_j \exp(a_j)} \end{aligned}$$

$$\text{with } a_k = \ln p(\mathbf{x} | \mathcal{C}_k) p(\mathcal{C}_k)$$

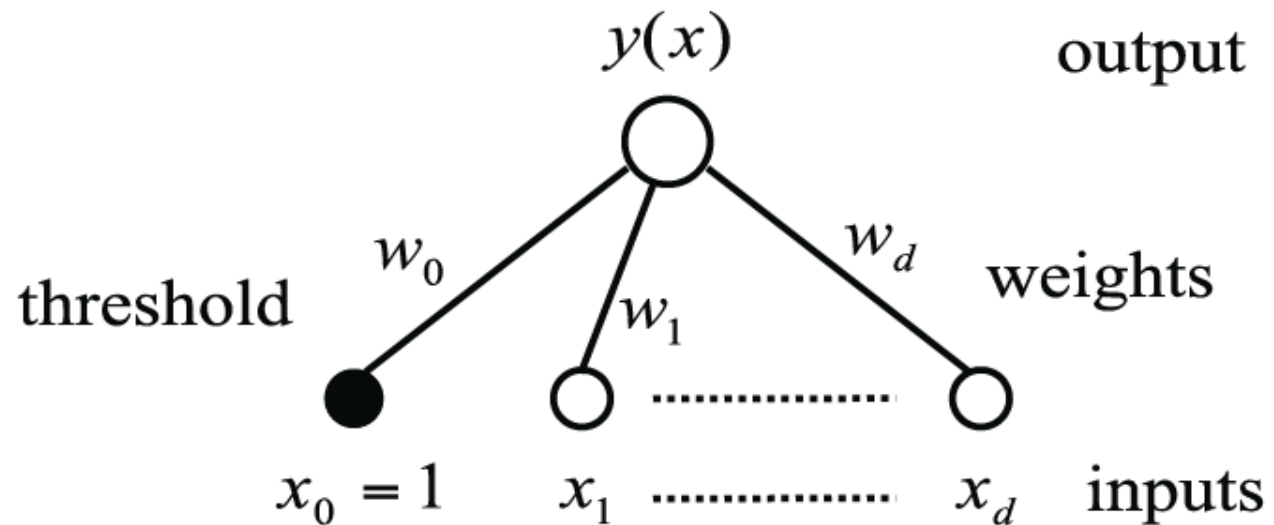
- This is known as the **normalized exponential** or **softmax function**
- Can be regarded as a multiclass generalization of the logistic sigmoid.

# Relationship to Neural Networks

- 2-Class case

$$y(\mathbf{x}) = g \left( \sum_{i=0}^D w_i x_i \right) \text{ with } x_0 = 1 \text{ constant}$$

- Neural network (“single-layer perceptron”)



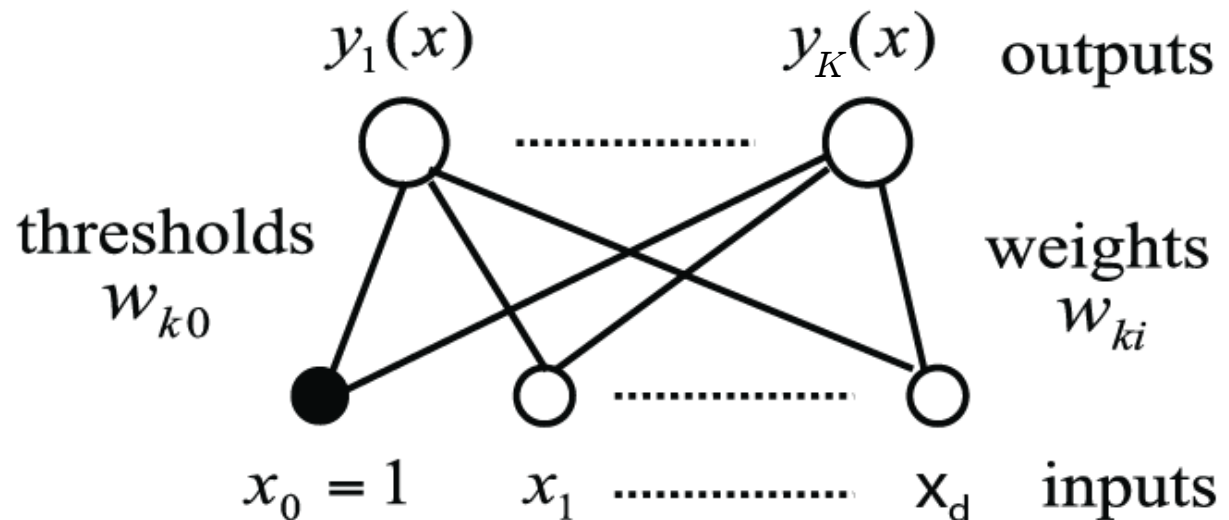


# Relationship to Neural Networks

- Multi-class case

$$y_k(\mathbf{x}) = g \left( \sum_{i=0}^D w_{ki} x_i \right) \text{ with } x_0 = 1 \text{ constant}$$

- Multi-class perceptron

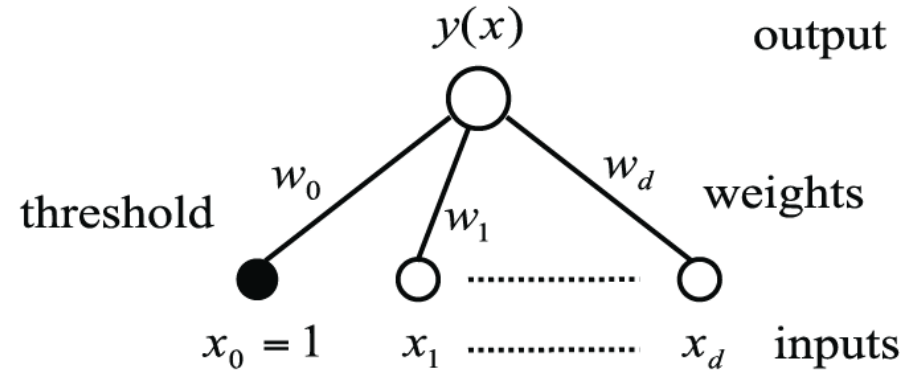


# Logistic Discrimination

- If we use the logistic sigmoid activation function...

$$g(a) \equiv \frac{1}{1 + \exp(-a)}$$

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



... then we can interpret the  $y(x)$  as posterior probabilities!

# Other Motivation for Nonlinearity

- Recall least-squares classification

- One of the problems was that data points that are “too correct” have a strong influence on the decision surface under a squared-error criterion.

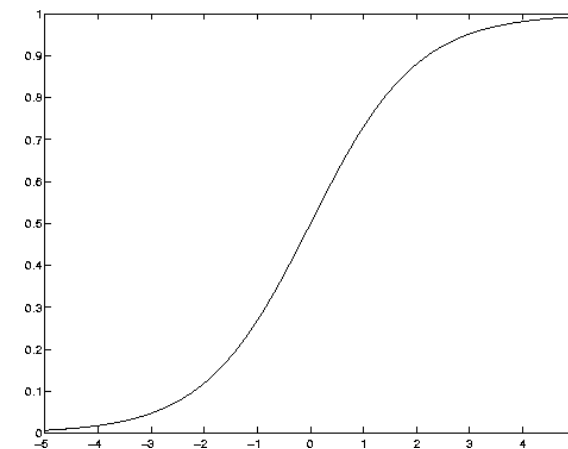
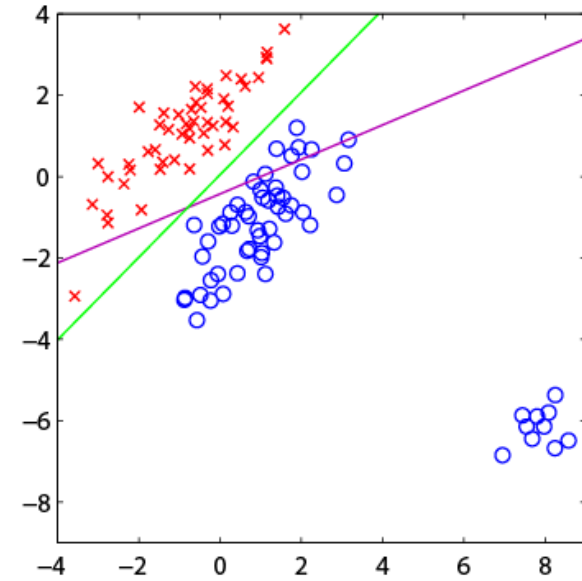
$$E(\mathbf{w}) = \sum_{n=1}^N (y(\mathbf{x}_n; \mathbf{w}) - t_n)^2$$

- Reason: the output of  $y(\mathbf{x}_n; \mathbf{w})$  can grow arbitrarily large for some  $\mathbf{x}_n$ :

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

- By choosing a suitable nonlinearity (e.g. a sigmoid), we can limit those influences

$$y(\mathbf{x}; \mathbf{w}) = g(\mathbf{w}^T \mathbf{x} + w_0)$$



# Discussion: Generalized Linear Models

- **Advantages**

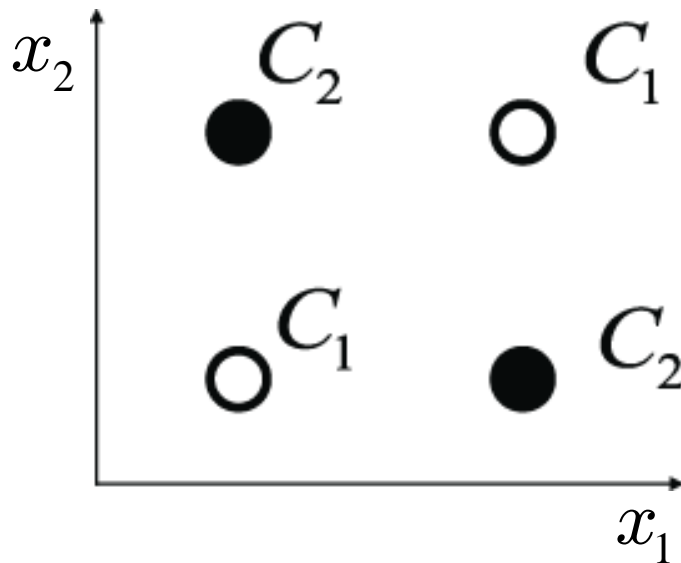
- The nonlinearity gives us more flexibility.
- Can be used to limit the effect of outliers.
- Choice of a sigmoid leads to a nice probabilistic interpretation.

- **Disadvantage**

- Least-squares minimization in general no longer leads to a closed-form analytical solution.
  - ⇒ Need to apply iterative methods.
  - ⇒ Gradient descent.

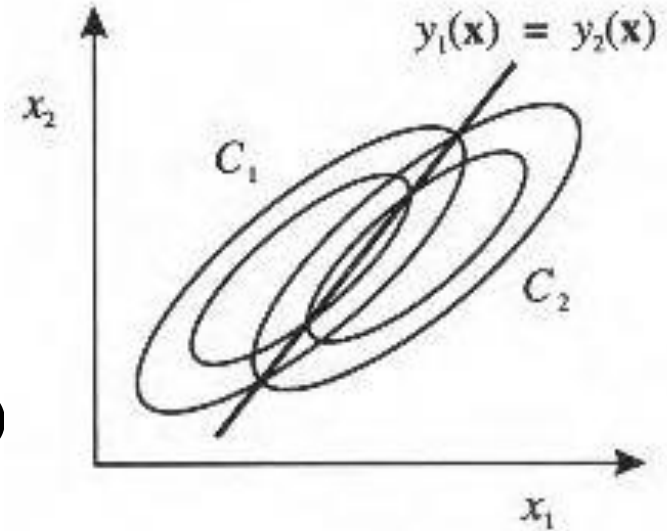
# Linear Separability

- Up to now: restrictive assumption
  - Only consider linear decision boundaries
  
- Classical counterexample: XOR



# Linear Separability

- Even if the data is not linearly separable, a linear decision boundary may still be “optimal”.
  - Generalization
  - E.g. in the case of Normal distributed data (with equal covariance matrices)



- Choice of the right discriminant function is important and should be based on
  - Prior knowledge (of the general functional form)
  - Empirical comparison of alternative models
  - Linear discriminants are often used as benchmark.

# Generalized Linear Discriminants

- **Generalization**

- Transform vector  $\mathbf{x}$  with  $M$  nonlinear basis functions  $\phi_j(\mathbf{x})$ :

$$y_k(\mathbf{x}) = \sum_{j=1}^M w_{kj} \phi_j(\mathbf{x}) + w_{k0}$$

- Purpose of  $\phi_j(\mathbf{x})$ : basis functions
- Allow non-linear decision boundaries.
- By choosing the right  $\phi_j$ , every continuous function can (in principle) be approximated with arbitrary accuracy.

- **Notation**

$$y_k(\mathbf{x}) = \sum_{j=0}^M w_{kj} \phi_j(\mathbf{x}) \quad \text{with } \phi_0(\mathbf{x}) = 1$$

# Generalized Linear Discriminants

- **Model**

$$y_k(\mathbf{x}) = \sum_{j=0}^M w_{kj} \phi_j(\mathbf{x}) = y_k(\mathbf{x}; \mathbf{w})$$

- $K$  functions (outputs)  $y_k(\mathbf{x}; \mathbf{w})$

- **Learning in Neural Networks**

- **Single-layer networks:**  $\phi_j$  are fixed, only weights  $\mathbf{w}$  are learned.
- **Multi-layer networks:** both the  $\mathbf{w}$  and the  $\phi_j$  are learned.
- In the following, we will not go into details about neural networks in particular, but consider generalized linear discriminants in general...



# Gradient Descent

- Learning the weights  $\mathbf{w}$ :

- $N$  training data points:  $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$
- $K$  outputs of decision functions:  $y_k(\mathbf{x}_n; \mathbf{w})$
- Target vector for each data point:  $\mathbf{T} = \{\mathbf{t}_1, \dots, \mathbf{t}_N\}$

- Error function (least-squares error) of linear model

$$\begin{aligned} E(\mathbf{w}) &= \frac{1}{2} \sum_{n=1}^N \sum_{k=1}^K (y_k(\mathbf{x}_n; \mathbf{w}) - t_{kn})^2 \\ &= \frac{1}{2} \sum_{n=1}^N \sum_{k=1}^K \left( \sum_{j=1}^M w_{kj} \phi_j(\mathbf{x}_n) - t_{kn} \right)^2 \end{aligned}$$

# Gradient Descent

- **Problem**

- The error function can in general no longer be minimized in closed form.

- **Idea (Gradient Descent)**

- Iterative minimization
- Start with an initial guess for the parameter values  $w_{kj}^{(0)}$ .
- Move towards a (local) minimum by following the gradient.

$$w_{kj}^{(\tau+1)} = w_{kj}^{(\tau)} - \eta \left. \frac{\partial E(\mathbf{w})}{\partial w_{kj}} \right|_{\mathbf{w}^{(\tau)}}$$

$\eta$ : Learning rate

- This simple scheme corresponds to a 1<sup>st</sup>-order Taylor expansion (There are more complex procedures available).

# Gradient Descent - Basic Strategies

- “Batch learning”

$$w_{kj}^{(\tau+1)} = w_{kj}^{(\tau)} - \eta \left. \frac{\partial E(\mathbf{w})}{\partial w_{kj}} \right|_{\mathbf{w}^{(\tau)}}$$

$\eta$ : Learning rate

- Compute the gradient based on all training data:

$$\frac{\partial E(\mathbf{w})}{\partial w_{kj}}$$

# Gradient Descent - Basic Strategies

- “Sequential updating”

$$E(\mathbf{w}) = \sum_{n=1}^N E_n(\mathbf{w})$$

$$w_{kj}^{(\tau+1)} = w_{kj}^{(\tau)} - \eta \left. \frac{\partial E_n(\mathbf{w})}{\partial w_{kj}} \right|_{\mathbf{w}^{(\tau)}}$$

$\eta$ : Learning rate

- Compute the gradient based on a single data point at a time:

$$\frac{\partial E_n(\mathbf{w})}{\partial w_{kj}}$$

# Gradient Descent

- Error function

$$E(\mathbf{w}) = \sum_{n=1}^N E_n(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \sum_{k=1}^K \left( \sum_{j=1}^M w_{kj} \phi_j(\mathbf{x}_n) - t_{kn} \right)^2$$

$$E_n(\mathbf{w}) = \frac{1}{2} \sum_{k=1}^K \left( \sum_{j=1}^M w_{kj} \phi_j(\mathbf{x}_n) - t_{kn} \right)^2$$

$$\frac{\partial E_n(\mathbf{w})}{\partial w_{kj}} = \left( \sum_{\tilde{j}=1}^M w_{k\tilde{j}} \phi_{\tilde{j}}(\mathbf{x}_n) - t_{kn} \right) \phi_j(\mathbf{x}_n)$$

$$= (y_k(\mathbf{x}_n; \mathbf{w}) - t_{kn}) \phi_j(\mathbf{x}_n)$$

# Gradient Descent

- Delta rule (=LMS rule)

$$\begin{aligned}w_{kj}^{(\tau+1)} &= w_{kj}^{(\tau)} - \eta (y_k(\mathbf{x}_n; \mathbf{w}) - t_{kn}) \phi_j(\mathbf{x}_n) \\ &= w_{kj}^{(\tau)} - \eta \delta_{kn} \phi_j(\mathbf{x}_n)\end{aligned}$$

- ▶ where

$$\delta_{kn} = y_k(\mathbf{x}_n; \mathbf{w}) - t_{kn}$$

⇒ Simply feed back the input data point, weighted by the classification error.

# Gradient Descent

- Cases with differentiable, non-linear activation function

$$y_k(\mathbf{x}) = g(a_k) = g \left( \sum_{j=0}^M w_{kj} \phi_j(\mathbf{x}_n) \right)$$

- Gradient descent

$$\frac{\partial E_n(\mathbf{w})}{\partial w_{kj}} = \frac{\partial g(a_k)}{\partial w_{kj}} (y_k(\mathbf{x}_n; \mathbf{w}) - t_{kn}) \phi_j(\mathbf{x}_n)$$

$$w_{kj}^{(\tau+1)} = w_{kj}^{(\tau)} - \eta \delta_{kn} \phi_j(\mathbf{x}_n)$$

$$\delta_{kn} = \frac{\partial g(a_k)}{\partial w_{kj}} (y_k(\mathbf{x}_n; \mathbf{w}) - t_{kn})$$

# Summary: Generalized Linear Discriminants

- **Properties**

- General class of decision functions.
- Nonlinearity  $g(\cdot)$  and basis functions  $\phi_j$  allow us to address linearly non-separable problems.
- Shown simple sequential learning approach for parameter estimation using gradient descent.
- Better 2<sup>nd</sup> order gradient descent approaches available (e.g. Newton-Raphson).

- **Limitations / Caveats**

- Flexibility of model is limited by curse of dimensionality
  - $g(\cdot)$  and  $\phi_j$  often introduce additional parameters.
  - Models are either limited to lower-dimensional input space or need to share parameters.
- Linearly separable case often leads to overfitting.
  - Several possible parameter choices minimize training error.



# References and Further Reading

- More information on Linear Discriminant Functions can be found in Chapter 4 of Bishop's book (in particular Chapter 4.1).

Christopher M. Bishop  
Pattern Recognition and Machine Learning  
Springer, 2006

