TECHNISCHE
UNIVERSITÄT
DARMSTADT

Statistical Machine Learning

Lecture 09: Classification

Kristian Kersting

TU Darmstadt

Summer Term 2020

Today’s Objectives

- Make you understand how to do build a discriminative classifier!
- Covered Topics:
  - Discriminant Functions
  - Multi-Class Classification
  - Fisher Discriminate Analysis
  - Perceptrons
  - Logistic Regression

Outline

1. Discriminant Functions

2. Fisher Discriminant Analysis

3. Perceptron Algorithm

4. Logistic Regression

5. Wrap-Up

Outline

1. **Discriminant Functions**

2. Fisher Discriminant Analysis

3. Perceptron Algorithm

4. Logistic Regression

5. Wrap-Up

Reminder of Bayesian Decision Theory

- We want to find the a-posteriori probability (posterior) of the class C_k given the observation (feature) x

$$p\left(C_k \mid x\right) = \frac{p\left(x \mid C_k\right) p\left(C_k\right)}{p\left(x\right)} = \frac{p\left(x \mid C_k\right) p\left(C_k\right)}{\sum_j p\left(x \mid C_j\right) p\left(C_j\right)}$$

- $p\left(C_k \mid x\right)$ - class posterior
- $p\left(x \mid C_k\right)$ - class-conditional probability (likelihood)
- $p\left(C_k\right)$ - class prior
- $p\left(x\right)$ - normalization term

Reminder of Bayesian Decision Theory

- Decision rule
  - Decide C_1 if $p\left(C_1 \mid x\right) > p\left(C_2 \mid x\right)$
  - Equivalent to

$$p\left(x \mid C_1\right) p\left(C_1\right) > p\left(x \mid C_2\right) p\left(C_2\right) \equiv \frac{p\left(x \mid C_1\right)}{p\left(x \mid C_2\right)} > \frac{p\left(C_2\right)}{p\left(C_1\right)}$$

- A classifier obeying this rule is called a Bayes optimal classifier

Reminder of Bayesian Decision Theory

- Current approach
  - $p(C_k \mid x) = p(x \mid C_k)\, p(C_k) / p(x)$
  - Model and estimate the class-conditional density $p(x \mid C_k)$ and the class prior $p(C_k)$
  - Compute posterior $p(C_k \mid x)$
  - Minimize the error probability by maximizing $p(C_k \mid x)$
- New approach
  - Directly encode the *decision boundary*
  - Without modeling the densities directly
  - Still minimize the error probability

Discriminant Functions

- Formulate classification using comparisons
  - Discriminant functions

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

  - Classify x as class C_k iff

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

- More formally, a **discriminant** maps a vector **x** to one of the K available classes

Discriminant Functions

- Example of discriminant functions from the Bayes classifier

$$
\begin{aligned}
y_k(x) &= p(C_k \mid x) \\
y_k(x) &= p(x \mid C_k)\, p(C_k) \\
y_k(x) &= \log p(x \mid C_k) + \log p(C_k)
\end{aligned}
$$

Discriminant Functions

- Base case with 2 classes

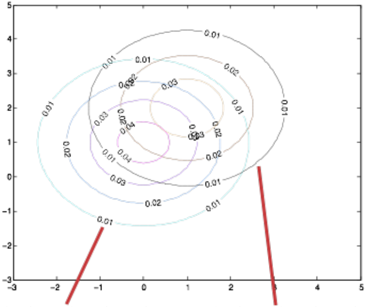
$$\begin{aligned} y_1(x) &> y_2(x) \\ y_1(x) - y_2(x) &> 0 \\ y(x) &> 0 \end{aligned}$$

- Example from the Bayes classifier

$$\begin{aligned} y(x) &= p(C_1 \mid x) - p(C_2 \mid x) \\ y(x) &= \log \frac{p(x \mid C_1)}{p(x \mid C_2)} + \log \frac{p(C_1)}{p(C_2)} \end{aligned}$$

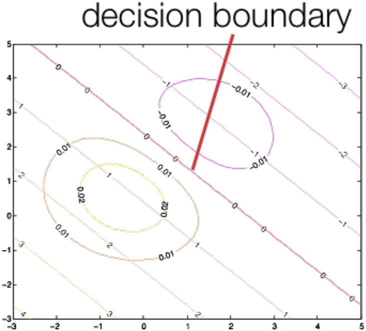
Example - Bayes Classifier

- Base case with 2 classes and Gaussian class-conditionals



$p(x|C_1)p(C_1)$ $p(x|C_2)p(C_2)$

decision boundary



$p(x|C_1)p(C_1) - p(x|C_2)p(C_2)$

$$\log \frac{p(x|C_1)p(C_1)}{p(x|C_2)p(C_2)}$$

Linear Discriminant Functions

- Base case with 2 classes

$$y(\mathbf{x}) > 0 \quad \text{decide class 1, otherwise class 2}$$

- Simplest case: linear decision boundary
  - In ***linear* discriminants**, the decision surfaces are (hyper)planes
  - Linear Discriminant Function

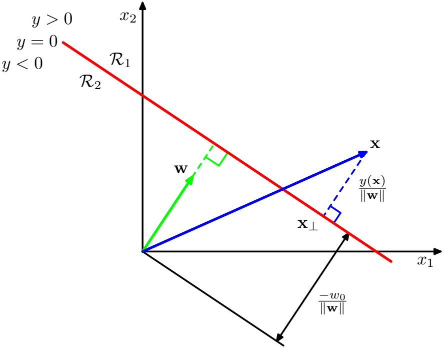
$$y(\mathbf{x}) = \mathbf{w}^{\intercal}\mathbf{x} + w_0$$

  - Where **w** is the normal vector and w_0 the offset

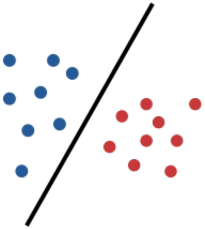
Linear Discriminant Functions

- Illustration of the 2D case

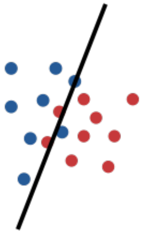
$$y(\mathbf{x}) = \mathbf{w}^{\mathsf{T}}\mathbf{x} + w_0, \quad \mathbf{x} = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^{\mathsf{T}}$$



Linear Discriminant Functions



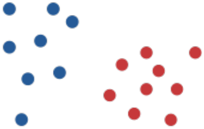
Linearly separable



Not linearly separable

Discriminant Functions

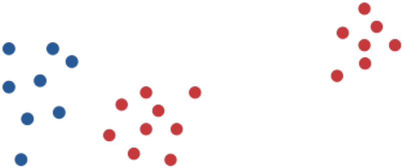
- Why might we want to use discriminant functions?



- We could easily fit the class-conditionals using Gaussians and use a Bayes classifier

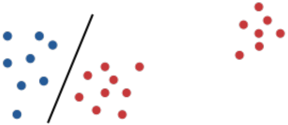
Discriminant Functions

- How about now? Do these points matter for making the decision between the two classes?



Distribution-free Classifiers

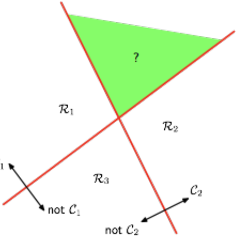
- We do not necessarily need to model all the details of the class-conditional distributions to come up with a good decision boundary. (The class-conditionals may have many intricacies that do not matter at the end of the day)



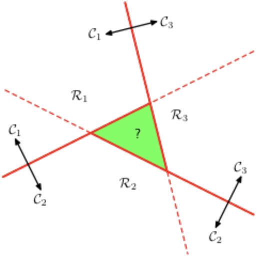
- If we can learn where to place the decision boundary directly, we can avoid some of the complexity
- It would be unwise to believe that such classifiers are inherently superior to probabilistic ones. We shall see why later...

Multi-Class Case

- What if we constructed a multi-class classifier from several 2-class classifiers?



One-versus-all
(one-versus-the-rest)



One-versus-one

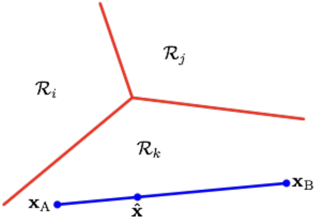
- If we base our decision rule on binary decisions, this may lead to ambiguities

Multi-Class Case - Better Solution

- Use a discriminant function to encode how strongly we believe in each class

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

- Decision rule: Decide k if $y_k(x) > y_j(x) \quad \forall j \neq k$



- If the discriminant functions are linear, the decision regions are connected and convex

Outline

1. Discriminant Functions

2. **Fisher Discriminant Analysis**

3. Perceptron Algorithm

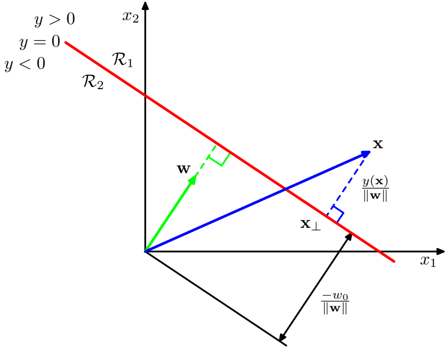
4. Logistic Regression

5. Wrap-Up

Linear Discriminant Functions

- Illustration of the 2D case

$$y(\mathbf{x}) = \mathbf{w}^\mathsf{T}\mathbf{x} + w_0, \quad \mathbf{x} = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^\mathsf{T}$$



First Attempt: Least Squares

- Try to achieve a certain value of the discriminative function

$$
\begin{aligned}
y(\boldsymbol{x}) = +1 \quad &\Leftrightarrow \quad \boldsymbol{x} \in C_1 \\
y(\boldsymbol{x}) = -1 \quad &\Leftrightarrow \quad \boldsymbol{x} \in C_2
\end{aligned}
$$

  - Training data inputs: $X = \left\{\boldsymbol{x}_1 \in \mathbb{R}^d, \ldots, \boldsymbol{x}_n\right\}$
  - Training data labels: $Y = \{y_1 \in \{-1, +1\}, \ldots, y_n\}$
- Linear Discriminant Function
  - Try to enforce $\boldsymbol{x}_i^\intercal \boldsymbol{w} + w_0 = y_i, \quad \forall i = 1, \ldots, n$
  - There is one linear equation for each training data point/label pair

First Attempt: Least Squares

- Linear system of equations

$$\boldsymbol{x}_i^\mathsf{T}\boldsymbol{w} + w_0 = y_i, \quad \forall i = 1, \ldots, n$$

  - Define $\hat{\boldsymbol{x}}_i = \begin{pmatrix} \boldsymbol{x}_i & 1 \end{pmatrix}^\mathsf{T} \in \mathbb{R}^{d\times 1}$, $\hat{\boldsymbol{w}} = \begin{pmatrix} \boldsymbol{w} & w_0 \end{pmatrix}^\mathsf{T} \in \mathbb{R}^{d\times 1}$
  - Rewrite the equation system

$$\hat{\boldsymbol{x}}_i^\mathsf{T}\hat{\boldsymbol{w}} = y_i, \quad \forall i = 1, \ldots, n$$

  - In matrix-vector notation we have

$$\hat{\boldsymbol{X}}^\mathsf{T}\hat{\boldsymbol{w}} = \boldsymbol{y}$$

    - With $\hat{\boldsymbol{X}} = [\hat{\boldsymbol{x}}_1, \ldots, \hat{\boldsymbol{x}}_n] \in \mathbb{R}^{d\times n}$ and $\boldsymbol{y} = [y_1, \ldots, y_n]^\mathsf{T}$

First Attempt: Least Squares

$$\hat{\boldsymbol{X}}^\intercal \hat{\boldsymbol{w}} = \boldsymbol{y}$$

- An overdetermined system of equations
- There are n equations and $d+1$ unknowns

First Attempt: Least Squares

- Look for the least squares solution

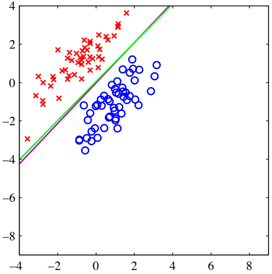
$$\begin{aligned}\hat{\boldsymbol{w}}^* &= \arg\min_{\hat{\boldsymbol{w}}} \left\| \hat{\boldsymbol{X}}^\top \hat{\boldsymbol{w}} - \boldsymbol{y} \right\|^2 \\ &= \arg\min_{\hat{\boldsymbol{w}}} \left(\hat{\boldsymbol{X}}^\top \hat{\boldsymbol{w}} - \boldsymbol{y} \right)^\top \left(\hat{\boldsymbol{X}}^\top \hat{\boldsymbol{w}} - \boldsymbol{y} \right) \\ &= \arg\min_{\hat{\boldsymbol{w}}} \hat{\boldsymbol{w}}^\top \hat{\boldsymbol{X}} \hat{\boldsymbol{X}}^\top \hat{\boldsymbol{w}} - 2\boldsymbol{y}^\top \hat{\boldsymbol{X}}^\top \hat{\boldsymbol{w}} + \boldsymbol{y}^\top \boldsymbol{y}\end{aligned}$$

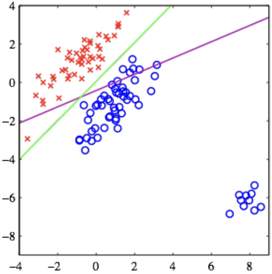
$$\nabla_{\hat{\boldsymbol{w}}} \left(\hat{\boldsymbol{w}}^\top \hat{\boldsymbol{X}} \hat{\boldsymbol{X}}^\top \hat{\boldsymbol{w}} - 2\boldsymbol{y}^\top \hat{\boldsymbol{X}}^\top \hat{\boldsymbol{w}} + \boldsymbol{y}^\top \boldsymbol{y} \right) = 0$$

$$\hat{\boldsymbol{w}} = \underbrace{\left(\hat{\boldsymbol{X}} \hat{\boldsymbol{X}}^\top \right)^{-1} \hat{\boldsymbol{X}}}_{\text{pseudo-inverse}} \boldsymbol{y}$$

First Attempt: Least Squares

- Problem: Least-squares is very sensitive to outliers



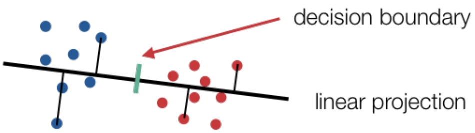


Without outliers least-squares discriminant works

With outliers least-squares discriminant breaks down

Fisher's Linear Discriminant

- Take a different view on linear classification
- Find a linear projection of our data and classify the projected values



- The same thing as a linear discriminant function
  - Projection: $y = \boldsymbol{w}^\intercal \boldsymbol{x}$
  - Checking against a threshold: $\boldsymbol{w}^\intercal \boldsymbol{x} \geq -w_0$ or $\boldsymbol{w}^\intercal \boldsymbol{x} + w_0 \geq 0$

Fisher's Linear Discriminant

- What is a good projection $\boldsymbol{w}$?
  - Idea: Maximize the "distance" between the two classes to allow for a good separation

- First attempt: Maximize the distance between the class means

$$\boldsymbol{m}_1 = \frac{1}{|C_1|} \sum_{i \in C_1} \boldsymbol{x}_i \quad \boldsymbol{m}_2 = \frac{1}{|C_2|} \sum_{i \in C_2} \boldsymbol{x}_i$$

  - Projection of the means

$$m_1 = \boldsymbol{w}^\intercal \boldsymbol{m}_1 \quad m_2 = \boldsymbol{w}^\intercal \boldsymbol{m}_2$$

  - Maximize squared distance between means

$$\max \left(m_1 - m_2\right)^2$$

Fisher's Linear Discriminant

- Maximize squared distance between means

$$\boldsymbol{w}^* = \arg\max_{\boldsymbol{w}} \left(\boldsymbol{w}^\intercal \boldsymbol{m}_1 - \boldsymbol{w}^\intercal \boldsymbol{m}_2\right)^2$$

  - Obvious problem: Grows unboundedly with the norm of ***w***
  - Obvious solution: Fix the norm of ***w***

$$\begin{aligned} \max_{\boldsymbol{w}} \quad & \left(\boldsymbol{w}^\intercal \boldsymbol{m}_1 - \boldsymbol{w}^\intercal \boldsymbol{m}_2\right)^2 \\ \text{s.t.} \quad & \|\boldsymbol{w}\|^2 = 1 \end{aligned}$$

  - Constrained optimization problem!

Fisher’s Linear Discriminant

$$\begin{aligned} \max_{\boldsymbol{w}} \quad & \left(\boldsymbol{w}^\intercal \boldsymbol{m}_1 - \boldsymbol{w}^\intercal \boldsymbol{m}_2\right)^2 \\ \text{s.t.} \quad & \left\|\boldsymbol{w}\right\|^2 = 1 \end{aligned}$$

- Necessary conditions

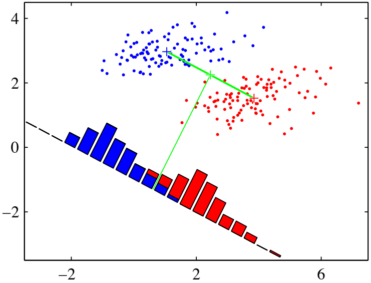
$$\begin{aligned} \nabla_{\boldsymbol{x}} f\left(\boldsymbol{x}\right) + \lambda \nabla_{\boldsymbol{x}} g\left(\boldsymbol{x}\right) &= 0 \\ 2\left(\boldsymbol{w}^\intercal \boldsymbol{m}_1 - \boldsymbol{w}^\intercal \boldsymbol{m}_2\right)\left(\boldsymbol{m}_1 - \boldsymbol{m}_2\right) + 2\lambda \boldsymbol{w} &= 0 \end{aligned}$$

- It follows that

$$\boldsymbol{w} = \frac{\boldsymbol{m}_1 - \boldsymbol{m}_2}{\left\|\boldsymbol{m}_1 - \boldsymbol{m}_2\right\|}$$

Fisher's Linear Discriminant

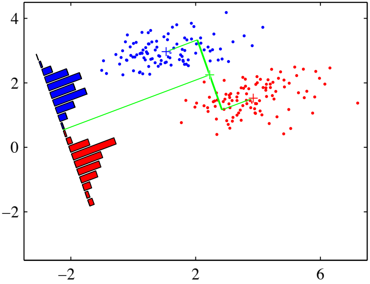
- Here's what we get



- Obvious problem: large class overlap

Fisher's Linear Discriminant

- Here's what we could get



- Much better separation between classes
- How do we get this?
  - Idea: Separate the means as far as possible while minimizing the variance of each class

Fisher's Linear Discriminant

- Second (and final) attempt:
  - Define within-class variances:

$$s_1^2 = \sum_{n \in C_1} \left(\boldsymbol{w}^\intercal \boldsymbol{x}_n - m_1\right)^2 \quad s_2^2 = \sum_{n \in C_2} \left(\boldsymbol{w}^\intercal \boldsymbol{x}_n - m_2\right)^2$$

where $m_1 = \boldsymbol{w}^\intercal \boldsymbol{m}_1$ and $m_2 = \boldsymbol{w}^\intercal \boldsymbol{m}_2$

- Fisher criterion

$$\max_{\mathbf{w}} J\left(\boldsymbol{w}\right) = \frac{\left(m_1 - m_2\right)^2}{s_1^2 + s_2^2}$$

Fisher's Linear Discriminant

- Fisher criterion

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

- Rewrite the numerator

$$\begin{aligned}(m_1 - m_2)^2 &= (\boldsymbol{w}^\intercal \boldsymbol{m}_1 - \boldsymbol{w}^\intercal \boldsymbol{m}_2)^2 \\ &= (\boldsymbol{w}^\intercal (\boldsymbol{m}_1 - \boldsymbol{m}_2))^2 \\ &= \boldsymbol{w}^\intercal \underbrace{(\boldsymbol{m}_1 - \boldsymbol{m}_2)(\boldsymbol{m}_1 - \boldsymbol{m}_2)^\intercal}_{\underbrace{=:}_{\text{between-class covariance}} \boldsymbol{S}_B} \boldsymbol{w}\end{aligned}$$

Fisher's Linear Discriminant

- Fisher criterion

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

- Rewrite the denominator

$$\begin{aligned}
s_1^2 + s_2^2 &= \sum_{n \in C_1} (\boldsymbol{w}^\intercal \boldsymbol{x}_n - m_1)^2 + \sum_{n \in C_2} (\boldsymbol{w}^\intercal \boldsymbol{x}_n - m_2)^2 \\
&= \sum_{n \in C_1} (\boldsymbol{w}^\intercal (\boldsymbol{x}_n - \boldsymbol{m}_1))^2 + \sum_{n \in C_2} (\boldsymbol{w}^\intercal (\boldsymbol{x}_n - \boldsymbol{m}_2))^2 \\
&= \sum_{n \in C_1} \boldsymbol{w}^\intercal (\boldsymbol{x}_n - \boldsymbol{m}_1)(\boldsymbol{x}_n - \boldsymbol{m}_1)^\intercal \boldsymbol{w} + \sum_{n \in C_2} \boldsymbol{w}^\intercal (\boldsymbol{x}_n - \boldsymbol{m}_2)(\boldsymbol{x}_n - \boldsymbol{m}_2)^\intercal \boldsymbol{w} \\
&= \boldsymbol{w}^\intercal \underbrace{\left[\sum_{n \in C_1} (\boldsymbol{x}_n - \boldsymbol{m}_1)(\boldsymbol{x}_n - \boldsymbol{m}_1)^\intercal + \sum_{n \in C_2} (\boldsymbol{x}_n - \boldsymbol{m}_2)(\boldsymbol{x}_n - \boldsymbol{m}_2)^\intercal \right] \boldsymbol{w}}_{\underbrace{=: \boldsymbol{S}_W}_{\text{within-class covariance}}}
\end{aligned}$$

Fisher's Linear Discriminant

- Fisher criterion

$$\max_{\mathbf{w}} J(\boldsymbol{w}) = \frac{(m_1 - m_2)^2}{s_1^2 + s_2^2} = \frac{\boldsymbol{w}^\intercal \boldsymbol{S}_B \boldsymbol{w}}{\boldsymbol{w}^\intercal \boldsymbol{S}_W \boldsymbol{w}}$$

- Differentiating w.r.t. $\boldsymbol{w}$ and setting to 0 we have

$$(\boldsymbol{w}^\intercal \boldsymbol{S}_B \boldsymbol{w}) \boldsymbol{S}_W \boldsymbol{w} = (\boldsymbol{w}^\intercal \boldsymbol{S}_W \boldsymbol{w}) \boldsymbol{S}_B \boldsymbol{w}$$

- Since $(\boldsymbol{w}^\intercal \boldsymbol{S}_B \boldsymbol{w})$ and $(\boldsymbol{w}^\intercal \boldsymbol{S}_W \boldsymbol{w})$ are scalars, we have that

$$\boldsymbol{S}_W \boldsymbol{w} \parallel \boldsymbol{S}_B \boldsymbol{w}$$

where $\parallel$ means collinearity

Fisher's Linear Discriminant

- Also, we know that

$$\boldsymbol{S}_B \boldsymbol{w} = (\boldsymbol{m}_1 - \boldsymbol{m}_2)(\boldsymbol{m}_1 - \boldsymbol{m}_2)^\intercal \boldsymbol{w} \implies \boldsymbol{S}_B \boldsymbol{w} \parallel (\boldsymbol{m}_1 - \boldsymbol{m}_2)$$

- Hence, we have

$$\begin{aligned} \boldsymbol{S}_W \boldsymbol{w} &\parallel (\boldsymbol{m}_1 - \boldsymbol{m}_2) \\ \boldsymbol{w} &\parallel \boldsymbol{S}_W^{-1}(\boldsymbol{m}_1 - \boldsymbol{m}_2) \end{aligned}$$

- Fisher's Linear Discriminant

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

Fisher's Linear Discriminant

$$\boldsymbol{w} \quad \propto \quad \boldsymbol{S}_W^{-1}\left(\boldsymbol{m}_1 - \boldsymbol{m}_2\right)$$

- The Fisher linear discriminant only gives us a projection
  - We still need to find the threshold
  - E.g., use Bayes classifier with Gaussian class-conditionals
- Bayes optimality
  - Fisher's linear discriminant is Bayes optimal if the class-conditional distributions are equal, with diagonal covariance
- Essentially equivalent to Linear Discriminant Analysis (LDA)

Fisher’s Linear Discriminant

- We won’t go through this here, but Fisher’s linear discriminant can be shown to be equivalent to a certain case of a least-squares linear classifier (see Bishop 4.1.5)
- Problem with this method: it is still very sensitive to noise!
- By The Way: This method is a true classic (it dates back to 1936)
  - Fisher, R.A., *The Use of Multiple Measurements in Taxonomic Problems*. Annals of Eugenics, 7: 179-188 (1936)

Outline

1. Discriminant Functions

2. Fisher Discriminant Analysis

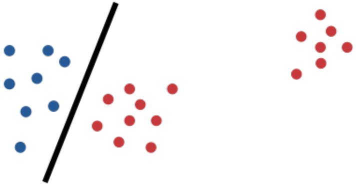
3. Perceptron Algorithm

4. Logistic Regression

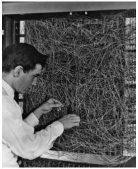
5. Wrap-Up

New Strategy

- If our classes are linearly separable, we want to make sure that we find a separating (hyper)plane



- First such algorithm we will see
  - The perceptron algorithm [Rosenblatt, 1962]



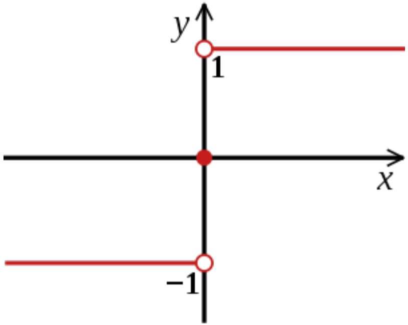
Rosenblatt [1928-1971]

Perceptron Algorithm

- Perceptron discriminant function

$$y(\boldsymbol{x}) = \operatorname{sign}\left(\boldsymbol{w}^{\intercal}\boldsymbol{x} + b\right)$$

  - where $\operatorname{sign}(x) = \{+1,\ x > 0;\ 0,\ x = 0;\ -1,\ x < 0\}$



Perceptron Algorithm

- **Perceptron Algorithm**
  - Initialize the weight vector $\boldsymbol{w}$ and bias b
  - For all pairs of data points $(\boldsymbol{x}_i, y_i)$, where $y_i \in \{-1, +1\}$, do
    - If $\boldsymbol{x}_i$ is correctly classified, i.e., $y(\boldsymbol{x}_i) = y_i$, do nothing
    - Else if $y_i = 1$ update the parameters with

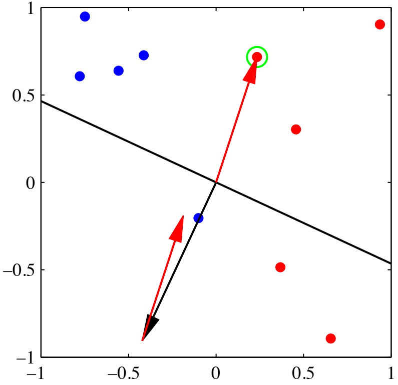
      $$\boldsymbol{w} \leftarrow \boldsymbol{w} + \boldsymbol{x}_i, \quad b \leftarrow b + 1$$

    - Else if $y_i = -1$ update the parameters with

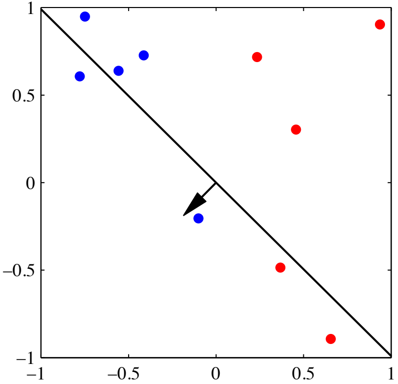
      $$\boldsymbol{w} \leftarrow \boldsymbol{w} - \boldsymbol{x}_i, \quad b \leftarrow b - 1$$

    - Repeat until convergence

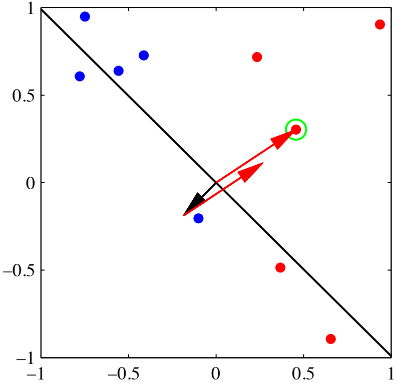
Perceptron Algorithm - Intuition



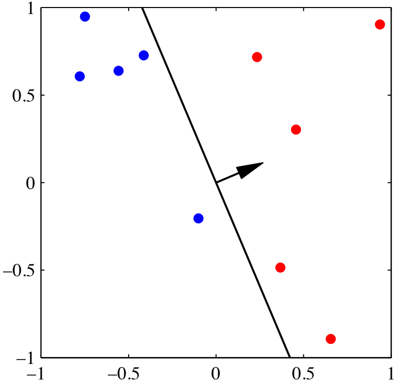
Perceptron Algorithm - Intuition



Perceptron Algorithm - Intuition



Perceptron Algorithm - Intuition



Perceptron Algorithm

- Why does this algorithm work?
- We have an optimization problem

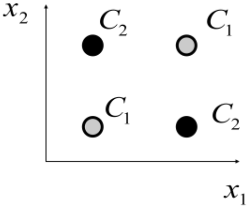
$$\max_{\mathbf{w}} J(\boldsymbol{w}) = \left|\{x \in X : \langle w, x \rangle < 0\}\right| = \sum_{x \in X : \langle w, x \rangle < 0} \langle w, x \rangle$$

- And also a gradient method

$$\frac{\partial J}{\partial w} = \sum_{x \in X : \langle w, x \rangle < 0} x$$

But is the Perceptron Algorithm useful?

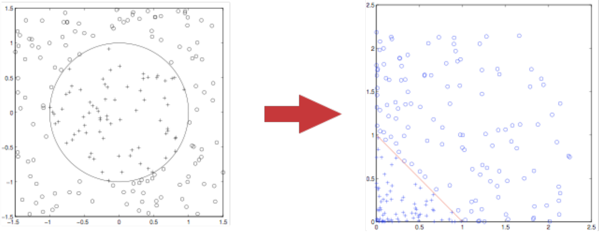
- How often is data linearly separable?
- A simple failure example is the XOR function



- History: Minsky & Papert [1969] criticized the perceptron for not being able to handle this case, which halted research on this and related techniques for decades

Other Feature Spaces

- It took a long time until people had realized that there is a simple way out
- Key idea: Transform the input data nonlinearly so that the problem becomes linearly separable!



- There is an important message to get out from this
  - Create features instead of learning from raw data
  - Neural networks do it *automagically* for you

Outline

1. Discriminant Functions

2. Fisher Discriminant Analysis

3. Perceptron Algorithm

4. **Logistic Regression**

5. Wrap-Up

Generative vs. Discriminative

- There are two different views to solve the classification problem
- Generative modelling
  - We model the class-conditional distributions $p(x \mid C_2)$ and $p(x \mid C_1)$
  - We classify by computing the class posterior using Bayes' rule
  - E.g.: Naive Bayes
- Discriminative modelling
  - We model the class-posterior directly, e.g. $p(C_1 \mid x)$
  - Consequence: We only care about getting the classification right, and not whether we fit the class-conditional well
  - E.g.: Logistic Regression

Probabilistic Discriminative Models

- For now, we will write the class posterior using Bayes' rule

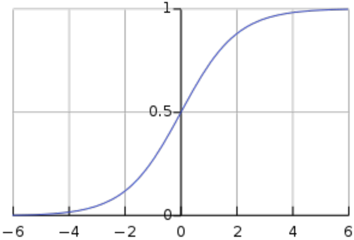
$$
\begin{aligned}
p\left(C_1 \mid \boldsymbol{x}\right) &= \frac{p\left(\boldsymbol{x} \mid C_1\right) p\left(C_1\right)}{p\left(\boldsymbol{x}\right)} = \frac{p\left(\boldsymbol{x} \mid C_1\right) p\left(C_1\right)}{\sum_i p\left(\boldsymbol{x}, C_i\right)} \\
&= \frac{p\left(\boldsymbol{x} \mid C_1\right) p\left(C_1\right)}{\sum_i p\left(\boldsymbol{x} \mid C_i\right) p\left(C_i\right)} \\
&= \frac{p\left(\boldsymbol{x} \mid C_1\right) p\left(C_1\right)}{p\left(\boldsymbol{x} \mid C_1\right) p\left(C_1\right) + p\left(\boldsymbol{x} \mid C_2\right) p\left(C_2\right)} \\
&= \frac{1}{1 + p\left(\boldsymbol{x} \mid C_2\right) p\left(C_2\right) / \left(p\left(\boldsymbol{x} \mid C_1\right) p\left(C_1\right)\right)} \\
&= \frac{1}{1 + \exp\left(-a\right)} = \sigma\left(a\right) \rightarrow \text{logistic sigmoid function}
\end{aligned}
$$

with $a = \log \frac{p(\boldsymbol{x}|C_1)p(C_1)}{p(\boldsymbol{x}|C_2)p(C_2)}$

Sigmoid

- Logistic / Sigmoid function

$$\sigma(a) = \frac{1}{1 + \exp(-a)}$$



[Wikipedia]

- Sigmoid: ’S-shaped’
- Squashes real numbers into the $[0, 1]$ interval

Probabilistic Discriminative Models

- Class posterior

$$p\left(C_1 \mid \boldsymbol{x}\right) = \sigma\left(a\right) \quad \text{with} \quad a = \log \frac{p\left(\boldsymbol{x} \mid C_1\right) p\left(C_1\right)}{p\left(\boldsymbol{x} \mid C_2\right) p\left(C_2\right)}$$

- Logistic regression
  - Assume that a is given by a linear discriminant function

$$p(C_1 \mid \boldsymbol{x}) = \sigma(\boldsymbol{w}^\intercal \boldsymbol{x} + w_0)$$

  - Find **w** and w_0 so that the class-posterior is modeled best
  - When is this an appropriate assumption?
    - When the class conditionals are Gaussians with equal covariance
    - But also for a number of other distributions
    - Some independence of the form of the class-conditionals

Logistic Regression

- Model the class posterior as

$$p(C_1 \mid \boldsymbol{x}) = \sigma(\boldsymbol{w}^\intercal \boldsymbol{x} + w_0)$$

- Maximize the likelihood
  - Data (as always) is i.i.d. and define $y_i = \begin{cases} 0 & \boldsymbol{x}_i \text{ belongs to } C_1 \\ 1 & \boldsymbol{x}_i \text{ belongs to } C_2 \end{cases}$

$$\begin{aligned} p\left(Y \,\middle|\, X; \boldsymbol{w}, w_0\right) &= \prod_{i=1}^{N} p\left(y_i \,\middle|\, \boldsymbol{x}_i; \boldsymbol{w}, w_0\right) \\ &= \prod_{i=1}^{N} p\left(C_1 \,\middle|\, \boldsymbol{x}_i; \boldsymbol{w}, w_0\right)^{1-y_i} p\left(C_2 \,\middle|\, \boldsymbol{x}_i; \boldsymbol{w}, w_0\right)^{y_i} \\ &= \prod_{i=1}^{N} \sigma(\boldsymbol{w}^\intercal \boldsymbol{x}_i + w_0)^{1-y_i} \left(1 - \sigma(\boldsymbol{w}^\intercal \boldsymbol{x}_i + w_0)\right)^{y_i} \end{aligned}$$

Logistic Regression

- We won’t do the derivation here (see Bishop 4.3), but basically you can apply the logarithm to $p\left(Y \,\middle|\, X; \boldsymbol{w}, w_0\right)$ and do gradient descent
- Similar to what we have seen in regression, we can get more robust classifiers by incorporating priors and taking a Bayesian approach
- Later, we will turn to a very different interpretation of this:
  - Logistic regression as a neural network

Outline

1. Discriminant Functions

2. Fisher Discriminant Analysis

3. Perceptron Algorithm

4. Logistic Regression

5. Wrap-Up

5. Wrap-Up

You know now:

- What a Bayesian Optimal Classifier is
- What a discriminant function is
- How to formalize (with intuition and mathematically) the classification problem as linearly-separable
- How to compute the least squares solution for classification and why it fails
- What Fisher’s Linear Discriminant is and how it differs from least-squares
- What the perceptron is, why it fails in the XOR problem and how to overcome it with feature spaces
- The difference between Generative and Discriminative modelling
- What logistic regression is

Self-Test Questions

- How do we get from Bayesian optimal decisions to discriminant functions?
- How to derive a discriminant function from a probability distribution?
- How to deal with more than two classes?
- What does linearly-separable mean?
- What is Fisher discriminant analysis? How does it relate to regression?
- Is Fisher's linear discriminant Bayes optimal?
- What are perceptrons? How can we train them?
- What is logistic regression? How to derive the parameter update rule?

Homework

- Reading Assignment for next week
  - Bishop 7.1.5 and 12.1
  - Murphy 6.5 and 12.2