

IN2110: Språkteknologiske metoder

Klassifikasjon

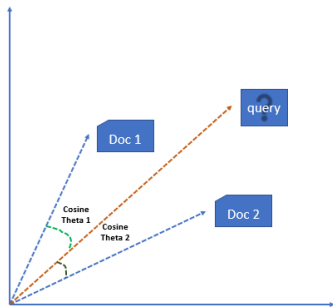
Egil Rønningstad

Språkteknologigruppen (LTG)

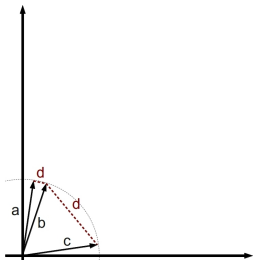
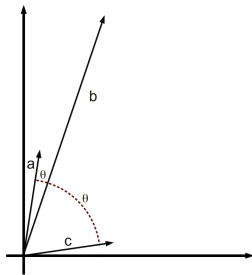
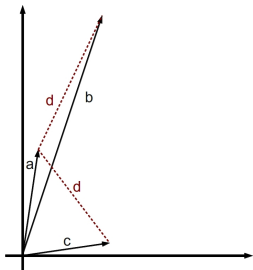
30. januar, 2024



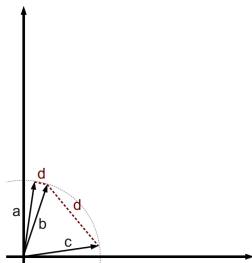
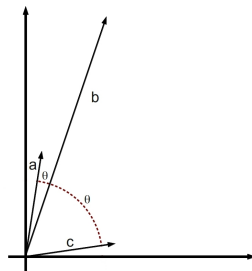
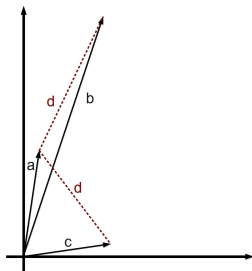
- ▶ General approach to representing data in a geometrical model.
- ▶ Example use case: vector spaces for IR.
- ▶ Documents represented as points/vectors in a **feature space**, $\mathbf{x} = \langle x_1, \dots, x_n \rangle \in \mathbb{R}^n$.
- ▶ **Bag-of-words**:
 - ▶ **Documents** represented by their 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



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**.

Classification

- ▶ **Supervised** learning, requiring **labeled** training data.
- ▶ Train a classifier to automatically assign *new* instances to *predefined* classes, given some set of training examples.
- ▶ (Topic for this week.)



Classification

- ▶ **Supervised** learning, requiring **labeled** training data.
- ▶ Train a classifier to automatically assign *new* instances to *predefined* classes, given some set of training examples.
- ▶ (Topic for this week.)

Clustering

- ▶ **Unsupervised** learning from **unlabeled** data.
- ▶ Automatically group similar objects together.
- ▶ No predefined classes or structure, we only specify the similarity measure.
- ▶ (The topic for next week.)

- ▶ Vector space representations well-suited for introducing classification:
- ▶ Very little math required – easy to understand on the basis of geometrical intuitions.
- ▶ We will consider some very simple but powerful methods.
- ▶ Relevant for assignment **1a**.
- ▶ Example task: text classification.

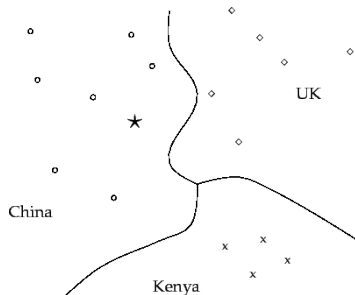
- ▶ Vector space representations well-suited for introducing classification:
- ▶ Very little math required – easy to understand on the basis of geometrical intuitions.
- ▶ We will consider some very simple but powerful methods.
- ▶ Relevant for assignment **1a**.
- ▶ Example task: text classification.

What we cover next:

- ▶ **Rocchio** classification
- ▶ **k-Nearest Neighbors** classification (k NN)
- ▶ Evaluation (next week)

- ▶ Topic classification of news articles
- ▶ Authorship attribution
- ▶ Spam detection
- ▶ Polarity classification (sentiment analysis)
- ▶ Language identification
- ▶ Abusive language detection
- ▶ Intent classification (chatbots)
- ▶ Content recommendation
- ▶ Psychometrics
- ▶ ...

- ▶ 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**.



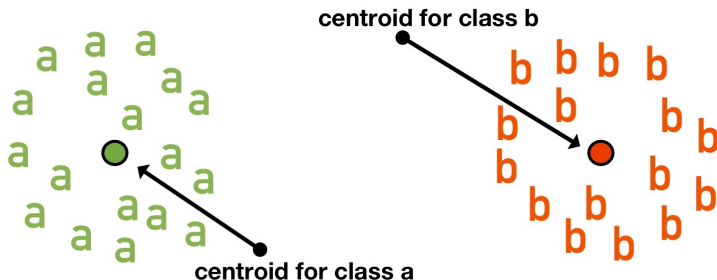


Rocchio classification (1:2)



- ▶ Each class C_i is represented by its **centroid**,
- ▶ or 'center of gravity'.
- ▶ Computed as the average of the vectors x_j of its members;

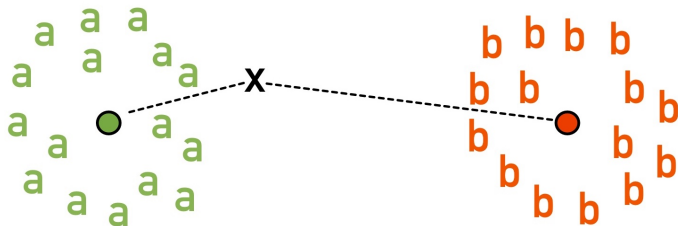
$$\mu_i = \frac{1}{|C_i|} \sum_{x_j \in C_i} x_j$$



- ▶ General formula: $\mu_i = \frac{1}{|C_i|} \sum_{\mathbf{x}_j \in C_i} \mathbf{x}_j$
- ▶ Given a class label '*music*' with 3 training documents represented by n -dimensional vectors $C_{music} = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3\}$:
 $\mathbf{x}_1 = \langle x_{11}, x_{12}, \dots, x_{1n} \rangle$
 $\mathbf{x}_2 = \langle x_{21}, x_{22}, \dots, x_{2n} \rangle$
 $\mathbf{x}_3 = \langle x_{31}, x_{32}, \dots, x_{3n} \rangle$
- ▶ The centroid would be computed as:

$$\mu_{music} = \left\langle \frac{x_{11}+x_{21}+x_{31}}{3}, \frac{x_{12}+x_{22}+x_{32}}{3}, \dots, \frac{x_{1n}+x_{2n}+x_{3n}}{3} \right\rangle$$

- ▶ The Rocchio **decision rule**:
- ▶ To classify a new object o_j (represented by a feature vector x_j);
 - determine which centroid μ_i that x_j is closest to,
 - and assign it the corresponding class label i .

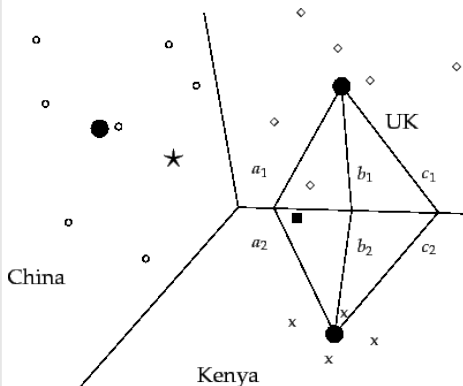


- ▶ AKA **nearest centroid classifier** or nearest prototype classifier.

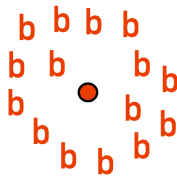
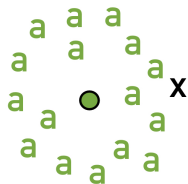
The decision boundary of the Rocchio classifier

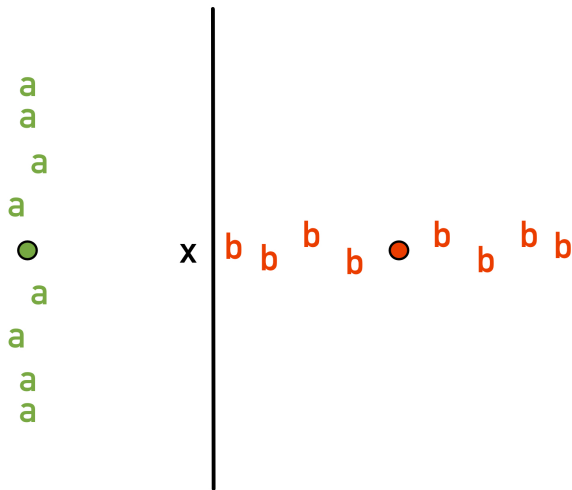


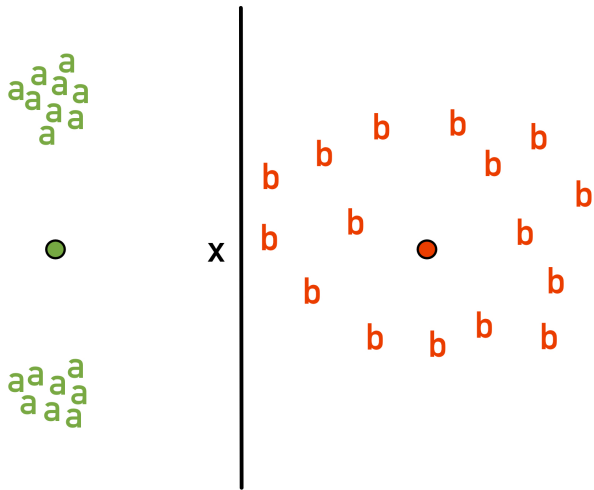
- ▶ Defines the boundary between two classes by the **set of points equidistant from the centroids**.
- ▶ In two dimensions, this set of points corresponds to a **line**.
- ▶ Corresponds to a plane in 3D or a **hyperplane** in higher dimensions.
- ▶ Boundaries *implied* by the decision rule; not explicitly computed.

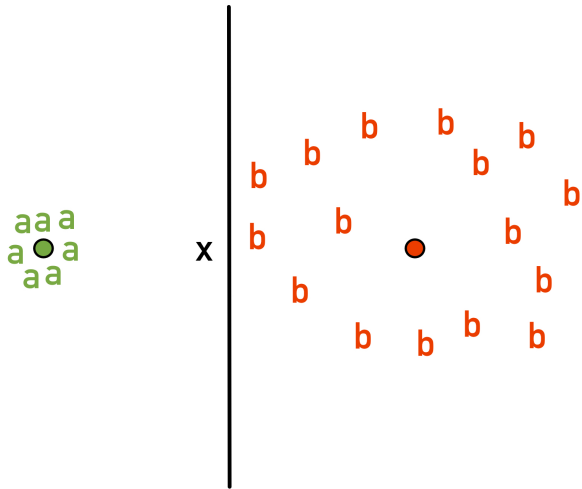


- ▶ The classification decision **ignores the distribution of members** locally within a class, only based on the centroid distance.
- ▶ Implicitly assumes that classes are **spheres** with **similar radii**.
- ▶ Does not work well for classes that cannot be accurately represented by a single prototype 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.

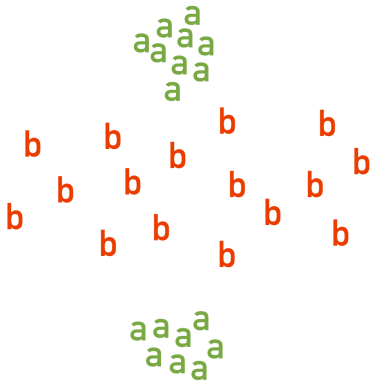






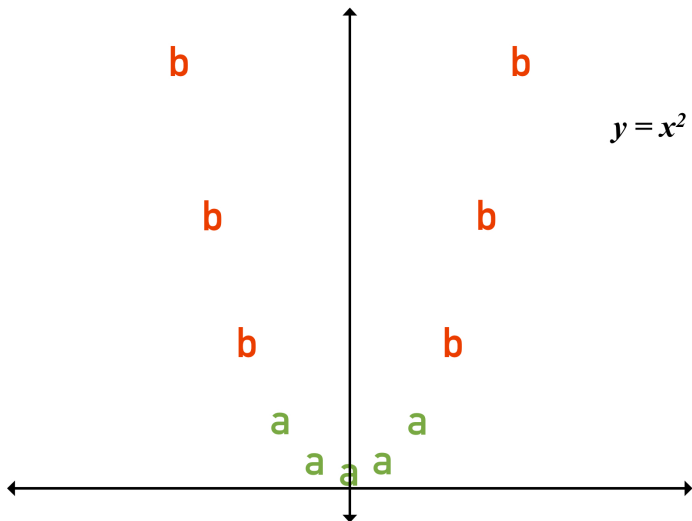


Problematic: Nonlinear boundary





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



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

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

Decision rule

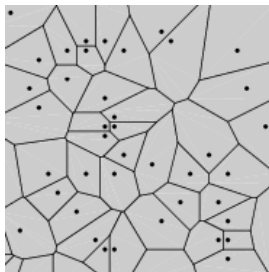
- ▶ For $k = 1$: Assign each object to the class of its closest neighbor.
- ▶ For $k > 1$: Assign each object to the majority class among its k closest neighbors.
- ▶ The parameter k must be specified in advance.

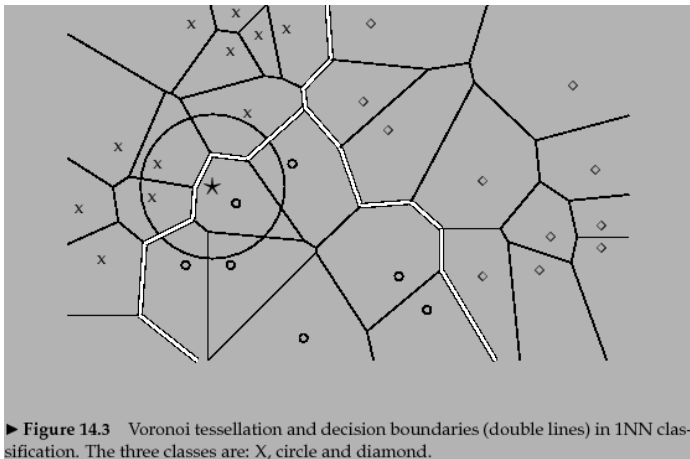
- ▶ k Nearest Neighbor classification.
- ▶ An example of a memory-based, non-linear classifier.

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 x_i .
- ▶ Unlike Rocchio, the k NN decision boundary is determined locally.
 - ▶ The decision boundary defined by the 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.



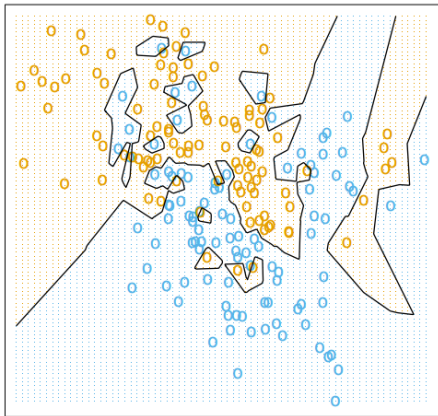


Decision boundary for 1NN: defined along the regions of Voronoi cells for the objects in each class. Shows the **non-linearity** of k NN.

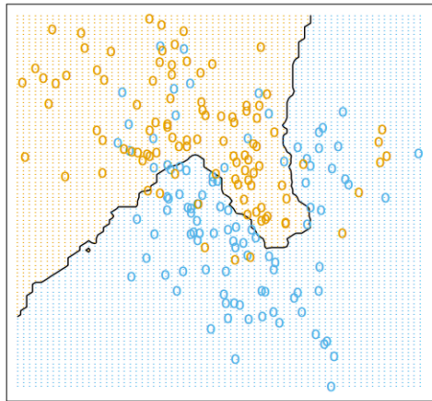
The effect of K



1-Nearest Neighbor Classifier



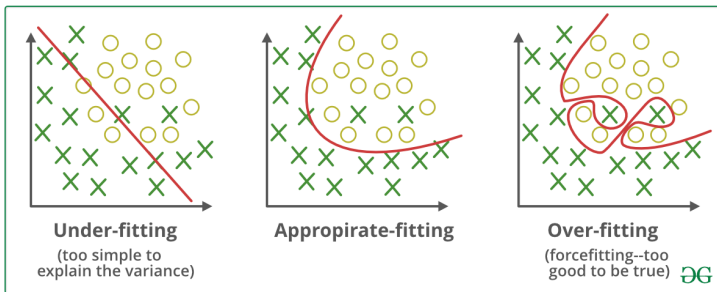
15-Nearest Neighbor Classifier



Figures from Elements of Statistical Learning

- What would happen if $K = N$?

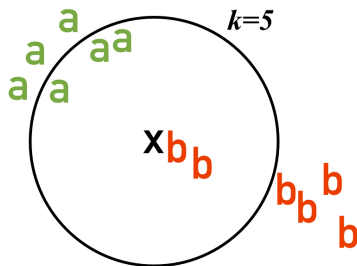
- ▶ **Overfitting:** the model fits the training data closely but generalizes poorly, i.e. has many test errors.
- ▶ **Underfitting:** the model makes many errors even for the training data, i.e. it is unable to describe the data.



- ▶ The **bias–variance tradeoff**; the balancing of two sources of errors:
- ▶ **Bias**: the impact of assumptions in the model. Oversimplified models have high bias, leading to **underfitting**.
- ▶ **Variance**: sensitivity to changes in training data. Highly complex models may have high variance leading to **overfitting**.
- ▶ Example for kNN:
 - ▶ $k=1$: very high variance, very low bias.
 - ▶ $k=N$: very low variance, very high bias.

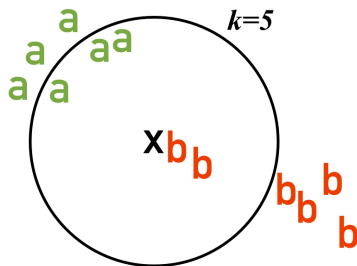
A probabilistic version

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



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

$$\text{score}(c_i, o_j) = \sum_{\mathbf{x}_n \in \text{knn}(\mathbf{x}_j)} \mathbf{I}(c_i, \mathbf{x}_n) \text{sim}(\mathbf{x}_n, \mathbf{x}_j)$$

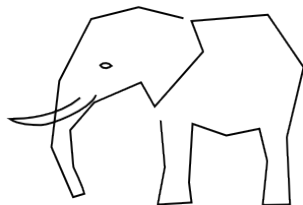
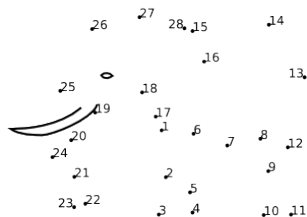
where $\text{knn}(\mathbf{x}_j)$ is the set of k nearest neighbors of $\mathbf{x}_j$, sim is the similarity measure, and $\mathbf{I}(c_i, \mathbf{x}_n)$ is 1 if $\mathbf{x}_n \in c_i$ and 0 otherwise.

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



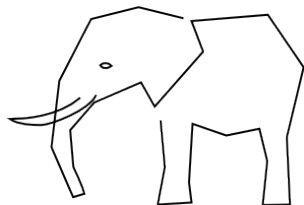
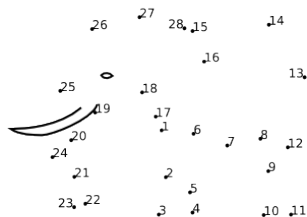
- ▶ Not really any *learning* or estimation going on at all;
- ▶ simply **memorizes** all training examples.
- ▶ Generally with ML; the more training data the better.
- ▶ But for k NN, large training sets come with an efficiency penalty.
- ▶ **Test time** is linear in the size of the training set,
- ▶ 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.

Connecting the dots...



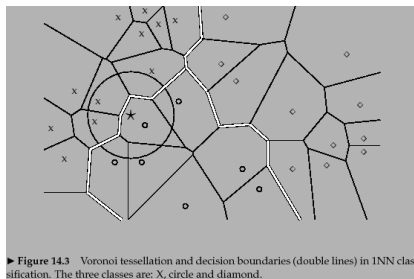
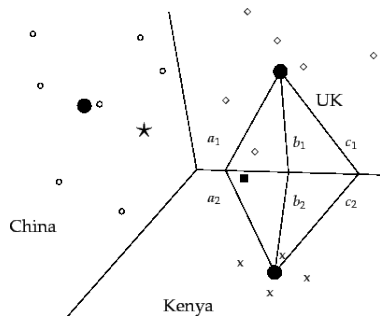
- **Rocchio** can be thought of as a **1 Nearest Neighbor** classifier with respect to the centroids.

Connecting the dots...



- Rocchio can be thought of as a 1 Nearest Neighbor classifier with respect to the centroids.
- How can this be? Isn't k NN non-linear and Rocchio linear?

- Recall that the k NN decision boundary is locally linear for each cell in the Voronoi diagram.
- For **Rocchio**, we only use one example per class, i.e. the centroid, yielding a single Voronoi cell for each entire class.





- ▶ Before training and applying a classifier we first have to create the **feature vectors** to represent our data.
- ▶ Sometimes referred to as **vectorization**.
- ▶ The **feature types** (e.g. the BoW vocabulary) needs to be defined relative to the training set (including parameters like frequency cut-offs, idf-weights, etc).
- ▶ When **vectorizing test data** we must use the same features as in training.
- ▶ Vectorization therefore often done in **two passes**: first defining the feature set based on the training data, then creating the feature vectors. (*Fit* and *transform* in scikit-learn terminology)



- ▶ **Evaluating** classifiers
- ▶ Unsupervised machine learning for class discovery: **Clustering**
- ▶ k -means clustering.
- ▶ Reading: *Manning, Raghavan & Schütze (2008)*, section 16, 16.1, 16.2, and 16.4 up until 16.4.1.