# IN2110: Språkteknologiske metoder

## *Klassifikasjon*

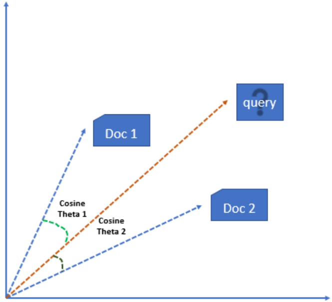

Erik Velldal

Språkteknologigruppen (LTG)


29. Januar, 2019

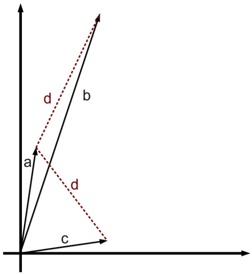
# Recap: Vector space models

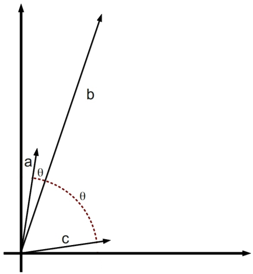
- General approach to representing data in a geometrical model.
- Example use case: vector spaces for IR.
- Documents represented as points/vectors in a feature space, $\boldsymbol{x} = \langle x_1, \ldots, x_n \rangle \in \mathbb{R}^n$.

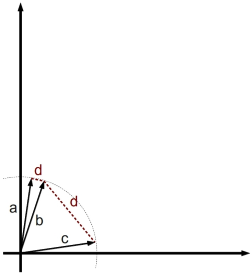



- Bag-of-words:
  - Documents represented by their unordered collection of word types.
  - Each dimension in the space corresponds to a word in the vocabulary.
- Similarity modeled by distance of documents (and queries) in the space.

# Recap: measuring similarity




- Euclidean distance between points.
- Cosine similarity of vector angles.
- Raw counts often weighted by TF-IDF.
- Can reduce length bias by normalization.

# Agenda

## Today

- Classification: supervised learning
- Rocchio
- $k$NN
- Representing classes and membership

## Next lecture

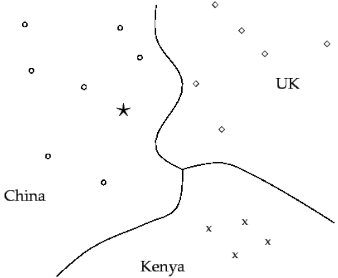
- Clustering: *un*supervised learning
- $c$-Means

# Example applications of text classification

- Topic classification of news articles
- Authorship attribution
- Spam detection
- Polarity classification (sentiment analysis)
- Language identification
- Abusive language detection
- Question type classification
- Content recommendation
- Political affiliation
- ...

# Classes and classification

- In our vector space model, objects are represented as points, so classes will correspond to collections of points; regions.
- Vector space classification is based on the contiguity hypothesis:




- Objects in the same class form a contiguous region, and regions of different classes do not overlap.
- Classification amounts to computing the boundaries in the space that separate the classes; *the decision boundaries*.
- How we draw the boundaries is influenced by how we choose to represent the classes.

# Different ways of representing classes

## Exemplar-based

- No abstraction. Every stored instance of a group can potentially represent the class.
- Used in so-called *instance based* or *memory based learning* (MBL).
- In its simplest form; the class = the collection of points.
- Another variant is to use *medoids*, – representing a class by a single member that is considered central, typically the object with maximum average similarity to other objects in the group.

## Centroid-based

- The average, or the *center of mass* in the region.
- Given a class $c_i$, where each object $o_j$ being a member is represented as a feature vector $\boldsymbol{x}_j$, we can compute the class centroid $\boldsymbol{\mu}_i$ as

$$\boldsymbol{\mu}_i = \frac{1}{|c_i|} \sum_{\boldsymbol{x}_j \in c_i} \boldsymbol{x}_j$$

# Different ways of representing classes (cont'd)

## Some more notes on centroids, medoids and typicality

- Both *centroids* and *medoids* represent a group by a single prototype.
- But while a *medoid* is an actual member of the group, a *centroid* is an *abstract* prototype; an average.
- *Typicality* can be defined by a member's distance to the prototype.
- The centroid could also be distance weighted:
  Let each member's contribution to the average be determined by its average pairwise similarity to the other members of the group.
- There are parallel discussions on how to represent classes and determine typicality within linguistic and psychological prototype theory.

# Rocchio classification

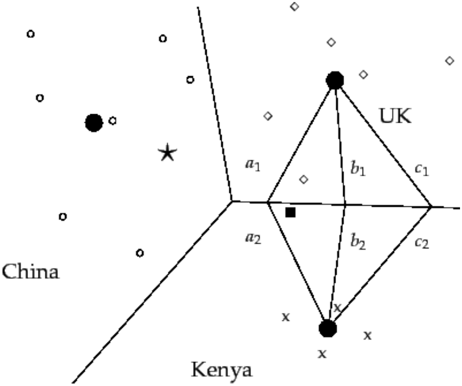
- AKA nearest centroid classifier or nearest prototype classifier.
- Uses centroids to represent classes.
- Each class $c_i$ is represented by its centroid $\boldsymbol{\mu}_i$, computed as the average of the vectors $\boldsymbol{x}_j$ of its members;

$$\boldsymbol{\mu}_i = \frac{1}{|c_i|} \sum_{\boldsymbol{x}_j \in c_i} \boldsymbol{x}_j$$

- The Rocchio decision rule:
- To classify a new object $o_j$ (represented by a feature vector $\boldsymbol{x_j}$);
  - determine which centroid $\boldsymbol{\mu}_i$ that $\boldsymbol{x_j}$ is closest to,
  - and assign it to the corresponding class $c_i$.

# The decision boundary of the Rocchio classifier

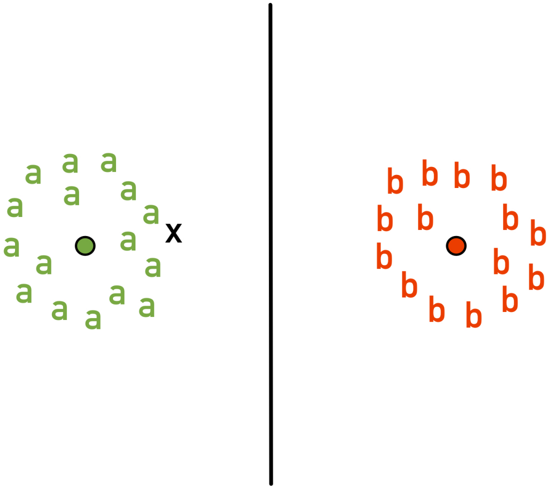
- Defines the boundary between two classes by the set of points equidistant from the centroids.
- In two dimensions, this set of points corresponds to a *line*.
- In multiple dimensions: A line in 2D corresponds to a *hyperplane* in a higher-dimensional space.
- The boundaries are not explicitly computed; implied by the decision rule.



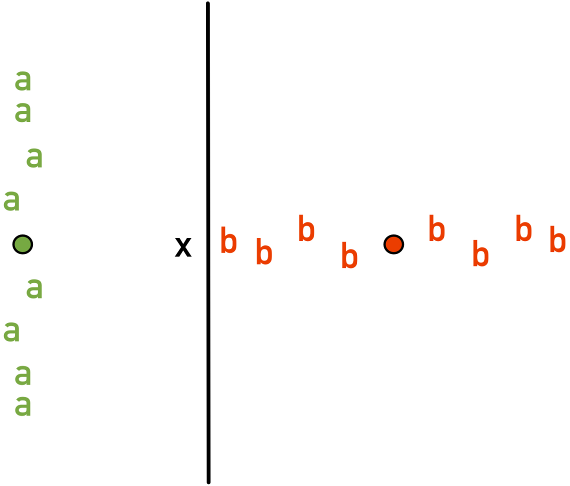
# Problems with the Rocchio classifier

- The classification decision ignores the distribution of members locally within a class, only based on the centroid distance.
- Implicitly assumes that classes are spheres with similar radii.
- Does not work well for classes than cannot be accurately represented by a single prototype or center (e.g. disconnected or elongated regions).
- Because the Rocchio classifier defines a linear decision boundary, it is only suitable for problems involving *linearly separable* classes.

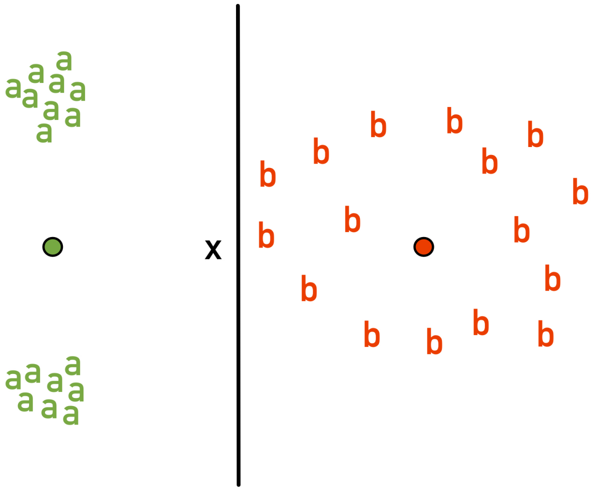
# Ideal



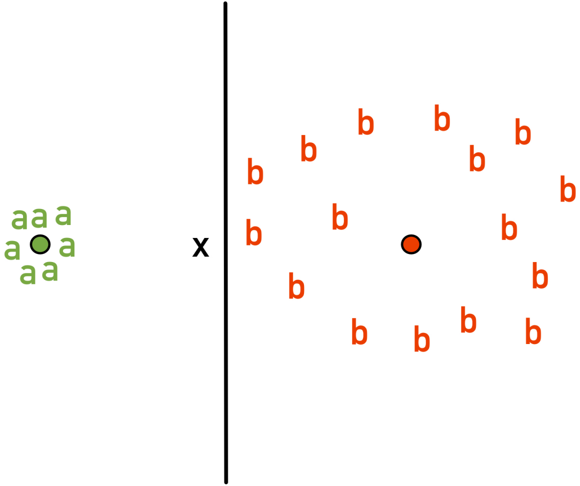
# Problematic: Elongated regions



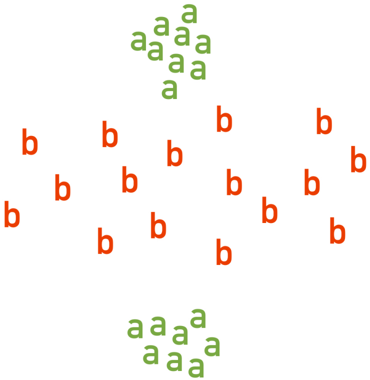
# Problematic: Non-contiguous regions



# Problematic: Different sizes



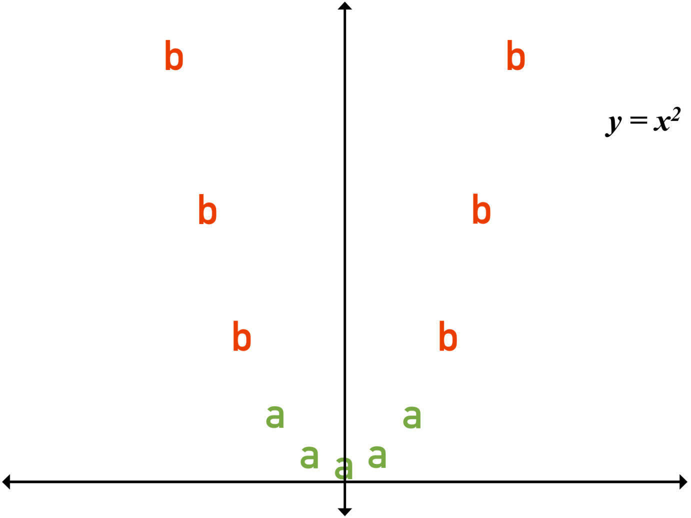
# Problematic: Nonlinear boundary



# A side-note on non-linearity


- Before we turn to talk about non-linear classifiers, note that: Classes that are not linearly separable in a given feature space. . .

# A side-note on non-linearity




- . . . may become linearly separable when the features are mapped to a higher-dimensional space (this is the basis for so-called kernel methods).

# $k$NN-classification

- $k$ Nearest Neighbor classification.
- An example of a memory-based, non-linear classifier.

## Decision rule

- For $k = 1$: Assign each object to the class of its closest neighbor.
- For $k > 1$: Assign each object to the majority class among its $k$ closest neighbors.

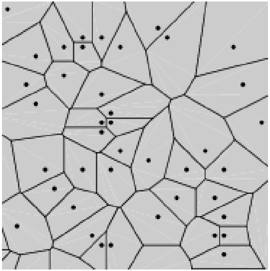
- The parameter $k$ must be specified in advance.
- Rationale: given *the contiguity hypothesis*, we expect a test object $o_i$ to have the same label as the training objects in the local region of $\boldsymbol{x_i}$.
- Unlike Rocchio, the $k$NN decision boundary is determined locally.
  - The decision boundary defined by the Voronoi tessellation.

# Voronoi tessellation

- Assuming $k = 1$: For a given set of objects in the space, let each object define a cell consisting of all points that are closer to that object than to other objects.

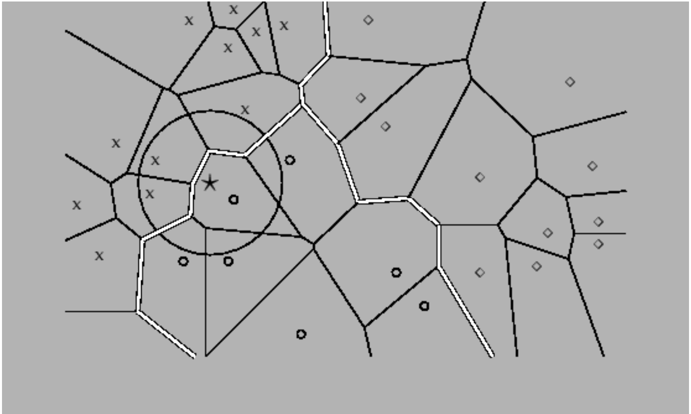
- Results in a set of convex polygons; so-called Voronoi cells.

- Decomposing a space into such cells gives us the so-called Voronoi tessellation.



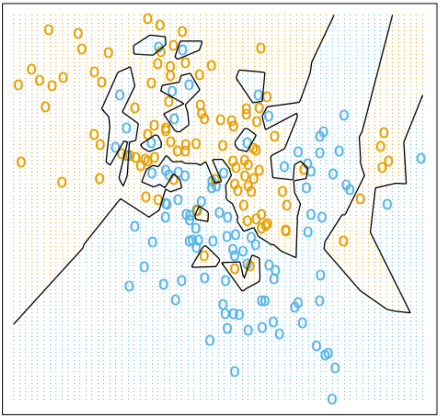
- In the general case of $k \geq 1$, the Voronoi cells are given by the regions in the space for which the set of $k$ nearest neighbors is the same.

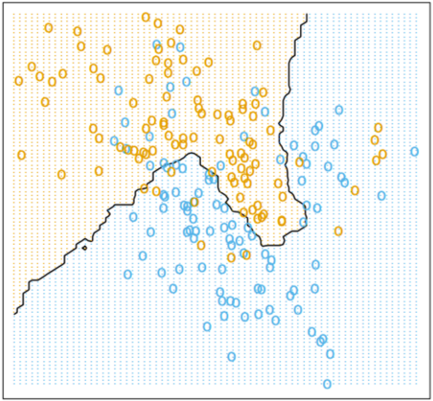
# Voronoi tessellation for 1NN



▶ **Figure 14.3** Voronoi tessellation and decision boundaries (double lines) in 1NN classification. The three classes are: X, circle and diamond.

Decision boundary for 1NN: defined along the regions of Voronoi cells for the objects in each class. Shows the non-linearity of $k$NN.



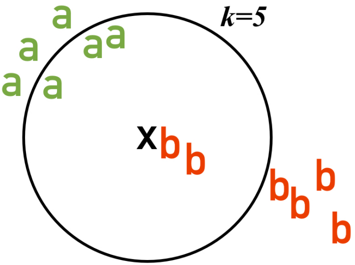

*Figures from Elements of Statistical Learning*

- What would happen if $K = N$?

# ‘Softened’ $k$NN-classification

## A probabilistic version

- The probability of membership in a class $c$ given by the proportion of the $k$ nearest neighbors in $c$.




## Distance weighted votes

- The score for a given class $c_i$ can be computed as

$$\mathrm{score}(c_i, o_j) = \sum_{\boldsymbol{x_n} \in \mathrm{knn}(\boldsymbol{x}_j)} \mathrm{I}(c_i, \boldsymbol{x}_n)\, \mathrm{sim}(\boldsymbol{x_n}, \boldsymbol{x_j})$$

  where $\mathrm{knn}(\boldsymbol{x}_j)$ is the set of $k$ nearest neighbors of $\boldsymbol{x}_j$, $\mathrm{sim}$ is the similarity measure, and $\mathrm{I}(c_i, \boldsymbol{x}_n)$ is 1 if $\boldsymbol{x}_n \in c_i$ and 0 otherwise.

- Can give more accurate results, and also help resolve ties.

# Peculiarities of $k$NN

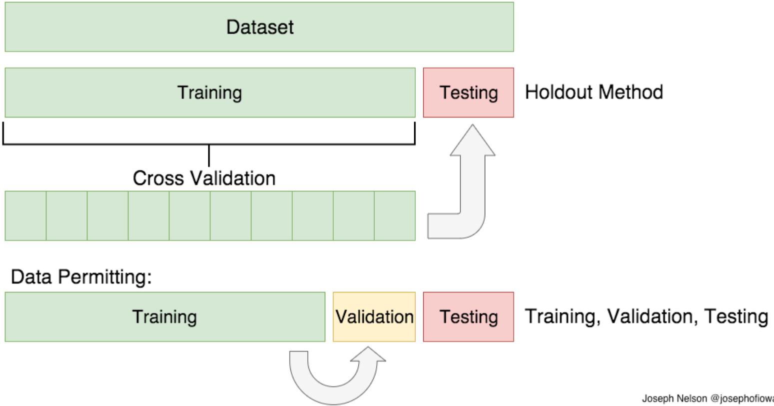
- Not really any *learning* or estimation going on at all;
- simply memorizes all training examples.
- Generally with ML; the more training data the better.
- But for $k$NN, large training sets come with an efficiency penalty.
- Test time is linear in the size of the training set,
- but independent of the number of classes.
- A potential advantage for problems with many classes.
- Notice the similarity to the problem of ad hoc retrieval (e.g., returning relevant documents for a given query);
  - Both are instances of finding nearest neighbors.

# Testing a classifier

- To evaluate a classifier, we measure its number of correct classification predictions on unseen test items.
- Labeled test data is sometimes referred to as the gold standard.
- We evaluate by comparing the predictions made by the model towards the gold labels.
- We will consider different evaluation metrics,
- and the different data splits: Training, development, and test sets.
- (Why can't we test on the training data?)

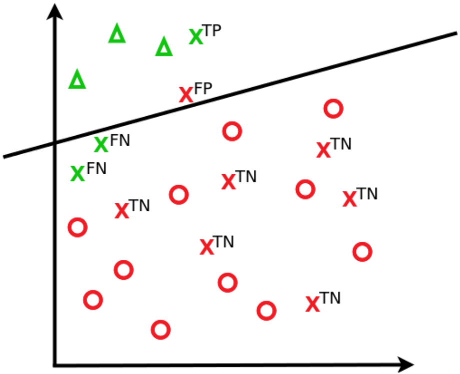
# Using data splits

- While tuning our model, estimated from the training set, we repeatedly evaluate towards the development or validation data.
- Or, if we have little data, by $n$-fold cross-validation.
- Then we want to evaluate how well our *final* model *generalizes* on a held-out test set.




Joseph Nelson @josephofiowa

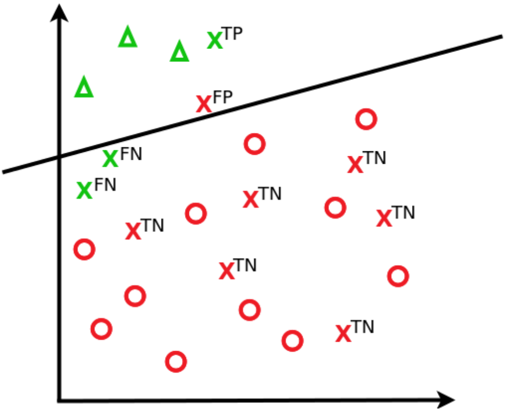
# Example: Evaluating classifier decisions




- Predictions for a given class can be wrong or correct in two ways:

| | gold = positive | gold = negative |
|---|---|---|
| prediction = positive | true positive (TP) | false positive (FP) |
| prediction = negative | false negative (FN) | true negative (TN) |

# Example: Evaluating classifier decisions




Accuracy $= \frac{TP+TN}{N}$
$= \frac{1+6}{10} = 0.7$

Precision $= \frac{TP}{TP+FP}$
$= \frac{1}{1+1} = 0.5$

Recall $= \frac{TP}{TP+FN}$
$= \frac{1}{1+2} = 0.33$

F-score
$= 2 \times \frac{precision \times recall}{precision + recall} = 0.4$

# Evaluation measures

- Accuracy $= \frac{TP+TN}{N} = \frac{TP+TN}{TP+TN+FP+FN}$
  - The ratio of correct predictions.
  - Not suitable for unbalanced numbers of positive / negative examples.
- Precision $= \frac{TP}{TP+FP}$
  - The number of detected class members that were correct.
- Recall $= \frac{TP}{TP+FN}$
  - The number of actual class members that were detected.
  - Trade-off: Positive predictions for all examples would give 100% recall but (typically) terrible precision.
- F-score $= 2 \times \frac{precision \times recall}{precision + recall}$
  - Balanced measure of precision and recall (harmonic mean).

# Evaluating multi-class predictions

## Macro-averaging

- Sum precision and recall for each class, and then compute global averages of these.
- The macro average will be highly influenced by the small classes.

## Micro-averaging

- Sum TPs, FPs, and FNs for all points/objects across all classes, and then compute global precision and recall.
- The micro average will be highly influenced by the large classes.

# Next lecture

- Unsupervised machine learning for class discovery: Clustering
- Flat vs. hierarchical clustering.
- C-Means Clustering.
- Reading: *Manning, Raghavan & Schütze (2008)*, section 16, 16.1, 16.2, and 16.4 up until 16.4.1.