Machine Learning

Lecture 3
Simple Classifiers: Nearest Centroids and Linear Classification

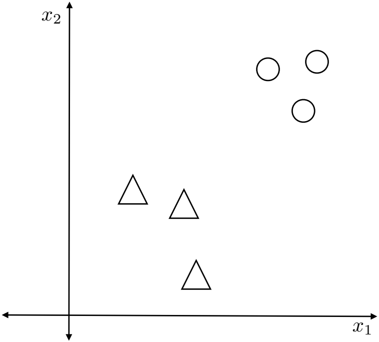
Felix Bießmann

Beuth University & Einstein Center for Digital Future

Overview of today's lecture

- Today we will introduce three simple classifiers
  1. Nearest Centroid Classifier (NCC)
  2. Perceptron
  3. K-Nearest Neighbor (KNN)
- These algorithms are extremely powerful
- Often they can compete with complex algorithms
- Some aspects can be motivated by biological cognition

Prototypes: Psychological Models of Abstract Ideas



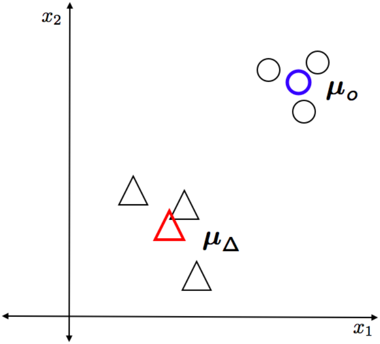
Psychologists postulated that we learn **prototypes** [??]

Toy data example:

Two dimensional input $\mathbf{x} \in \mathbb{R}^2$

Two *classes* of data, Δ and $\circ$

Prototypes: Psychological Models of Abstract Ideas



Prototypes $\boldsymbol{\mu}_\Delta$ and $\boldsymbol{\mu}_o$ can be the class means

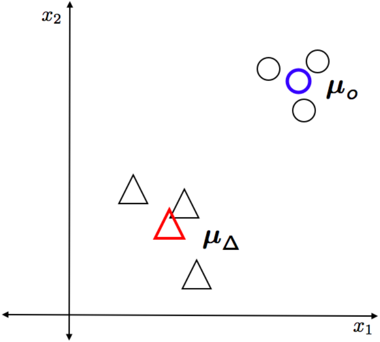
$$\boldsymbol{\mu}_\Delta = 1/N_\Delta \sum_n^{N_\Delta} \mathbf{x}_{\Delta,n}$$

$$\boldsymbol{\mu}_o = 1/N_o \sum_n^{N_o} \mathbf{x}_{o,n}$$

Distance from w_Δ to new data x

$$\|\boldsymbol{\mu}_\Delta - \mathbf{x}\|_2$$

Prototypes: Psychological Models of Abstract Ideas



For new data x check:
Is x more similar to μ_o?

$$\|\boldsymbol{\mu}_\Delta - \mathbf{x}\| > \|\boldsymbol{\mu}_o - \mathbf{x}\|$$

yes? $\rightarrow$ **x** belongs to $\boldsymbol{\mu}_o$

no? $\rightarrow$ **x** belongs to $\boldsymbol{\mu}_\Delta$

This is called a
nearest centroid classifier

Nearest Centroid Classification Algorithm (Batch Mode)

Algorithm 1 Computation of Class-Centroids

Require: data $\mathbf{x}_1, \dots, \mathbf{x}_N \in \mathbb{R}^D$, labels $y_1, \dots, y_N \in \{1, \dots, K\}$
Ensure: Class means $\boldsymbol{\mu}_k, \; k \in \{1, \dots, K\}$
1: # Initialize means and counters for each class
2: # Computation of class means
3: **for** Class $k = 1, \dots, K$ **do**
4: $\quad \boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{i=1}^{N_k} \mathbf{x}_i$
5: **end for**

Batch Computations vs. Streaming

Solutions for algorithms can be obtained

- In Batch Mode:
  - Use all available data at once
  - Requires to store all data in memory
- In Streaming Mode:
  - Use one data point at a time
  - Requires to store **only** centroids

Iterative Computation of the Mean

Given the mean $\boldsymbol{\mu}_{N-1}$ computed from $N-1$ samples we want to update $\boldsymbol{\mu}_{N-1}$ with the Nth sample $\mathbf{x}_N$ to obtain $\boldsymbol{\mu}_N$

$$
\begin{aligned}
\boldsymbol{\mu}_N =& \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n \\
=& \frac{1}{N}\sum_{n=1}^{N-1}\mathbf{x}_n + \frac{1}{N}\mathbf{x}_N \\
=& \frac{N-1}{N}\underbrace{\frac{1}{N-1}\sum_{n=1}^{N-1}\mathbf{x}_n}_{\boldsymbol{\mu}_{N-1}} + \frac{1}{N}\mathbf{x}_N \\
=& \frac{N-1}{N}\boldsymbol{\mu}_{N-1} + \frac{1}{N}\mathbf{x}_N
\end{aligned}
$$

Nearest Centroid Classification Algorithm (Streaming)

Algorithm 2 Iterative computation of Class-Centroids

Require: data $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbb{R}^D$, labels $y_1, \ldots, y_N \in \{1, \ldots, K\}$
Ensure: Class means $\boldsymbol{\mu}_k,\ k \in \{1, \ldots, K\}$

```
1: # Initialize means and counters for each class
2: ∀ k :   μ_k = I · 0,  N_k = 0
3: # Iterative computation of class means
4: for Data point i = 1, ..., N do
5:     # Update means and counters
6:     k = y_i
7:     μ_k = N_k/(N_k+1) μ_k + 1/(N_k+1) x_i
8:     N_k = N_k + 1
9: end for
```

Line 2: $\forall\, k : \quad \boldsymbol{\mu}_k = \mathbf{I} \cdot 0,\ N_k = 0$

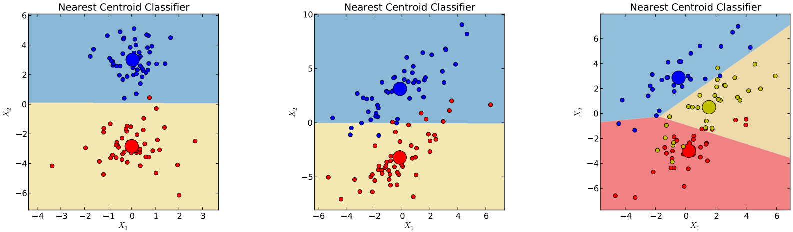
Line 7: $\boldsymbol{\mu}_k = \frac{N_k}{N_k+1}\ \boldsymbol{\mu}_k + \frac{1}{N_k+1}\ \mathbf{x}_i$

Nearest Centroid Classification

Algorithm 3 Nearest Centroid Prediction

Require: Data point $\mathbf{x} \in \mathbb{R}^D$, class centroids $\boldsymbol{\mu}_k,\ k \in \{1, \ldots, K\}$
Ensure: Class membership k^*
1: # Compute nearest class centroid in discriminative subspace
2: $k^* = \text{argmin}_k \|\boldsymbol{\mu}_k - \mathbf{x}\|_2$.

Toy Data Example NCC



From Prototypes to Linear Classification

$$
\begin{aligned}
\text{distance}(\mathbf{x}, \boldsymbol{\mu}_\Delta) >& \text{distance}(\mathbf{x}, \boldsymbol{\mu}_o) \\
\|\mathbf{x} - \boldsymbol{\mu}_\Delta\| >& \|\mathbf{x} - \boldsymbol{\mu}_o\|
\end{aligned} \tag{1}
$$

From Prototypes to Linear Classification

$$
\begin{aligned}
\text{distance}(\mathbf{x}, \boldsymbol{\mu}_\Delta) &> \text{distance}(\mathbf{x}, \boldsymbol{\mu}_o) \qquad (1)\\
\|\mathbf{x} - \boldsymbol{\mu}_\Delta\| &> \|\mathbf{x} - \boldsymbol{\mu}_o\| \\
\Leftrightarrow \|\mathbf{x} - \boldsymbol{\mu}_\Delta\|^2 &> \|\mathbf{x} - \boldsymbol{\mu}_o\|^2 \\
\Leftrightarrow \mathbf{x}^\top \mathbf{x} - 2\boldsymbol{\mu}_\Delta^\top \mathbf{x} + \boldsymbol{\mu}_\Delta^\top \boldsymbol{\mu}_\Delta &> \mathbf{x}^\top \mathbf{x} - 2\boldsymbol{\mu}_o^\top \mathbf{x} + \boldsymbol{\mu}_o^\top \boldsymbol{\mu}_o \\
\Leftrightarrow \boldsymbol{\mu}_\Delta^\top \mathbf{x} - \boldsymbol{\mu}_\Delta^2/2 &< \boldsymbol{\mu}_o^\top \mathbf{x} - \boldsymbol{\mu}_o^2/2 \\
\Leftrightarrow 0 &< \underbrace{(\boldsymbol{\mu}_o - \boldsymbol{\mu}_\Delta)}_{\mathbf{w}}{}^\top \mathbf{x} - 1/2\, \underbrace{(\boldsymbol{\mu}_o^\top \boldsymbol{\mu}_o - \boldsymbol{\mu}_\Delta^\top \boldsymbol{\mu}_\Delta)}_{\beta}
\end{aligned}
$$

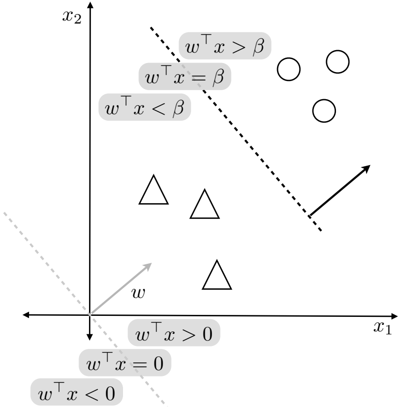
From Prototypes to Linear Classification

$$
\begin{aligned}
\text{distance}(\mathbf{x}, \boldsymbol{\mu}_\Delta) >& \text{distance}(\mathbf{x}, \boldsymbol{\mu}_o) \qquad (1)\\
\|\mathbf{x} - \boldsymbol{\mu}_\Delta\| >& \|\mathbf{x} - \boldsymbol{\mu}_o\| \\
\Leftrightarrow \|\mathbf{x} - \boldsymbol{\mu}_\Delta\|^2 >& \|\mathbf{x} - \boldsymbol{\mu}_o\|^2 \\
\Leftrightarrow \mathbf{x}^\top\mathbf{x} - 2\boldsymbol{\mu}_\Delta^\top\mathbf{x} + \boldsymbol{\mu}_\Delta^\top\boldsymbol{\mu}_\Delta >& \mathbf{x}^\top\mathbf{x} - 2\boldsymbol{\mu}_o^\top\mathbf{x} + \boldsymbol{\mu}_o^\top\boldsymbol{\mu}_o \\
\Leftrightarrow \boldsymbol{\mu}_\Delta^\top\mathbf{x} - \boldsymbol{\mu}_\Delta^2/2 <& \boldsymbol{\mu}_o^\top\mathbf{x} - \boldsymbol{\mu}_o^2/2 \\
\Leftrightarrow 0 <& \underbrace{(\boldsymbol{\mu}_o - \boldsymbol{\mu}_\Delta)}_{\mathbf{w}}{}^\top\mathbf{x} - 1/2\,\underbrace{(\boldsymbol{\mu}_o^\top\boldsymbol{\mu}_o - \boldsymbol{\mu}_\Delta^\top\boldsymbol{\mu}_\Delta)}_{\beta}
\end{aligned}
$$

Linear Classification

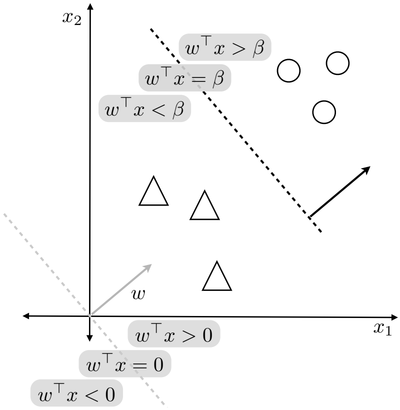
$$
\mathbf{w}^\top\mathbf{x} - \beta = \begin{cases} > 0 & \text{if } \mathbf{x} \text{ belongs to class } \circ \\ < 0 & \text{if } \mathbf{x} \text{ belongs to class } \Delta \end{cases} \qquad (2)
$$

Linear Classification



$$\mathbf{w}^\top \mathbf{x} - \beta = \begin{cases} > 0 & \text{if } \mathbf{x} \text{ belongs to } o \\ < 0 & \text{if } \mathbf{x} \text{ belongs to } \Delta \end{cases}$$

Linear Classification



$$\mathbf{w}^\top \mathbf{x} - \beta = \begin{cases} > 0 & \text{if } \mathbf{x} \text{ belongs to } o \\ < 0 & \text{if } \mathbf{x} \text{ belongs to } \Delta \end{cases}$$

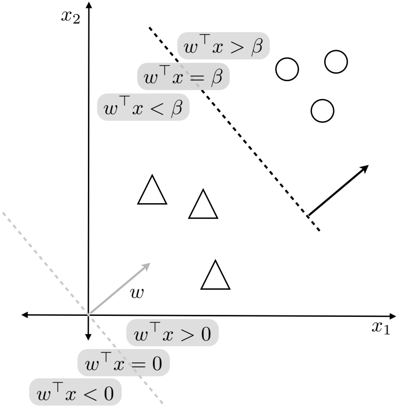
The *offset* β can be included in $\mathbf{w}$

$$\tilde{\mathbf{x}} \leftarrow \begin{bmatrix} 1 \\ \mathbf{x} \end{bmatrix} \qquad \tilde{\mathbf{w}} \leftarrow \begin{bmatrix} -\beta \\ \mathbf{w} \end{bmatrix}$$

such that

$$\tilde{\mathbf{w}}^\top \tilde{\mathbf{x}} = \mathbf{w}^\top \mathbf{x} - \beta.$$

Linear Classification



What is a good **w**?

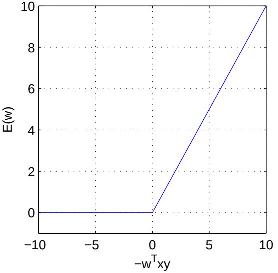
$\rightarrow$ We need an **error function** that tells us how good **w** is.

Two classical Error Functions

Given data $\mathbf{x} \in \mathbb{R}^D$ and corresponding labels $y \in \{-1, +1\}$, two classical error functions $\mathcal{E}(\mathbf{x}, y, \mathbf{w})$ to find the optimal $\mathbf{w} \in \mathbb{R}^D$ are:

Error Function	Used in
$\frac{1}{2}(y - \mathbf{w}^\top \mathbf{x})^2$	Adaline [?]
$\max(0, -y\mathbf{w}^\top \mathbf{x})$	Perceptron [?]

Classification Error as Function of Weights



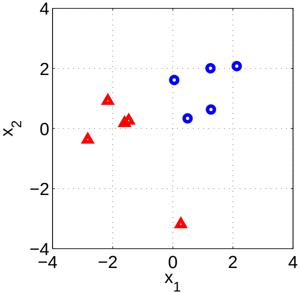
Given data $\mathbf{x} \in \mathbb{R}^D$ and corresponding labels $y \in \{-1, +1\}$ the classification error $\mathcal{E}$ is a function of the weights $\mathbf{w}$ (and the data $\mathbf{x}$, y)

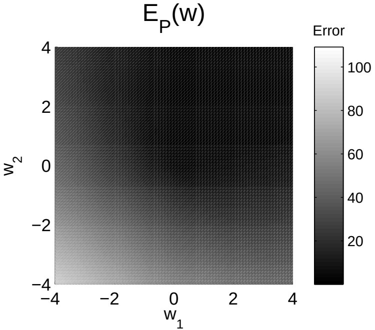
$$\mathcal{E}(\mathbf{w}, \mathbf{x}_m, y_m) = -\sum_{m \in \mathcal{M}} \mathbf{w}^\top \mathbf{x}_m y_m \quad (3)$$

where $\mathcal{M}$ denotes the index set of all *misclassified* data $\mathbf{x}_m$

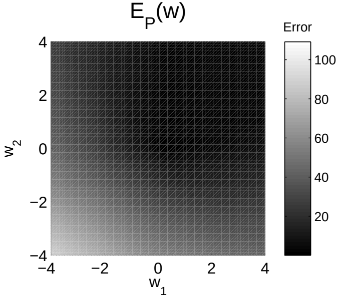
Classification Error as Function of Weights

Data $\mathbf{x} \in \mathbb{R}^2$





Gradient Descent

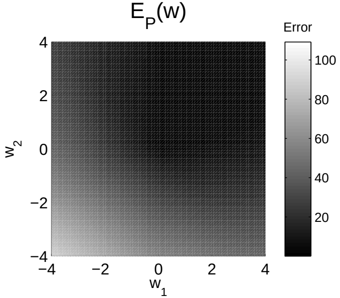


How to minimize the error function?

$$\mathcal{E}(\mathbf{w}, \mathbf{x}_m, y_m) = -\sum_{m \in \mathcal{M}} \mathbf{w}^\top \mathbf{x}_m y_m$$

$\rightarrow$ **Gradient Descent**

Gradient Descent

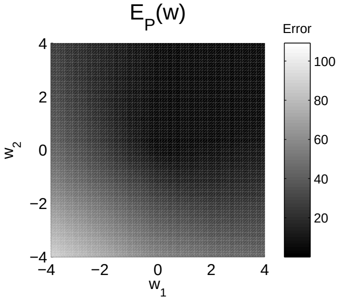


We minimize $\mathcal{E}(\mathbf{w}, \mathbf{x}_m, y_m)$ by walking in the opposite direction of the gradient.

$$\mathbf{w}^{\text{new}} \leftarrow \mathbf{w}^{\text{old}} - \eta \frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} \nabla \mathcal{E}(\mathbf{w}, \mathbf{x}_i, y_i)$$

where $\mathcal{X}$ is the set of data points and η is called a **learning rate**.

Stochastic Gradient Descent



A noisy estimate of

$$\frac{1}{|\mathcal{X}|}\sum_{i=1}^{|\mathcal{X}|}\nabla\mathcal{E}(\mathbf{w},\mathbf{x},y)$$

is obtained by [?]

$$\mathbf{w}^{\text{new}} \leftarrow \mathbf{w}^{\text{old}} - \eta\nabla\mathcal{E}(\mathbf{w},\mathbf{x}_i,y_i)$$

Note that only **w** is stored and only one data point $\mathbf{x}_i$ and label y_i are considered at a time!

$\rightarrow$ Scales to large data sets [?]

References