—INF4820—

Classification
Clustering

Erik Velldal

University of Oslo

Nov. 10, 2009

Topics for Today

- Quick recap from last lecture:
  - Neighbor relations; kNN and RNN
  - Ways to represent classes (exemplar-based vs. centroid-based).
  - Ways to represent class membership (hard vs. soft).
  - The classification problem in vector space models.
- More on Rocchio classifiers and kNN classifiers
- Linear vs. non-linear classifiers
- Voronoi Tessellations
- Unsupervised machine learning for class discovery: Clustering
- Flat vs. hierarchical clustering
- k-Means Clustering
- Reading: Chapter 16 in Manning, Raghavan & Schütze (2008), *Introduction to Information Retrieval*; `http://informationretrieval.org/`.

The Classification Task

- The task of automatically assigning objects to pre-defined classes.
- A core problem in machine learning (ML).
- Example of a supervised learning task (the training data contains labeled data, indicating what we want to learn).
- Vector space classification relies on the assumption that objects (i.e. points) in the same class form contiguous and non-overlapping regions in the space. ("The contiguity hypothesis")
- Classification amounts to defining boundaries in the space that separate objects in different classes: *The decision boundaries*.
- The goal is to find boundaries that gives high classification accuracy on unseen test items.

Rocchio Classification

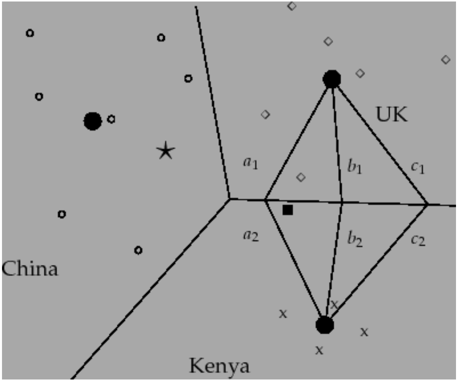
- Uses centroids to represent classes and define the boundaries of the class regions.
- Each class c_i represented by its centroid $\vec{\mu}_i$, computed as the average of the normalized vectors $\vec{x}_j$ of its members;

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

- 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 Decision Boundary of the Rocchio Classifier

- Defines the boundary between two classes by the set of points that are equidistant from the centroids.
- In two dimensions: This set of points always corresponds to a *line*.
- In multiple dimensions: A line in 2D corresponds to a *hyperplane* in a higher-dimensional space.



Problems with the Rocchio Classifier

- Implicitly assumes that classes are *spheres with similar radii*.
- Ignores details of the distribution of points within a class, only based on the centroid distance.
- Does not work well for classes than cannot be accurately represented by a single prototype or "center" (e.g. classes covering disconnected or elongated regions).

Problems with the Rocchio Classifier

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

KNN-classification

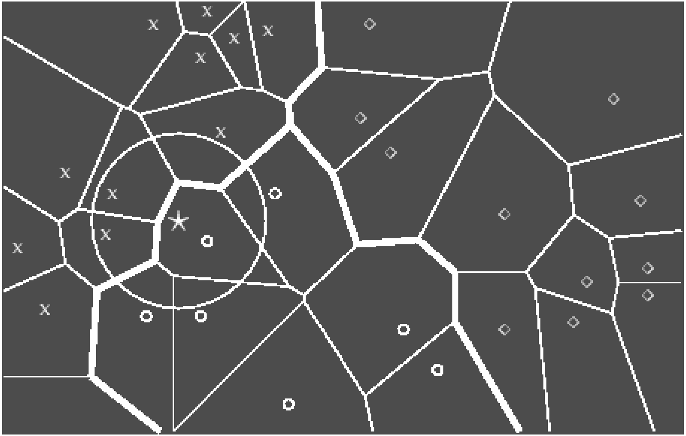
- k Nearest Neighbor classification.
- An example of a non-linear classifier.
- For $k = 1$: Assign each object to the class of its closest neighbor.
- For $k > 1$: Assign each object to the majority class among its k closest neighbors.
- Rationale: given *the contiguity hypothesis*, we expect a test 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 by manually or by optimizing on held-out data.

KNN-classification (cont'd)

The Voronoi Tessellation

- Defines the decision boundaries of the kNN classifier.
- 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.
- Each such *Voronoi cell* will be a convex polygon.
- Decomposing a space into such Voronoi cells gives us the so-called Voronoi tessellation.
- In the general case of $k \geq 1$, the Voronoi cells will be given by the regions in the space for which the set of k nearest neighbors is the same. Partitions the space into convex polygons.

Voronoi Tessellation for 1NN (Manning, Raghavan & Schütze 2008)



The decision boundary for the 1NN classifier is defined along the regions of Voronoi cells for the objects in each class. Shows the non-linearity of kNN.

“Softened” KNN-classification

A Probabilistic Version

- Estimate the probability of membership in class c as the proportion of the k nearest neighbors in c.

"Softened" KNN-classification

A Probabilistic Version

- Estimate the probability of membership in class c as the proportion of the k nearest neighbors in c.

A Distance Weighted Version

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

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

  where $\text{knn}(\vec{x}_j)$ is the set of k nearest neighbors of $\vec{x}_j$, sim is whatever similarity measure we're using, e.g. the cosine function, and $\text{I}(c_i, \vec{x}_n)$ is simply a membership function returning 1 if $\vec{x}_n \in c_i$ and 0 otherwise.
- Such distance weighted votes can often give more accurate results, e.g. in the case of ties.

Two Categorization Tasks in Machine Learning

Classification

- Supervised learning, requiring labeled training data.
- Train a classifier to automatically assign *new* instances to *predefined* classes.

Two Categorization Tasks in Machine Learning

Classification

- Supervised learning, requiring labeled training data.
- Train a classifier to automatically assign *new* instances to *predefined* classes.

Clustering

- Unsupervised learning from unlabeled data.
- Automatically group similar objects together.
- No predefined classes or structure, we only specify the similarity measure. Relies on "self-organization".
- 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 explorative data analysis.

Example Applications of Cluster Analysis

- Visualization and explorative data analysis.
- Generalization and abstraction. "Reason by analogy".
  - Lets us define class-based models even when predefined classes are not available.
  - E.g. using cluster-analysis of words to define class-based language models.
  - Helps alleviating the sparse data problem.

Example Applications of Cluster Analysis

- Visualization and explorative data analysis.
- Generalization and abstraction. "Reason by analogy".
  - Lets us define class-based models even when predefined classes are not available.
  - E.g. using cluster-analysis of words to define class-based language models.
  - Helps alleviating the sparse data problem.
- Many applications within IR. Examples:
  - Speed up search: For a clustered document collection, 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 or Google's *wonder wheel*).

Example Applications of Cluster Analysis

- Visualization and explorative data analysis.
- Generalization and abstraction. "Reason by analogy".
  - Lets us define class-based models even when predefined classes are not available.
  - E.g. using cluster-analysis of words to define class-based language models.
  - Helps alleviating the sparse data problem.
- Many applications within IR. Examples:
  - Speed up search: For a clustered document collection, 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 or Google's *wonder wheel*).
- News aggregation / topic directories.
- Social network analysis; identify sub-communities and user segments.

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
- Divisive (top-down): Let all objects be members of the same cluster; then successively split the group into smaller and maximally dissimilar clusters until all objects is its own singleton cluster.
- Agglomerative (bottom-up): Let each object define its own cluster; then successively merge most similar clusters until only one remains.

Types of Clustering Methods (cont'd)

Hierarchical

- Creates a tree structure of hierarchically nested clusters
- Divisive (top-down): Let all objects be members of the same cluster; then successively split the group into smaller and maximally dissimilar clusters until all objects is its own singleton cluster.
- Agglomerative (bottom-up): Let each object define its own cluster; then successively merge most similar clusters until only one remains.

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 $O = \{o_1, \ldots, o_n\}$, a hard flat clustering algorithm seeks to construct a set of clusters $C = \{c_1, \ldots, c_k\}$, where each object o_i is assigned to a single cluster c_i.
- More formally, we want to define an assignment $\gamma : O \rightarrow C$ that optimizes some objective function $F_s(\gamma)$.
- The cardinality k (= the number of clusters) must typically be manually specified as a parameter to the algorithm.
- But the most important parameter is the similarity function s.
- The objective function is defined in terms of the similarity function, and generally 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 of partitionings of O.
  - Naive solution: enumerate all possible assignments $\Gamma = \{\gamma_1, \ldots, \gamma_m\}$ and choose

$$\hat{\gamma} = \underset{\gamma \in \Gamma}{\arg\min}\, F_s(\gamma)$$

Flat Clustering (cont'd)

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

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

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

Next Week

- More on flat clustering: k-Means
- Different ways of measuring the distance between classes or clusters.
- Flat vs. hierarchical clustering
- Agglomerative vs. divisive hierarchical clustering
- Reading: Chapter 17 in Manning, Raghavan & Schütze (2008), *Introduction to Information Retrieval*; `http://informationretrieval.org/` (see course web-page for the relevant sections).