INF4820: Algorithms for AI and NLP

Clustering

Milen Kouylekov & Stephan Oepen

Language Technology Group
University of Oslo

Oct. 1, 2014

Last week

- Supervised vs unsupervised learning.
- Vectors space classification.
- How to represent classes and class membership.
- Rocchio + kNN.
- Linear vs non-linear decision boundaries.

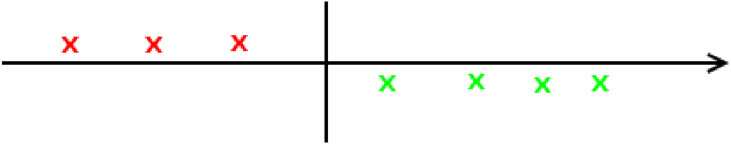
Today

- Refresh
  - Vector Space
  - Clasifiers
  - Evaluation
- Unsupervised machine learning for class discovery: Clustering
- Flat vs. hierarchical clustering.
- k-Means Clustering

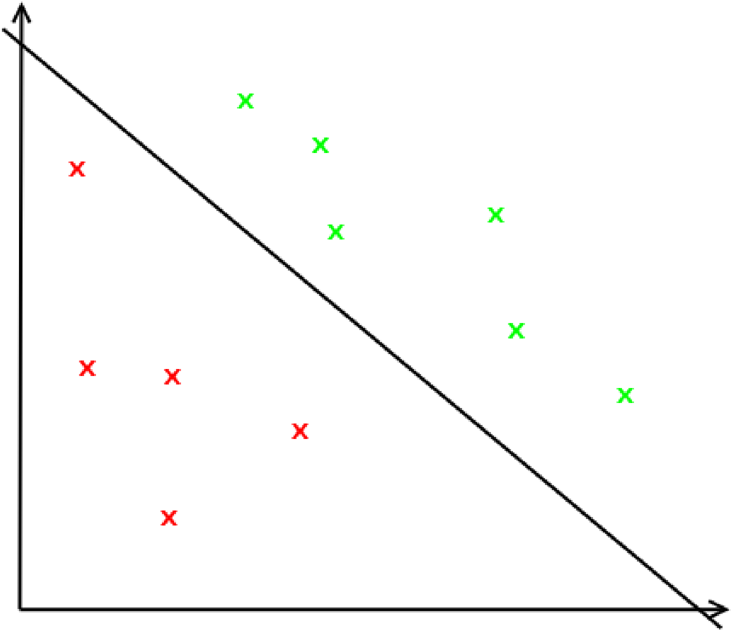
Vector Space Model and Classification

- Describe objects as set of features that describe them.
- Objects are represented as points in space
- Each dimension of the space corresponds feature
- We calculate the their similarity by measuring the distance between them in the space.
- We classify an object by:
  - Creating a plane in the space that separates them (Rocchio Classifier)
  - Proximity of other objects of the same class (KNN Classifier)

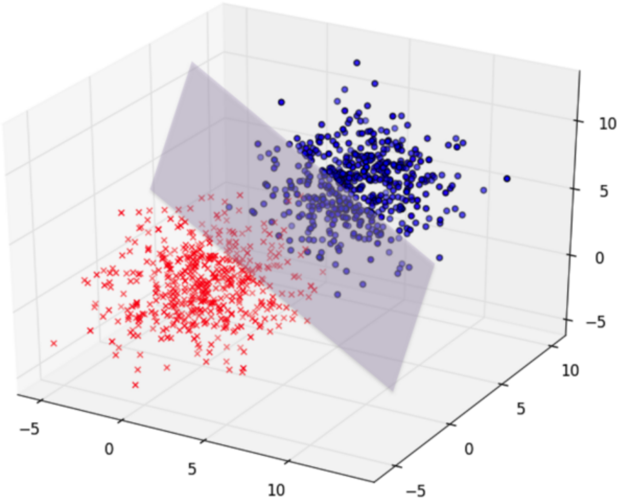
Space (1)



Space (2)



Space (3)



Vector vs Point vs Feature Vector

- Point - coordinates in each dimensions
- Vector - coordinates of 2 points (start and end)
- Feature Vector - The start is 0 on each dimension and the end is the point defined by the values of the features.

Rocchio classification

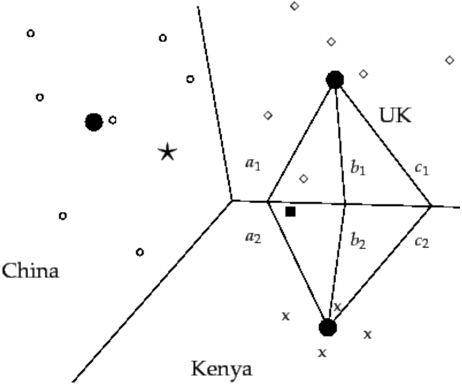
- Uses centroids to represent classes.
- Each class c_i is represented by its centroid $\vec{\mu}_i$, computed as the average of the normalized vectors $\vec{x}_j$ of its members;

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

- To classify a new object o_j (represented by a feature vector $\vec{x}_j$);
  - – determine which centroid $\vec{\mu}_i$ that $\vec{x}_j$ is closest to,
  - – and assign it to the corresponding class c_i.
- The centroids define the boundaries of the class regions.

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.



kNN-classification

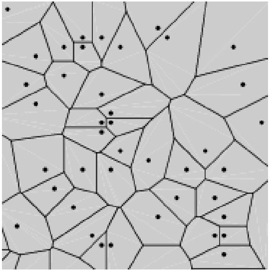
- k Nearest Neighbor classification.
- For $k = 1$: Assign each object to the class of its closest neighbor.
- For $k > 1$: Assign each object to the majority class among its k closest neighbors.
- Rationale: given *the contiguity hypothesis*, we expect a test object o_i to have the same label as the training objects located in the local region surrounding $\vec{x_i}$.
- The parameter k must be specified in advance, either manually or by optimizing on held-out data.
- An example of a non-linear classifier.
- Unlike Rocchio, the kNN 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.

Text Classification

- Task: Classify texts in two domains: financial and political
- Features - count words in the texts:
  - **Feature1**: bank
  - **Feature2**: minster
  - **Feature3**: president
  - **Feature4**: exchange
- Examples:
  - I work for the bank [1,0,0,0]
  - The president met with the minister [0,1,1,0]
  - The minister went in vacation [0,1,0,0]
  - The stock exchange rise after bank news [1,0,0,1]

Sentiment Analysis

- Task: Classify texts in two classes positive or negative.
- Features - presense of words in the texts:
  - **Feature1**: good
  - **Feature2**: bad
  - **Feature3**: excellent
  - **Feature4**: awful
- Examples from movie review dataset:
  - This was good movie [1,0,0,0]
  - Excellent actors in Matrix [0,0,1,0]
  - Excellent actors in good movie [1,0,1,0]
  - Awful film to watch [0,0,0,1]

- Task: Classify Entities in categories. For example: Person - names of people, Location - names of cities, countries etc. and Organization - names of companies,institution etc.
- Features - words that interact with the entities:
  - **Feature1**: invade
  - **Feature2**: elect
  - **Feature3**: bankrupt
  - **Feature4**: buy
- Examples:
  - **Yahoo** bought Overture. - *"Yahoo"* - [0,0,0,1]
  - The barbarians invaded **Rome** - *"Rome"* - [1,0,0,0]
  - **John** went bankrupt after he was not elected - *"John"* - [0,1,1,0]
  - The **Unicredit** bank went bankrupt after it bought NEK - *"Unicredit"* [0,0,1,1]

Textual Entailment

- Task: Recognize a relation that holds between two texts we call **Text** and **Hypothesis**:
  - Example **Entailment**:
    T: Yahoo bought Overture
    H: Yahoo acquired Overture
  - Example **Contradiction**:
    T: Yahoo bought Overture
    H: Yahoo did not acquired Overture
  - Example **Unknown**:
    T: Yahoo bought Overture
    H: Yahoo talked with Overture about collaboration

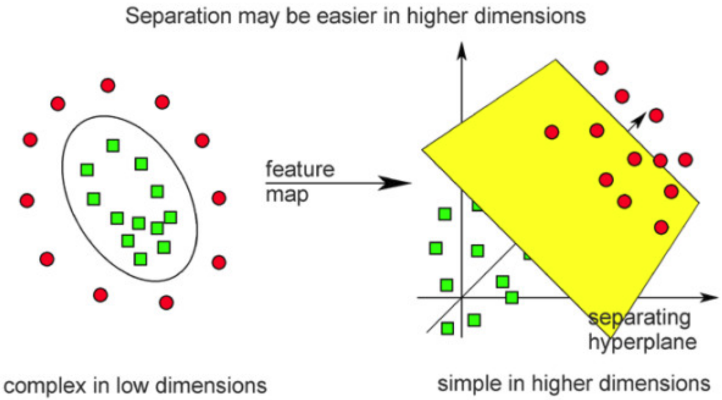
Textual Entailment

- Task: Recognize a relation that holds between two texts we call **Text** and **Hypothesis**:
- Features: -
  - **Feature1**: Word Overlap between T and H
  - **Feature2**: Presence of Negation words (not, never, etc)

Coreference Resolution

- Task: Recognize the referent of a pronoun (it, he she they) from a list of previously recognized names of people.
  - Example
    John walked to school. **He** saw a dog.
  - Example
    John met with **Petter**. **He** recieved a book.
  - Example
    John met with **Merry**. **She** recieved a book.
- Features: Sentence Analysis: Gender Subject etc

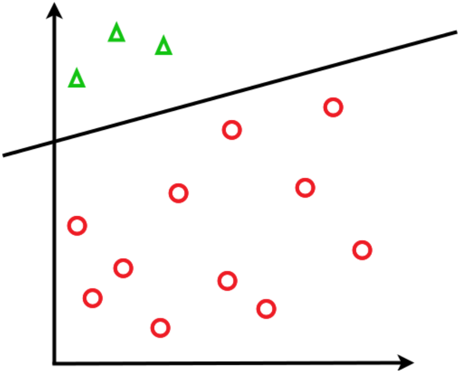
When to add features

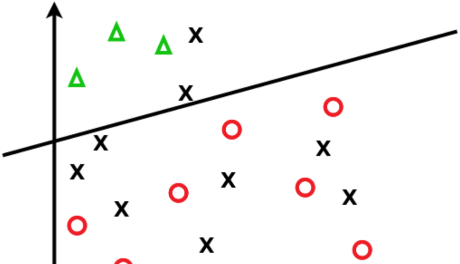


Testing a classifier

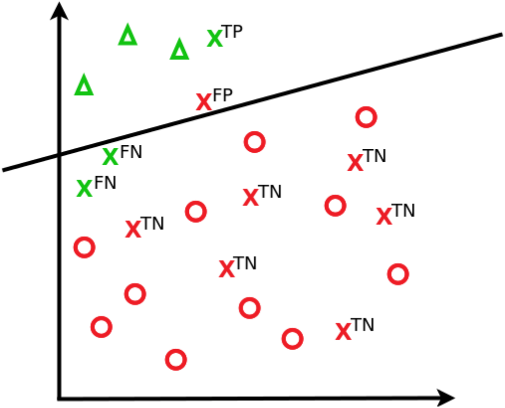
- We've seen how vector space classification amounts to computing the boundaries in the space that separate the class regions; *the decision boundaries*.
- To evaluate the boundary, we measure the number of correct classification predictions on unseeen test items.
  - Many ways to do this...
- We want to test how well a model *generalizes* on a held-out test set.
- (Or, if we have little data, by n-fold cross-validation.)
- Labeled test data is sometimes refered to as the gold standard.
- Why can't we test on the training data?

Example: Evaluating classifier decisions





Example: Evaluating classifier decisions



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

$$\textit{precision} = \frac{TP}{TP+FP} = \frac{1}{1+1} = 0.5$$

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

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

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.
- $F\text{-}score = \frac{2 \times precision \times recall}{precision + recall}$
  - Balanced measure of precision and recall (harmonic mean).

Evaluating multi-class predictions

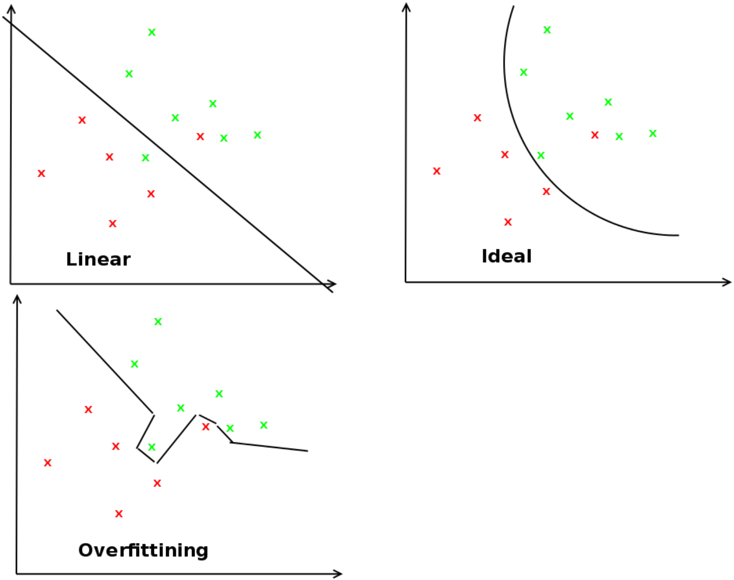
Macro-averaging

- 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.

Over-Fitting



Two categorization tasks in machine learning

Classification

- Supervised learning, requiring labeled training data.
- Given some training set of examples with class labels, train a classifier to predict the class labels of new objects.

Clustering

- Unsupervised learning from unlabeled data.
- Automatically group similar objects together.
- No pre-defined classes: we only specify the similarity measure.
- General objective:
  - Partition the data into subsets, so that the similarity among members of the same group is high (homogeneity) while the similarity between the groups themselves is low (heterogeneity).

Example applications of cluster analysis

- Visualization and exploratory data analysis.
- Many applications within IR. Examples:
  - Speed up search: First retrieve the most relevant cluster, then retrieve documents from within the cluster.
  - Presenting the search results: Instead of ranked lists, organize the results as clusters (see e.g. `clusty.com`).
- Dimensionality reduction / class-based features.
- News aggregation / topic directories.
- Social network analysis; identify sub-communities and user segments.
- Image segmentation, product recommendations, demographic analysis, ...

Types of clustering methods

Different methods can be divided according to the *memberships* they create and the *procedure* by which the clusters are formed:

$$\text{Procedure} \begin{cases} \text{Flat} \\ \text{Hierarchical} \begin{cases} \text{Agglomerative} \\ \text{Divisive} \end{cases} \\ \text{Hybrid} \end{cases}$$

$$\text{Memberships} \begin{cases} \text{Hard} \\ \text{Soft} \\ \text{Disjunctive} \end{cases}$$

Types of clustering methods (cont'd)

Hierarchical

- Creates a tree structure of hierarchically nested clusters.
- Topic of the next lecture.

Flat

- Often referred to as partitional clustering when assuming hard and disjoint clusters. (But can also be soft.)
- Tries to directly decompose the data into a set of clusters.

Flat clustering

- Given a set of objects $O = \{o_1, \ldots, o_n\}$, construct a set of clusters $C = \{c_1, \ldots, c_k\}$, where each object o_i is assigned to a cluster c_i.
- Parameters:
  - The cardinality k (the number of clusters).
  - The similarity function s.
- More formally, we want to define an assignment $\gamma : O \rightarrow C$ that optimizes some objective function $F_s(\gamma)$.
- In general terms, we want to optimize for:
  - High intra-cluster similarity
  - Low inter-cluster similarity

Flat clustering (cont'd)

Optimization problems are search problems:

- There's a finite number of possible partitionings of O.
- Naive solution: enumerate all possible assignments $\Gamma = \{\gamma_1, \ldots, \gamma_m\}$ and choose the best one,

$$\hat{\gamma} = \arg\min_{\gamma \in \Gamma} F_s(\gamma)$$

- Problem: Exponentially many possible partitions.
- Approximate the solution by iteratively improving on an initial (possibly random) partition until some stopping criterion is met.

k-Means

- Unsupervised variant of the Rocchio classifier.
- Goal: Partition the n observed objects into k clusters C so that each point $\vec{x}_j$ belongs to the cluster c_i with the nearest centroid $\vec{\mu}_i$.
- Typically assumes Euclidean distance as the similarity function s.
- The optimization problem: For each cluster, minimize the *within-cluster sum of squares*, $F_s = \text{WCSS}$:

$$\text{WCSS} = \sum_{c_i \in C} \sum_{\vec{x}_j \in c_i} \|\vec{x}_j - \vec{\mu}_i\|^2$$

- Equivalent to minimizing the average squared distance between objects and their cluster centroids (since n is fixed), —a measure of how well each centroid represents the members assigned to the cluster.

k-Means (cont'd)

Algorithm

Initialize: Compute centroids for k seeds.

Iterate:

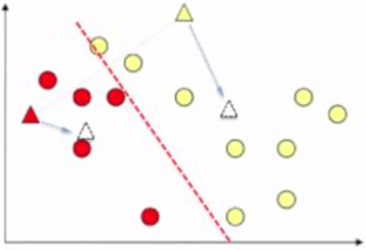
- Assign each object to the cluster with the nearest centroid.
- Compute new centroids for the clusters.

Terminate: When stopping criterion is satisfied.

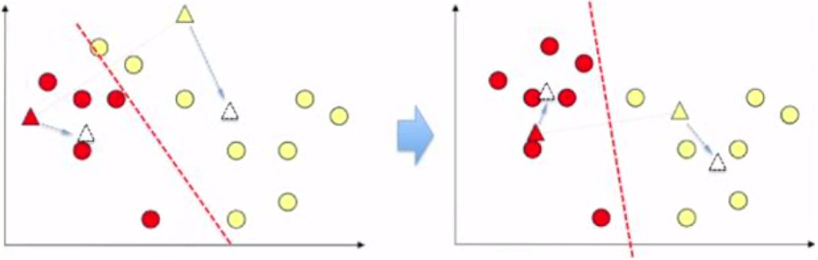
Properties

- In short, we iteratively reassign memberships and recompute centroids until the configuration stabilizes.
- WCSS is monotonically decreasing (or unchanged) for each iteration.
- Guaranteed to converge but not to find the global minimum.
- The time complexity is linear, $\mathrm{O}(kn)$.

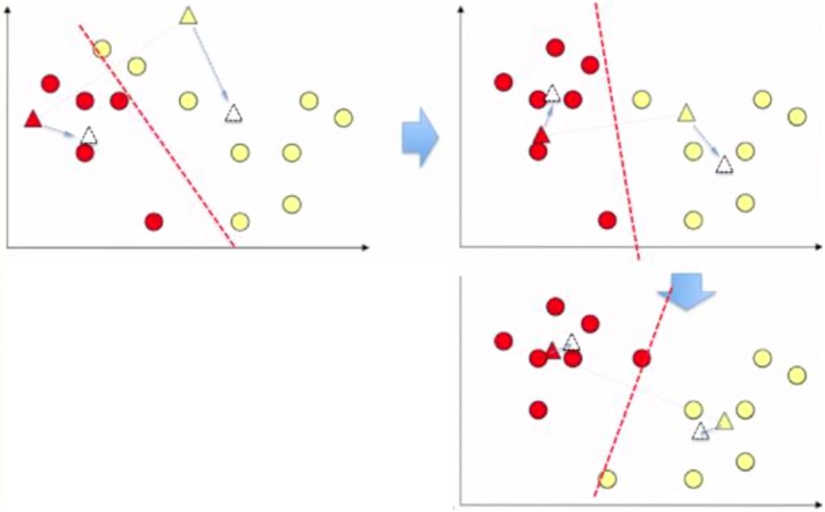
kMeans Example



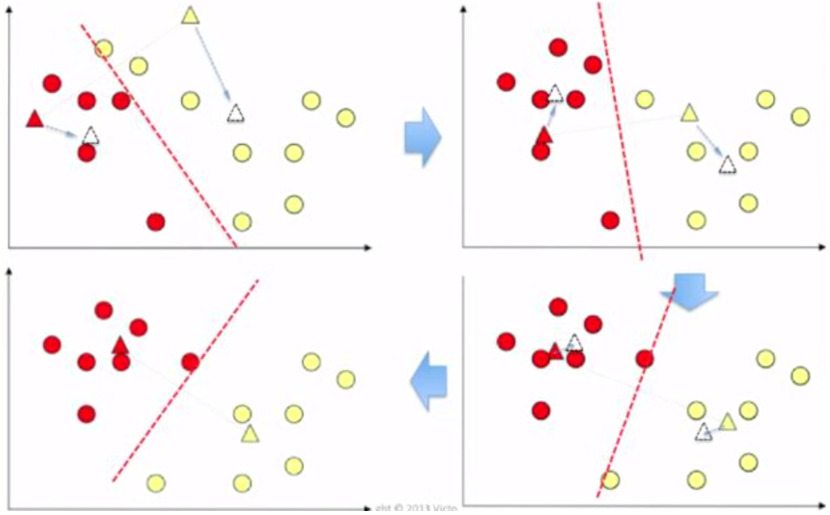
kMeans Example



kMeans Example



kMeans Example



Comments on k-Means

"Seeding"

- We initialize the algorithm by choosing random *seeds* that we use to compute the first set of centroids.
- Many possible heuristics for selecting the seeds:
  - pick k random objects from the collection;
  - pick k random points in the space;
  - pick k sets of m random points and compute centroids for each set;
  - compute an hierarchical clustering on a subset of the data to find k initial clusters; etc..
- The initial seeds can have a large impact on the resulting clustering (because we typically end up only finding a local minimum of the objective function).
- Outliers are troublemakers.

Comments on k-Means

Possible termination criterions

- Fixed number of iterations
- Clusters or centroids are unchanged between iterations.
- Threshold on the decrease of the objective function (absolute or relative to previous iteration)

Some Close Relatives of k-Means

- k-Medoids: Like k-means but uses medoids instead of centroids to represent the cluster centers.
- Fuzzy c-Means (FCM): Like k-means but assigns soft memberships in $[0, 1]$, where membership is a function of the centroid distance.
  - The computations of both WCSS and centroids are weighted by the membership function.

Flat Clustering: The good and the bad

Pros

- Conceptually simple, and easy to implement.
- Efficient. Typically linear in the number of objects.

Cons

- The dependence on the random seeds makes the clustering non-deterministic.
- The number of clusters k must be pre-specified. Often no principled means of *a priori* specifying k.
- The clustering quality often considered inferior to that of the less efficient hierarchical methods.
- Not as informative as the more stuctured clusterings produced by hierarchical methods.

Next

- Hierarchical clustering:
- Agglomerative clustering
  - Bottom-up hierarchical clustering
- Divisive clustering
  - Top-down hierarchical clustering
- How to measure the inter-cluster similarity ("linkage criterions").