Machine Learning

Lecture 3
Simple Classifiers: Nearest Centroids and KNN

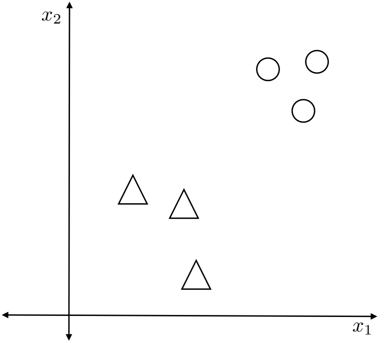
Felix Bießmann

Beuth University & Einstein Center for Digital Future

Overview of today's lecture

- Today we will introduce two simple classifiers
  1. Nearest Centroid Classifier (NCC)
  2. K-Nearest Neighbor (KNN)
- These algorithms are extremely powerful
- Often they can compete with complex algorithms

Prototypes: Psychological Models of Abstract Ideas



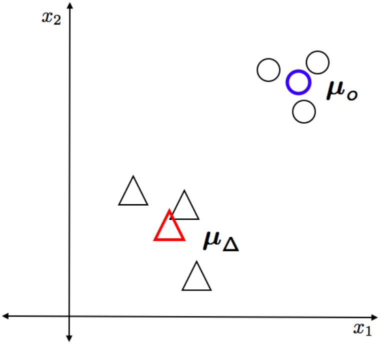
Psychologists postulated that we learn **prototypes** [Jaekel, 2007; Posner and Keele, 1968]

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

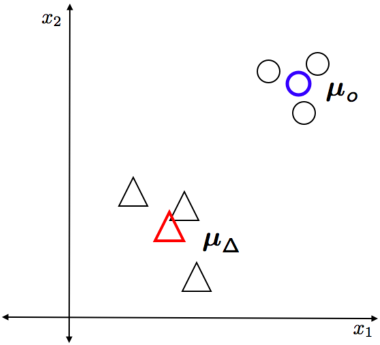
$$\boldsymbol{\mu}_\Delta = 1/N_\Delta \sum_n^{N_\Delta} \mathbf{x}_{\Delta,n}$$

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

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 $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbb{R}^D$, 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: # Computation of class means
3: **for** Class $k = 1, \ldots, K$ **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

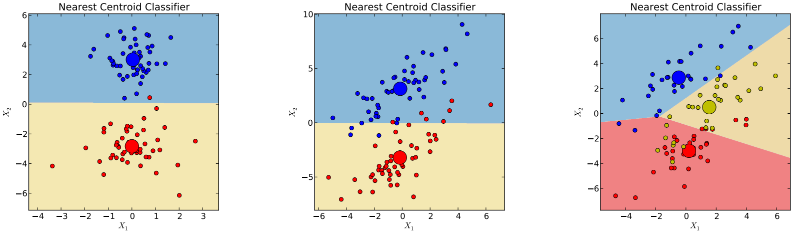
```
Require: data x_1, ..., x_N ∈ R^D, labels y_1, ..., y_N ∈ {1, ..., K}
Ensure: Class means μ_k, k ∈ {1, ..., K}
# Initialize means and counters for each class
∀ k :   μ_k = I · 0,  N_k = 0
# Iterative computation of class means
for Data point i = 1, ..., N do
    # Update means and counters
    k = y_i
    μ_k = N_k/(N_k+1) μ_k + 1/(N_k+1) x_i
    N_k = N_k + 1
end for
```

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
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

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

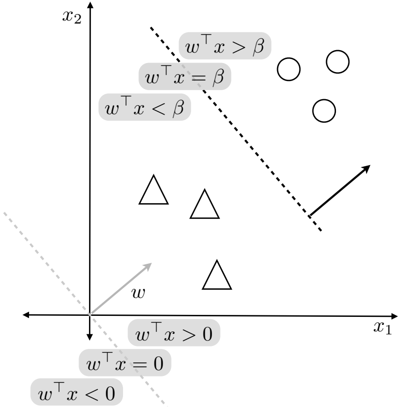
From Prototypes to Linear Classification

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

Linear Classification

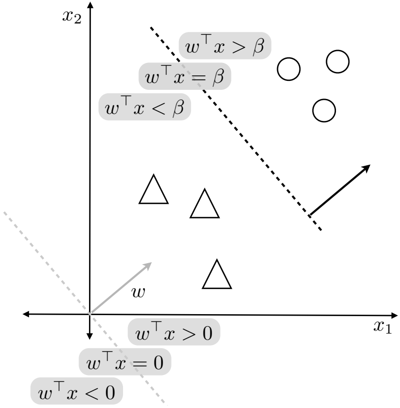
$$
\mathbf{w}^\top\mathbf{x} - \beta = \begin{cases} > 0 & \text{if } \mathbf{x} \text{ belongs to class } \circ \\ < 0 & \text{if } \mathbf{x} \text{ belongs to class } \Delta \end{cases} \qquad (2)
$$

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



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

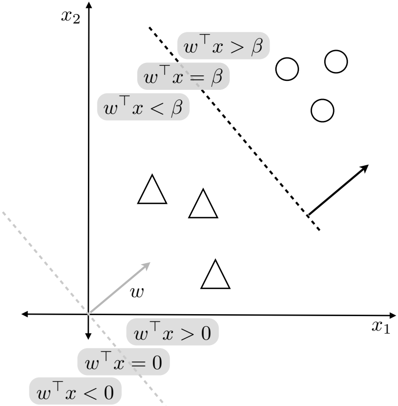
The *offset* β can be included in $\mathbf{w}$

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

such that

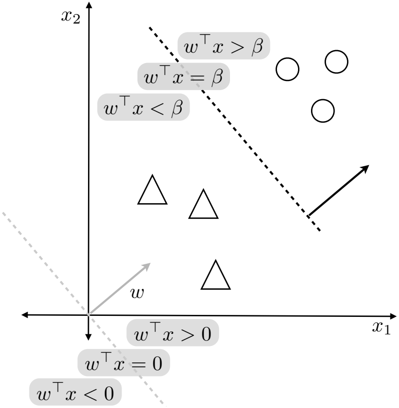
$$\tilde{\mathbf{w}}^\top \tilde{\mathbf{x}} = \mathbf{w}^\top \mathbf{x} - \beta.$$

Linear Classification



- What is a good **w**?
- Some proposals:
  - Logistic Regression
  - Perceptrons
  - Support Vector Machines
  - Ridge Regression
- Linear methods have different ways of defining what a good **w** is

Linear Classification



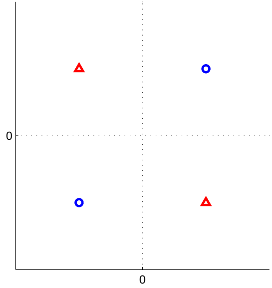
Nearest Centroid Classification is a simple linear classifier

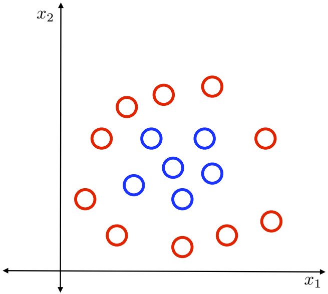
$$\mathbf{w} = \boldsymbol{\mu}_o - \boldsymbol{\mu}_\Delta \tag{3}$$

$$\beta = -1/2(\boldsymbol{\mu}_o^\top \boldsymbol{\mu}_o - \boldsymbol{\mu}_\Delta^\top \boldsymbol{\mu}_\Delta)$$

Problems with Linear Classification

Linear Classifiers fail on non-linear problems:

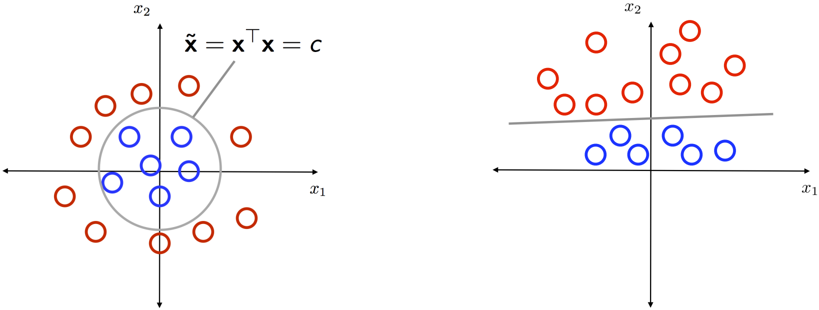




Try to separate these two-class data sets with **a single line**

Problems with Perceptrons

If we can model the non-linearity,
we can create new features $\tilde{\mathbf{x}}$ which are linearly separable



But what if we do not know the non-linearity?

K-Nearest Neighbor Classifier

- Nearest Centroid Classifiers require estimation of centroids
- K-Nearest Neighbor is simpler
- Idea:
    1. Find the k closest neighbors for a new data point $\mathbf{x}$
    2. Look up labels of k closest neighbors
    3. Assign majority vote label for $\mathbf{x}$

K-Nearest Neighbor Classifier

We do not need to train a model for KNN –
the data is the model
But (as with NCC) we need a distance function
Usually the euclidean distance is chosen:

$$d(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|_2 \qquad j \in [1, \ldots, N] \tag{4}$$

where N is the number of data points

K-Nearest Neighbor Classifier: Pseudocode

Algorithm 4 K-Nearest Neighbour Prediction

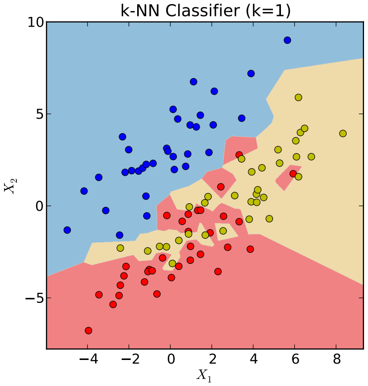
Require: Test data point $\mathbf{x}_i \in \mathbb{R}^D$, training data $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N] \in \mathbb{R}^{D \times N}$, corresponding labels $\mathbf{y} = [y_1, \ldots, y_N]^\top \in \mathbb{R}^N$, number of neighbours K

Ensure: Predicted label $\mathbf{y}_i$

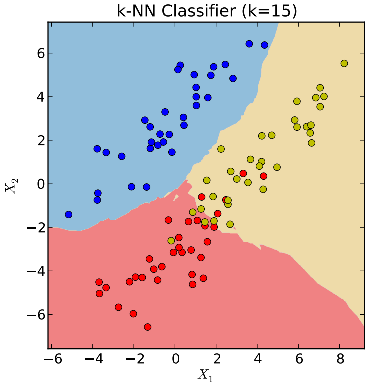
```
for x_j in X do
    # Compute distance between test data x_i and training data x_j
end for
Initialize a K-dimensional zero vector γ
for k in  do
    Count label of the k-nearest neighbor
    # γ_{y_k} ← γ_{y_k} + 1
end for
Predicted label is that with most votes (ties are broken at random)
y_i = argmax_k(γ)
```

K-Nearest Neighbor Classifier

Toy data problem: Linear classification



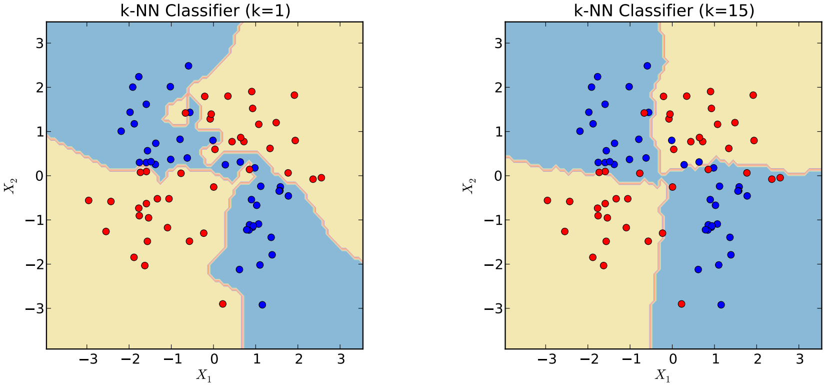
$k = 1$



$k = 15$

K-Nearest Neighbor Classifier

Toy data problem: Nonlinear classification



$k = 1$ $\quad$ $k = 15$

Problems with KNN

- Hyperparameter k needs to be set appropriately:
  - k small: complex decision boundaries
  - k large: smooth/simple decision boundaries
- Consider N data points $\mathbf{x} \in \mathbb{R}^D$
- Find K Neighbors requires O(NND) operations
- For large data sets this is too costly
- Speedups can be gained by:
  - Trees for distance computations
  - Locality Sensitive Hashing for finding neighbors

Summary

Psychologists postulated we learn **Prototypes**

- Prototypes can be the class means
- Prototype theory is closely related to linear classification

Nearest Centroid Classification

- New data is assigned to class with closest centroid
- Memory efficient: only requires to store D-dimensional centroids
- Iterative/Streaming version can scale to large data sets

K-Nearest Neighbor Classifier

- Simple nonlinear classifier
- State-of-the-art prediction performance
- No training required
- Needs to evaluate pairwise distances of **all** data points

$\rightarrow$ **Slow**

References

F. Jaekel. *Some Theoretical Aspects of Human Categorization Behaviour: Similarity and Generalization*. PhD thesis, 2007.

M. I. Posner and S. W. Keele. On the genesis of abstract ideas. *Journal of Experimental Psychology*, 77(3):353–363, 1968.