INF4820: Algorithms for AI and NLP

Classification

Milen Kouylekov & Stephan Oepen

Language Technology Group
University of Oslo

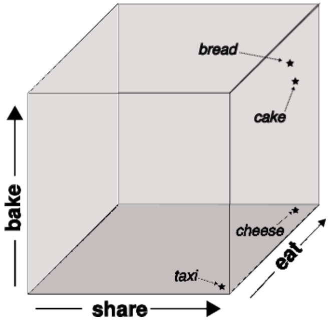
Sep. 18, 2014

Today

- Vector spaces
  - Quick recap
  - Vector space models for Information Retrieval (IR)
- Machine learning: Classification
  - Representing classes and membership
  - Rocchio classifiers
  - kNN classifiers

Summing up

- Semantic spaces: Vector space models for distributional semantics.
- Words are represented as points/vectors in a space, positioned by their co-occurrence counts for various context features.
- For each word, extract context features across a corpus.
- Let each feature type correspond to a dimension in space.
- Each word o_i is represented by a (length-normalized) n-dimensional feature vector $\vec{x}_i = \langle x_{i1}, \ldots, x_{in} \rangle \in \Re^n$.
- We can now measure, say, the Euclidean distance of words in the space, $d(\vec{x}, \vec{y})$.
- Semantic relatedness $\approx$ distributional similarity $\approx$ spatial proximity



An aside: Term–document spaces for IR

- So far we've looked at vector space models for detecting *words* with similar *meanings*.
- It's important to realize that vector space models are widely used for other purposes as well.
- For example, vector space models are commonly used in IR for finding *documents* with similar *content*.
- Each document d_j is represented by a feature vector, with features corresponding to the terms $t_1, \ldots, t_n$ occurring in the documents.
- Spatial distance $\approx$ similarity of content.

An aside: Term–document spaces for IR (cont'd)

- The term–document vectors can also be used for scoring and ranking a document's relevance relative to a given *search query*.
  - Represent the search query as a vector, just like for the documents.
  - The relevance of documents relative to the query can be ranked according to their distance to the query in the feature space.

Classification Example

- Task: Named Entity Recognition
  - Recognize Entities
  - Assign them a class (ex. Person Location and Organization)
- Simplification: Classify upper case words/phrases in classes
- Classify using similarity to examples: London , Paris , Oslo , Clinton ...

Classification Example 2

- Task: Sentiment Analysis
  - Classify Sentences into classes Positive, Negative Neutral
- Vector of features is assigned to entire sentence
- Use example sentences
- Tailored subset of words in context (ex. good, nice awful ..)

Classification Example 3

- Task: Textual Entailment
  - Classify pair of sentences A and B into 2 classes: YES (A implies B) and NO (A does not imply B)
- Vector of features is assigned to the pair
- Use example pairs
- Features: Word Overlap , Longest Common Subsequence, Levenstein Distance

Two categorization tasks in machine learning

Clustering

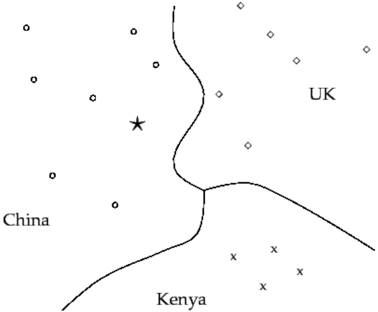
- Unsupervised learning from unlabeled data.
- Automatically group similar objects together.
- No predefined classes or structure, we only specify the similarity measure. Relies on "self-organization".
- Topic of the next lecture(s).

Classification

- Supervised learning, requiring labeled training data.
- Train a classifier to automatically assign *new* instances to *predefined* classes, given some set of examples.
- We'll look at two examples of classifiers that use a vector space representation: Rocchio and kNN.

Classes and classification

- A class can simply be thought of as a collection of objects.
- In our vector space model, objects are represented as *points*, so a class will correspond to a collection of points; a region.
- Vector space classification is based on the 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 $\vec{x}_j$, we can compute the class centroid $\vec{\mu}_i$ as

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

Different ways of representing classes (cont'd)

Some more notes on centroids, medoids and typicality

- *Centroids* and *medoids* both represent a group of objects by a single point, a prototype.
- But while a *medoid* is an actual member of the group, a *centroid* is an *abstract* prototype; an average.
- The *typicality* of class members can be determined by their 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.
- The discussion of how to represent classes in machine learning parallels the discussion of how to represent classes and determine typicality within linguistic and psychological prototype theory.

Representing class membership

Hard Classes

- Membership considered a Boolean property: a given object is either part of the class or it is not.
- A *crisp* membership function.
- A variant: disjunctive classes. Objects can be members of more than one class, but the memberships are still crisp.

Soft Classes

- Class membership is a graded property.
- Probabilistic. The degree of membership for a given restricted to $[0, 1]$, and the sum across classes must be 1.
- Fuzzy: The membership function is still restricted to $[0, 1]$, but without the probabilistic constraint on the sum.

Rocchio classification

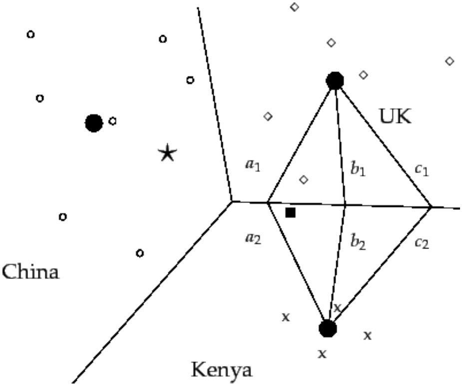
- Uses centroids to represent classes.
- Each class c_i is represented by its centroid $\vec{\mu}_i$, computed as the average of the normalized vectors $\vec{x}_j$ of its members;

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

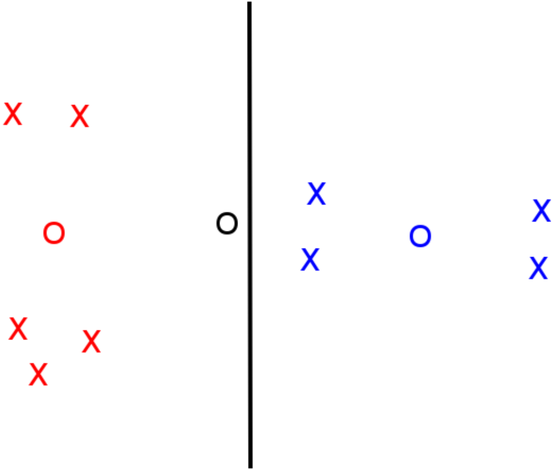
- To classify a new object o_j (represented by a feature vector $\vec{x}_j$);
  – determine which centroid $\vec{\mu}_i$ that $\vec{x}_j$ is closest to,
  – and assign it to the corresponding class c_i.
- The centroids define the boundaries of the class regions.

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.



Problems with the Rocchio classifier



Problems with the Rocchio classifier

- Ignores details of the distribution of points 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.

kNN-classification

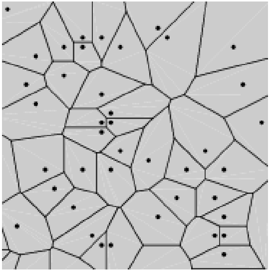
- k Nearest Neighbor classification.
- For $k = 1$: Assign each object to the class of its closest neighbor.
- For $k > 1$: Assign each object to the majority class among its k closest neighbors.
- Rationale: given *the contiguity hypothesis*, we expect a test object o_i to have the same label as the 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.

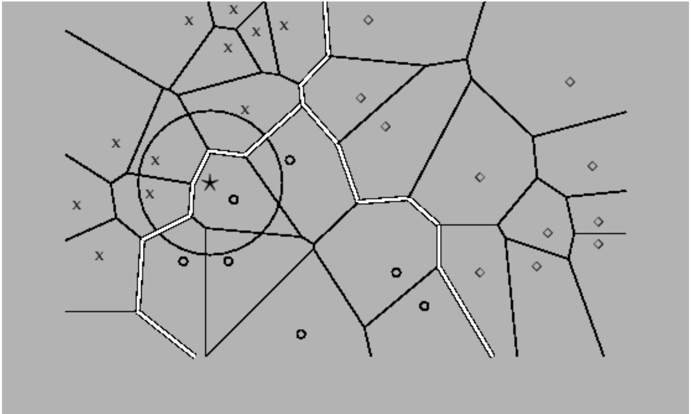
- Results in a set of convex polygons; so-called Voronoi cells.

- 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.

Voronoi tessellation for 1NN



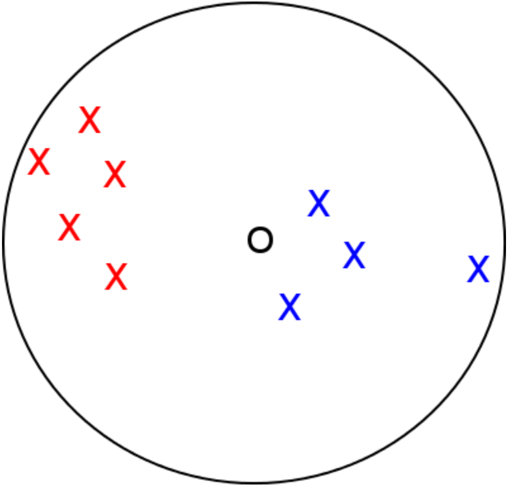
▶ **Figure 14.3** Voronoi tessellation and decision boundaries (double lines) in 1NN classification. The three classes are: X, circle and diamond.

Decision boundary for 1NN: defined along the regions of Voronoi cells for the objects in each class. Shows the non-linearity of 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, 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, and also help resolve ties.

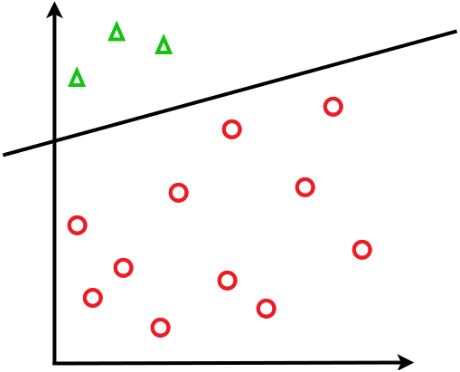
Some peculiarities of kNN

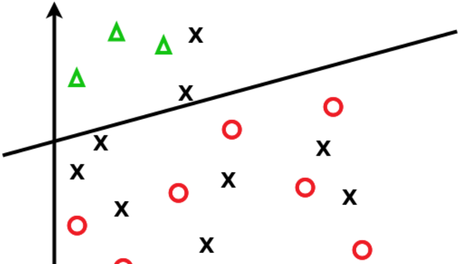
- Not really any *learning* or estimation going on at all;
- simply memorizes all training examples.
- Example of so-called *memory-based learning* or instance-based learning.
- In general in machine learning, the more training data the better.
- But for kNN, large training sets comes with an efficiency penalty in classification.
- 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.
- Test time is linear in the size of the training set,
- and independent of the number of classes.
- A potential advantage for problems with many classes.

Testing a classifier

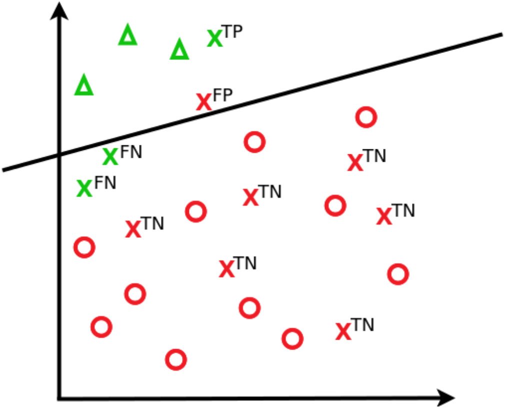
- We've seen how vector space classification amounts to computing the boundaries in the space that separate the class regions; *the decision boundaries*.
- To evaluate the boundary, we measure the number of correct classification predictions on unseeen test items.
  - Many ways to do this…
- We want 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.

Next Lecture

- Unsupervised machine learning for class discovery: Clustering
- Flat vs. hierarchical clustering.
- C-Means Clustering.
- Reading: Chapters 16 and 17 in Manning, Raghavan & Schütze (2008) (see course page for the relevant sections).