— INF4820 — Algorithms for AI and NLP

Evaluating Classifiers
Clustering

Erik Velldal & Stephan Oepen

Language Technology Group (LTG)

September 23, 2015

Agenda

Last week

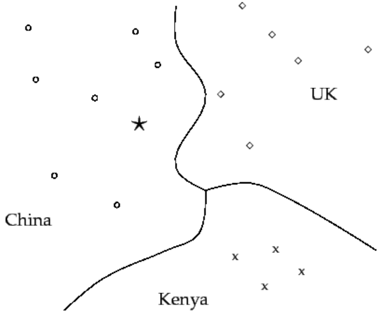
- Supervised vs unsupervised learning.
- Vectors space classification.
- How to represent classes and class membership.
- Rocchio + kNN.
- Linear vs non-linear decision boundaries.

Today

- Evaluation of classifiers
- Unsupervised machine learning for class discovery: Clustering
- Flat vs. hierarchical clustering.
- k-means clustering
- Vector space quiz

Testing a classifier

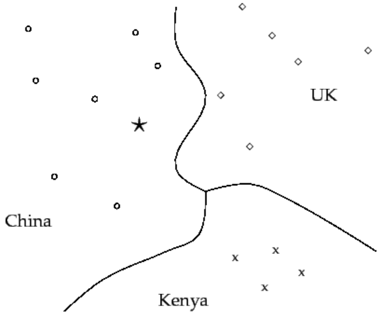
- Vector space classification amounts to computing the boundaries in the space that separate the class regions: *the decision boundaries*.
- To evaluate the boundary, we measure the number of correct classification predictions on unseeen test items.
- Many ways to do this...



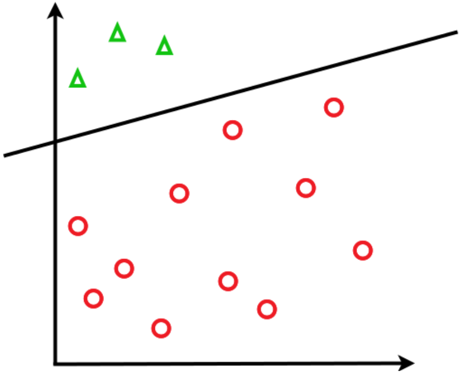
Testing a classifier

- 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.
- Labeled test data is sometimes refered to as the gold standard.
- Why can't we test on the training data?



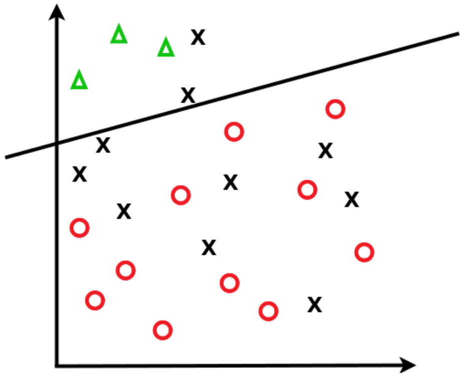
Example: Evaluating classifier decisions



- 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)

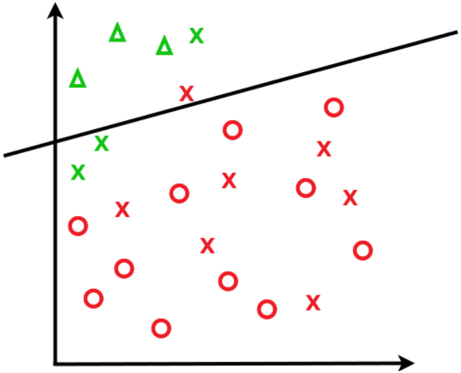
Example: Evaluating classifier decisions



- 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)

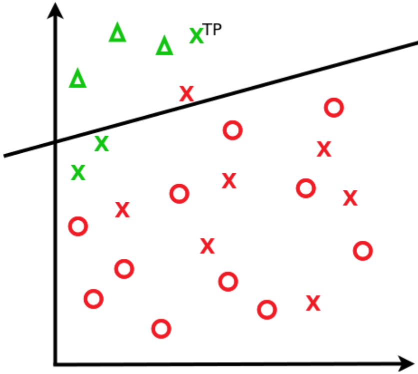
Example: Evaluating classifier decisions



- 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)

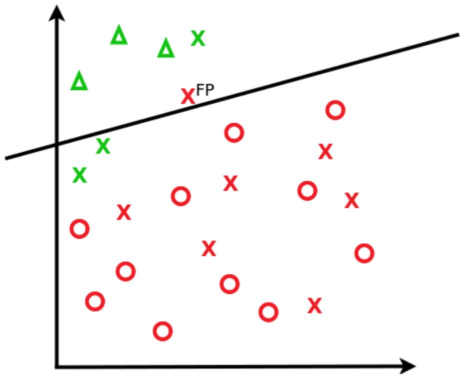
Example: Evaluating classifier decisions



- 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)

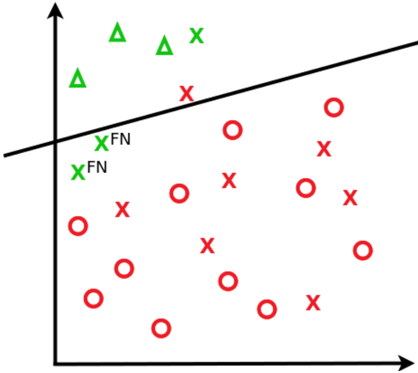
Example: Evaluating classifier decisions



- 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)

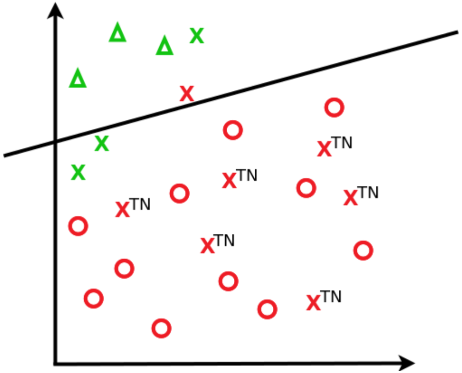
Example: Evaluating classifier decisions



- 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)

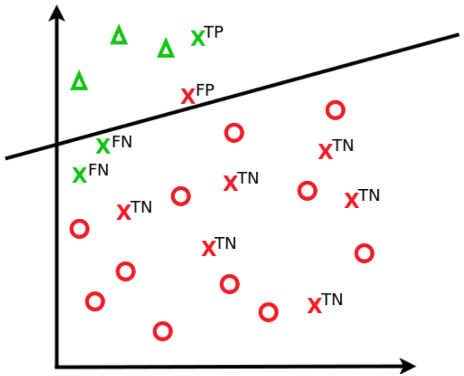
Example: Evaluating classifier decisions



- 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)

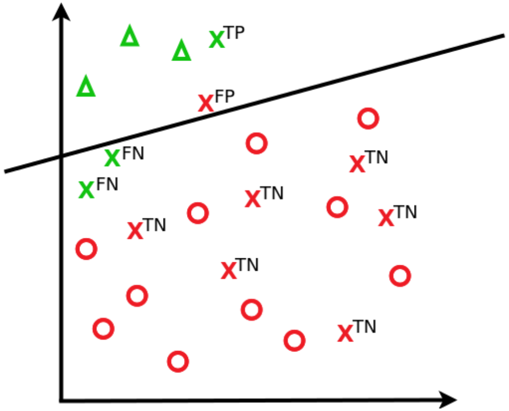
Example: Evaluating classifier decisions



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

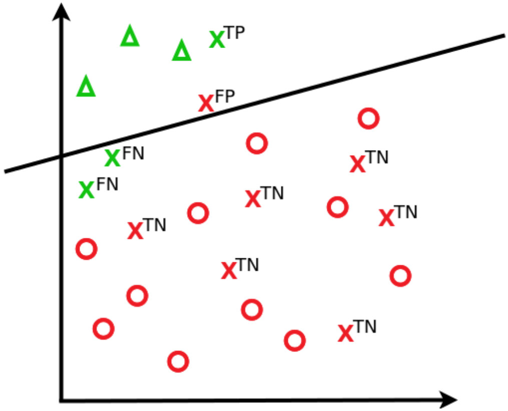
	gold = positive	gold = negative
prediction = positive	true positive (TP)	false positive (FP)
prediction = negative	false negative (FN)	true negative (TN)

Example: Evaluating classifier decisions



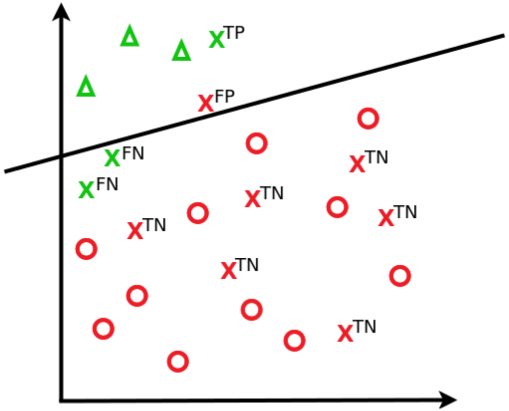
$$accuracy = \frac{TP+TN}{N}$$

Example: Evaluating classifier decisions



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

Example: Evaluating classifier decisions

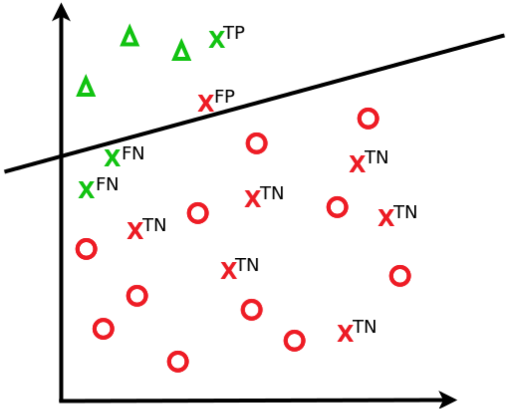


$$accuracy = \frac{TP+TN}{N}$$
$$= \frac{1+6}{10} = 0.7$$

$$precision = \frac{TP}{TP+FP}$$

$$recall = \frac{TP}{TP+FN}$$

Example: Evaluating classifier decisions

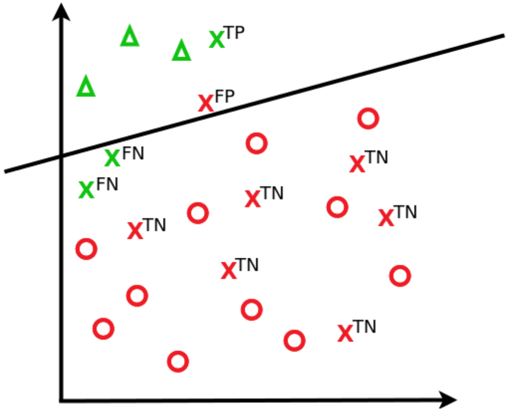


$$accuracy = \frac{TP+TN}{N} = \frac{1+6}{10} = 0.7$$

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

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

Example: Evaluating classifier decisions



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

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

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

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

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

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.

A note on obligatory assignment 2b

- Builds on oblig 2a: Vector space representation of a set of words based on BoW features extracted from a sample of the Brown corpus.
- For 2b we'll provide class labels for most of the words.
- Train a Rocchio classifier to predict labels for a set of unlabeled words.

Label	**Examples**
FOOD	*potato, food, bread, fish, eggs . . .*
INSTITUTION	*embassy, institute, college, government, school . . .*
TITLE	*president, professor, dr, governor, doctor . . .*
PLACE_NAME	*italy, dallas, france, america, england . . .*
PERSON_NAME	*lizzie, david, bill, howard, john . . .*
UNKNOWN	*department, egypt, robert, butter, senator . . .*

A note on obligatory assignment 2b

- For a given set of objects $\{o_1, \ldots, o_m\}$ the proximity matrix R is a square $m \times m$ matrix where R_{ij} stores the proximity of o_i and o_j.
- For our word space, R_{ij} would give the dot-product of the normalized feature vectors $\vec{x}_i$ and $\vec{x}_j$, representing the words o_i and o_j.

A note on obligatory assignment 2b

- For a given set of objects $\{o_1, \ldots, o_m\}$ the proximity matrix R is a square $m \times m$ matrix where R_{ij} stores the proximity of o_i and o_j.
- For our word space, R_{ij} would give the dot-product of the normalized feature vectors $\vec{x}_i$ and $\vec{x}_j$, representing the words o_i and o_j.
- Note that, if our similarity measure sim is symmetric, i.e. $\mathrm{sim}(\vec{x}, \vec{y}) = \mathrm{sim}(\vec{y}, \vec{x})$, then R will also be symmetric, i.e. $R_{ij} = R_{ji}$

A note on obligatory assignment 2b

- For a given set of objects $\{o_1, \ldots, o_m\}$ the proximity matrix R is a square $m \times m$ matrix where R_{ij} stores the proximity of o_i and o_j.
- For our word space, R_{ij} would give the dot-product of the normalized feature vectors $\vec{x}_i$ and $\vec{x}_j$, representing the words o_i and o_j.
- Note that, if our similarity measure sim is symmetric, i.e. $\mathrm{sim}(\vec{x}, \vec{y}) = \mathrm{sim}(\vec{y}, \vec{x})$, then R will also be symmetric, i.e. $R_{ij} = R_{ji}$
- Computing all the pairwise similarities *once* and then storing them in R can help save time in many applications.
  - R will provide the input to many clustering methods.
  - By sorting the row elements of R, we get access to an important type of similarity relation; nearest neighbors.
- For 2b we will implement a proximity matrix for retrieving knn relations.

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.
- "*The search for structure in data*" (Bezdek, 1981)
- 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.

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.

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.
- Dimensionality reduction / class-based features.

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.
- Dimensionality reduction / class-based features.
- News aggregation / topic directories.
- Social network analysis; identify sub-communities and user segments.
- Image segmentation, product recommendations, demographic analysis, . . .

Main types of clustering methods

Hierarchical

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

Flat

- Often referred to as partitional clustering.
- Tries to directly decompose the data into a set of clusters.
- Topic of today.

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,

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

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.

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*, $F_s = \mathrm{WCSS}$:

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

- 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.

k-means (cont'd)

Algorithm

Initialize: Compute centroids for k seeds.

Iterate:

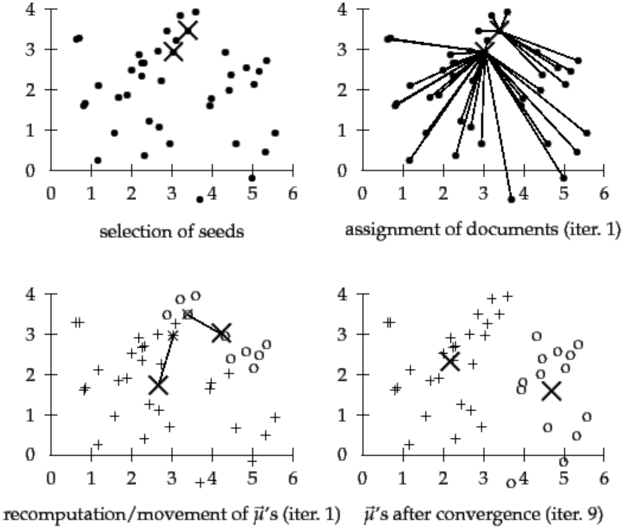
- Assign each object to the cluster with the nearest centroid.
- Compute new centroids for the clusters.

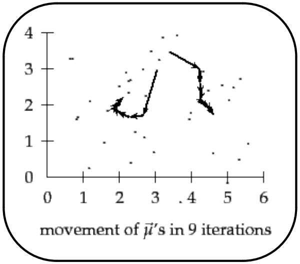
Terminate: When stopping criterion is satisfied.

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, $\mathrm{O}(kn)$.

k-means example for $k = 2$ in R^2 (Manning, Raghavan & Schütze 2008)





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 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 a hierarchical clustering on a subset of the data to find k initial clusters; etc..

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 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 a 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)

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.

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 random seeds as in k-means 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.

Connecting the dots

- Focus of the last two lectures: Rocchio / nearest centroid classification, kNN classification, and k-means clustering.
- Note how k-means clustering can be thought of as performing Rocchio classification in each iteration.

Connecting the dots

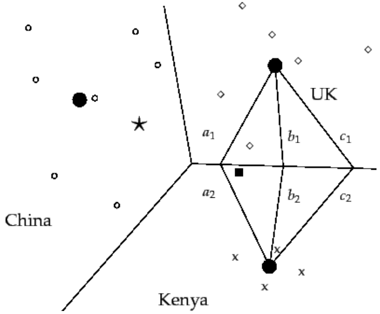
- Focus of the last two lectures: Rocchio / nearest centroid classification, kNN classification, and k-means clustering.
- Note how k-means clustering can be thought of as performing Rocchio classification in each iteration.
- Moreover, Rocchio can be thought of as a 1 Nearest Neighbor classifier with respect to the centroids.

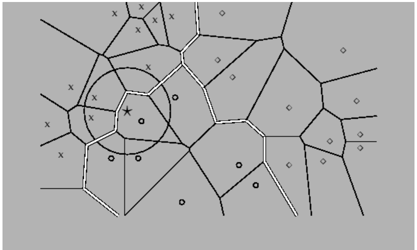
Connecting the dots

- Focus of the last two lectures: Rocchio / nearest centroid classification, kNN classification, and k-means clustering.
- Note how k-means clustering can be thought of as performing Rocchio classification in each iteration.
- Moreover, Rocchio can be thought of as a 1 Nearest Neighbor classifier with respect to the centroids.
- How can this be? Isn't kNN non-linear and Rocchio linear?

Connecting the dots

- Recall that the kNN decision boundary is locally linear for each cell in the Voronoi diagram.
- For both Rocchio and k-means, we're partitioning the observations according to the Voronoi diagram generated by the centroids.





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

Next

- Hierarchical clustering.
- 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.
- How to measure the inter-cluster similarity ("linkage criterions").

Agglomerative clustering

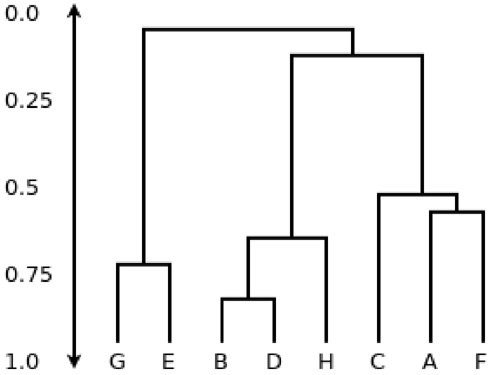
- Initially; regards each object as its own singleton cluster.
- Iteratively "agglomerates" (merges) the groups in a bottom-up fashion.
- Each merge defines a binary branch in the tree.
- Terminates; when only one cluster remains (the root).

```
parameters: $\{o_1, o_2, \ldots, o_n\}$, sim
$C = \{\{o_1\}, \{o_2\}, \ldots, \{o_n\}\}$
$T = []$
do for $i = 1$ to $n - 1$
    $\{c_j, c_k\} \leftarrow \underset{\{c_j, c_k\} \subseteq C \,\wedge\, j \neq k}{\arg\max} \text{sim}(c_j, c_k)$
    $C \leftarrow C \backslash \{c_j, c_k\}$
    $C \leftarrow C \cup \{c_j \cup c_k\}$
    $T[i] \leftarrow \{c_j, c_k\}$
```

- At each stage, we merge the pair of clusters that are most similar, as defined by some measure of inter-cluster similarity; sim.
- Plugging in a different sim gives us a different sequence of merges T.

Dendrograms

- A hierarchical clustering is often visualized as a binary tree structure known as a *dendrogram*.
- A merge is shown as a horizontal line connecting two clusters.
- The y-axis coordinate of the line corresponds to the *similarity* of the merged clusters.



- We here assume dot-products of normalized vectors (self-similarity $= 1$).

Definitions of inter-cluster similarity

- So far we've looked at ways to the define the similarity between
  - pairs of objects.
  - objects and a class.
- Now we'll look at ways to define the similarity between *collections*.

Definitions of inter-cluster similarity

- So far we've looked at ways to the define the similarity between
  - pairs of objects.
  - objects and a class.
- Now we'll look at ways to define the similarity between *collections*.
- In agglomerative clustering, a measure of cluster similarity $\mathrm{sim}(c_i, c_j)$ is usually referred to as a *linkage criterion*:
  - Single-linkage
  - Complete-linkage
  - Centroid-linkage
  - Average-linkage
- The linkage criterion determines which pair of clusters we will merge to a new cluster in each step.

Bezdek, J. C. (1981). *Pattern recognition with fuzzy objective function algorithms.* Plenum Press.