Web-Semantik-Technologien

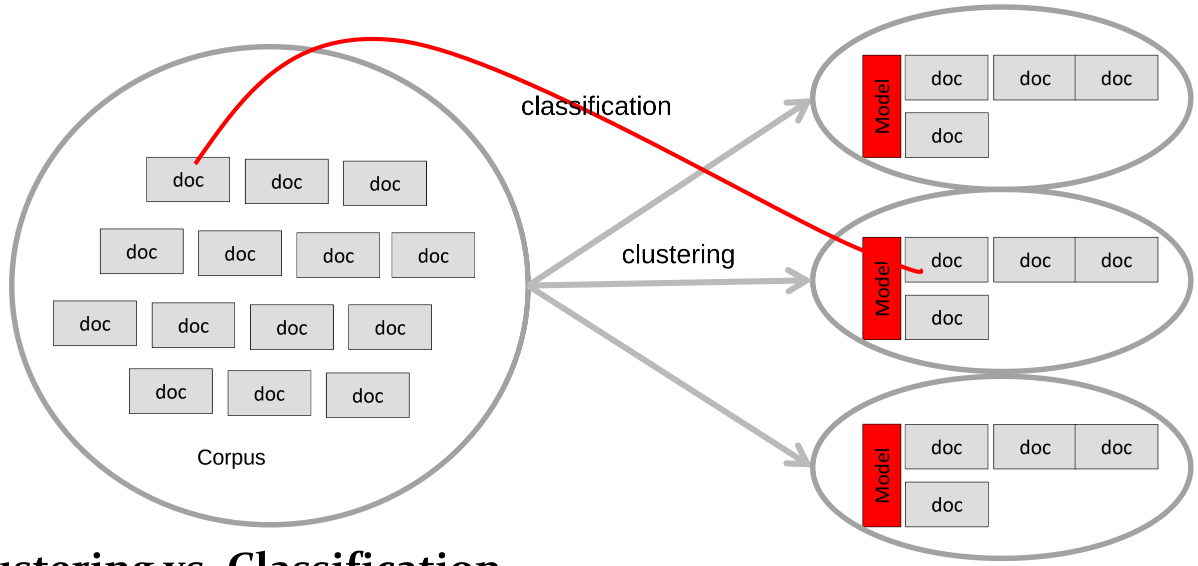
Vorlesung WS 2019/20

thomas.kern@fh-hagenberg.at

Clustering and Classification in IR (Overview)

What is clustering?

- ***Clustering:*** the process of grouping a set of objects (documents) into classes of similar objects (documents)
  - **Documents within a cluster should be similar**
  - **Documents from different clusters should be dissimilar**



Clustering vs. Classification

Clustering	***infers*** groups based on clustered objects
Classification	***assigns*** objects to ***predefined*** groups

Applications of clustering in IR

- ***Clustering:*** The commonest form of unsupervised learning
  - ***Unsupervised learning*** = *learning from raw data, as opposed to supervised data where a classification of examples is given*

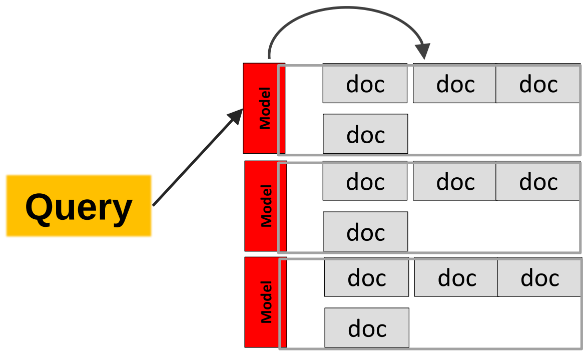
- ***Cluster hypothesis:*** Documents in the same cluster behave similarly with respect to relevance to information needs.

- Therefore, to ***improve search recall***:
  1. Cluster documents in corpus a priori
  2. When a query matches a document *d*, also return other documents in the cluster containing *d*

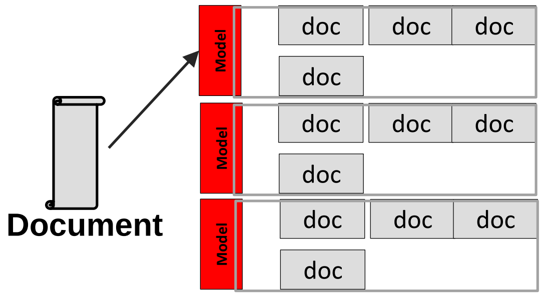
For example the query *car* will also return docs containing *automobile*, because clustering grouped together docs containing *car* with those containing *automobile*.

Applications of clustering in IR

- **For speeding up vector space retrieval**
  Cluster-based retrieval gives faster search



- **For improving recall in search applications**
  Better search results



Some requirements and problems

- **Fast**
  - Immediate response to query
- **Flexible**
  - Web content changes constantly
- **User-oriented**
  - Main goal is to aid the user in finding sought information
- **Online or offline clustering?**
- **What to use as input**
  - Entire documents
  - Structure information (links)
  - Other data (i.e. click-through)
  - Use stop word lists, stemming, etc.
- **How to define similarity?**
  - Content (vector-space model)
  - Link analysis
  - Usage statistics
- **How to group similar documents?**
  - What does similarity mean semantically?
  - Statistical vs. semantical proximity?
- **How to label the groups?**

Main issues for clustering

- **Representation for clustering**
  - Document representation
    - Vector space? Vectors are sparsely filled. Length: ~ 10000 terms
    - Normalization? Centroids are not length normalized
- Need a notion of similarity/distance; Distance: Comparison is time intensive and not very effective
- **How many clusters?**
  - Fixed a priori?
  - Completely data driven?
    - Avoid trivial clusters - too large or too small
    - If a cluster's too large, then for navigation purposes you have wasted an extra user click without whittling down the set of documents much.
  - How to find the optimal number of clusters?
- **Flat or hierarchical?**
- **Overlapping? Hard or soft?**
- **Incremental?**

Clustering algorithms

- **Distance-based**
  - **Flat**
    Usually start with a random (partial) partitioning. Refine it iteratively
    - K-mean (can be fuzzy C-mean)
    - Single-pass (incremental)
    - Model based clustering

- **Hierarchical**
  - Agglomerative Hierarchical Clustering (HAC)
  - Bottom-up, agglomerative
  - Top-down, divisive

- **Other**
  - Suffix Tree Clustering (Grouper)
  - Self-organizing maps (SOM Kohonen- neural networks)
  - Latent Semantic Indexing (LSI) (reducing the dimensionality of the vector-space)

Notion of similarity/distance

- **Ideal:** semantic similarity.
- **Practical:** term-statistical similarity

- ***Documents as vectors.***
- We will mostly speak of Euclidean distance. ***But real implementations use cosine similarity***
- For many algorithms, it is easier to think in terms of a ***distance (rather than similarity)*** between docs.

General Partitioning Algorithm

- ***Partitioning method:***
  Construct a partition of *m* documents into a set of *k* clusters. Given: a set of documents and the number *k*.

- ***Task:***
  Find a partition of *k* clusters that optimizes the chosen partitioning criterion

- Typically the number of desired clusters *k* has to be provided.

1. Randomly choose *k* instances as seeds, one per cluster.
2. Form initial clusters based on these seeds.
3. Iterate, repeatedly reallocating instances to different clusters to improve the overall clustering.
4. Stop when clustering converges or after a fixed number of iterations.

Clustering Examples
K-Means

- Assumes documents are real-valued vectors.
- Clusters based on *centroids* (the *center of gravity* or *mean*) of points in a cluster c

$$\vec{\mu}(\mathrm{c}) = \frac{1}{|c|} \sum_{\vec{x} \in c} \vec{x}$$

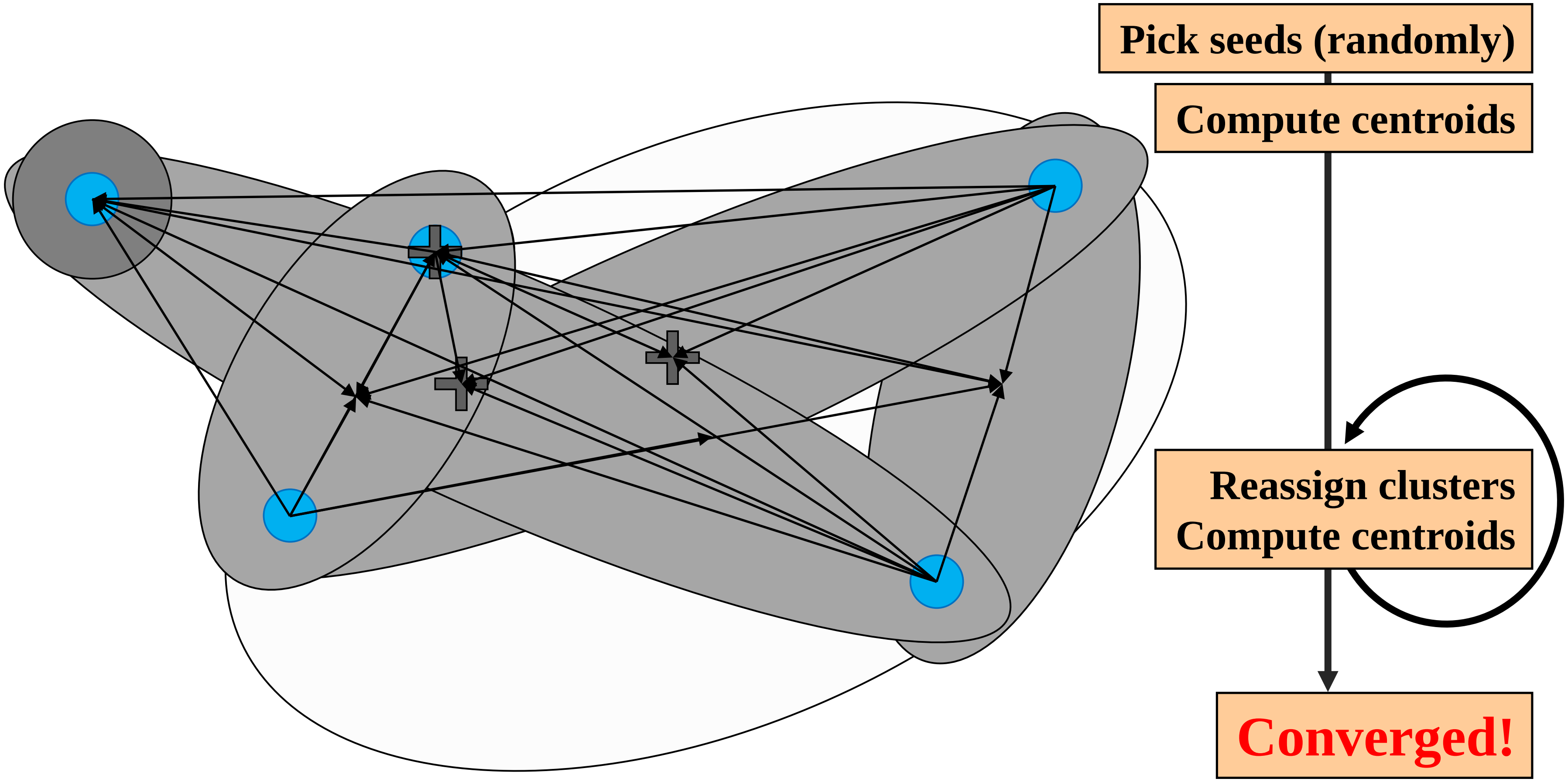
- Reassignment of instances to clusters is based on distance to the current cluster centroids.

Clustering Examples K-Means Algorithm

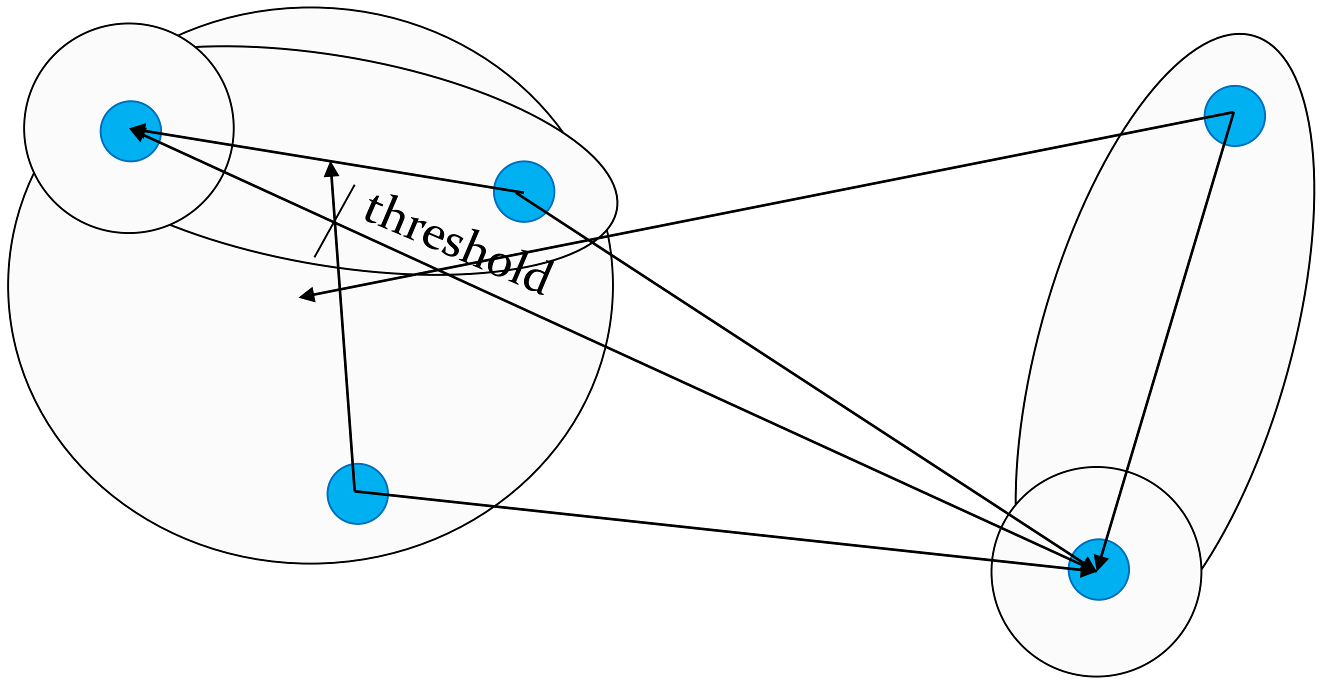
```
1. Select $K$ random documents $\{s_1, s_2, \ldots s_K\}$ as seeds.

2. Until clustering *converges* (or other stopping criterion):
       For each doc $d_i$:
           Assign $d_i$ to the cluster $c_j$ such that $dist(x_i, s_j)$
           is minimal.
           Next, update the seeds to the centroid of each
           cluster
       For each cluster $c_j$ compute
           $s_j = \mu(c_j)$
```

Clustering Examples
K-Means Algorithm

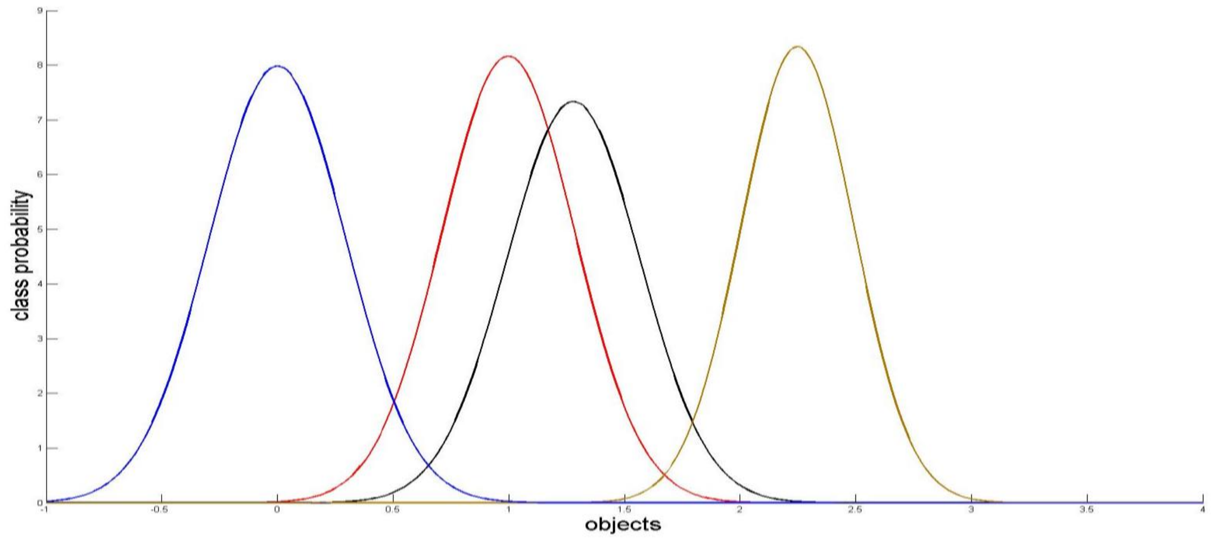


Clustering Examples
Single-pass Algorithm



Clustering Examples
Soft Clustering

- Clustering typically assumes that each instance is given a “hard” assignment to exactly one cluster.
- Does not allow uncertainty in class membership or for an instance to belong to more than one cluster.
- *Soft clustering* gives probabilities that an instance belongs to each of a set of clusters.
- Each instance is assigned a probability distribution across a set of discovered categories (probabilities of all categories must sum to 1).



Clustering Examples
Fuzzy - Clustering

Fuzzy c-means (FCM) is a method of clustering which allows one piece of data to belong to two or more clusters. This method is frequently used in pattern recognition. It is based on minimization of the following objective function:

$$J(\mathbf{X},\mathbf{U},\mathbf{B}) = \sum_{j=1}^{n}\sum_{i=1}^{c} u_{i,j}^{m} \cdot d^2(\vec{\beta}_i, \vec{x}_j).$$

where ***x*** is document vector, ***β*** is centroid vector of cluster, ***u*** is vector of membership functions. Constraints:

$$\begin{array}{ll} \sum_{i=1}^{c} u_{i,j} = 1 & \text{für alle } j \in \{1,\ldots,n\} \\ \sum_{j=1}^{n} u_{i,j} > 0 & \text{für alle } i \in \{1,\ldots,c\} \end{array}$$

Iterative calculation of membership values

$$u_{i,j} = \begin{cases} \dfrac{1}{\sum\limits_{k=1}^{c} \left(\dfrac{d^2(\vec{x}_j, \vec{\beta}_i)}{d^2(\vec{x}_j, \vec{\beta}_k)} \right)^{\frac{1}{m-1}}}, \\ 0, \\ x, x \in [0,1], \text{ so daß } \sum_{i \in I_j} u_{i,j} = 1 \text{ gilt}, \end{cases}$$

Clustering Examples
Model based clustering

K-mean is a special case of a general procedure known as the Expectation Maximization (EM) algorithm.

- Probabilistic method for soft clustering, soft version of *K*-mean.
- Direct method that assumes *K* clusters: $\{c_1, c_2, \ldots c_K\}$
- Assumes a probabilistic model of categories that allows computing $P(c_i \mid E)$ for each category c_i for a given example E.
- For text, typically assume a naïve -Bayes category model.
    - A naïve Bayes classifier assumes that the presence or absence of a particular feature *(x)* is unrelated to the presence or absence of any other feature, given the class variable.

- **Parameters** $\theta = \{P(c_i), P(x_j \mid c_i) \mid i \in \{1,\ldots k\} \text{ and } j \in \{1,\ldots,|V|\}\}$

Naïve Bayes Model – Example Training set

	x				c
Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Naïve Bayes Model – Example Play Tennis?

- **Example:** <Outlk=sun, Temp=cool, Humid=high, Wind=strong>

- **Estimatet parameters** θ
  - P(yes) = 9/14
  - P(no) = 5/14
  - P(Wind=strong|yes) = 3/9
  - P(Wind=strong|no) = 3/5
  - …

- **We compute**
  - P(yes) P(sun|yes) P(cool|yes) P(high|yes) P(strong|yes) =
    9/14 * 2/9 * 3/9 * 3/9 * 3/9 = 0.005
  - P(no) P(sun|no) P(cool|no) P(high|no) P(strong|no) =
    5/14 * 3/5 * 1/5 * 4/5 * 3/5 = 0.021

- **Therefore this new example is classified to "no"**

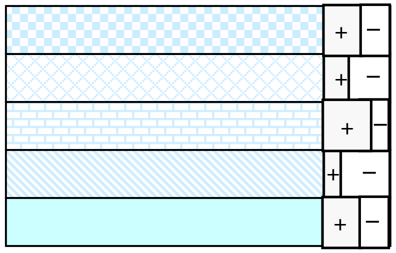
EM Algorithm

- Iterative method for learning probabilistic categorization model from unsupervised data.
- Initially assume random assignment of examples to categories.
- Use standard naive-Bayes training to learn a probabilistic model with parameters θ from the labeled data.
- Until convergence or until maximum number of iterations reached:
  - **Expectation (E-Step):** Use the naive Bayes model θ to compute $P(c_i \mid E)$ for each category and example, and re-label each example using these probability values as soft category labels (based on these posterior probability estimates)
  - **Maximization (M-Step):** Use standard naive-Bayes training to re-estimate the parameters θ using these new probabilistic category labels.

Expectation Maximization (EM)

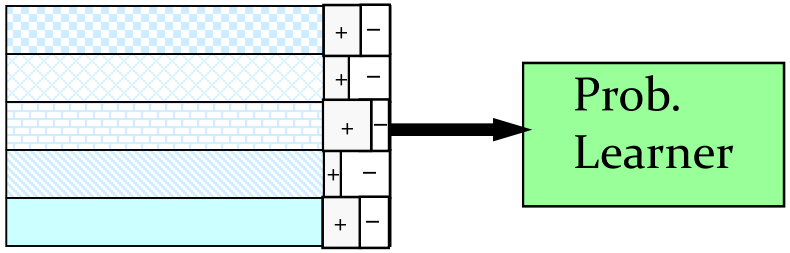
- **Initialize:** Assign random probabilistic labels to unlabeled data

Unlabeled Examples



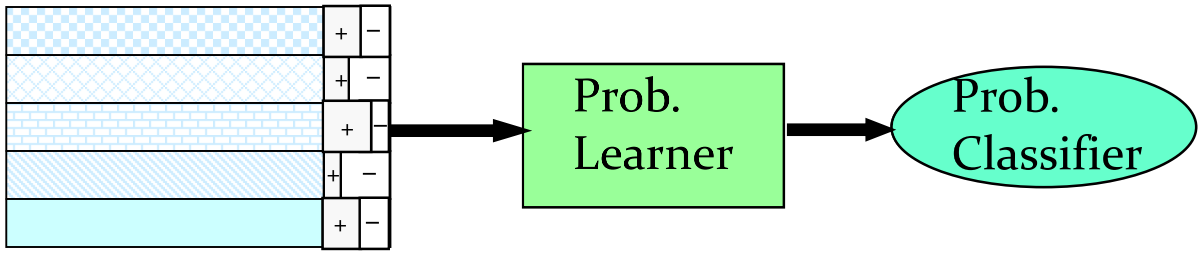
Expectation Maximization (EM)

- **Initialize:** Give soft-labeled training data to a probabilistic learner



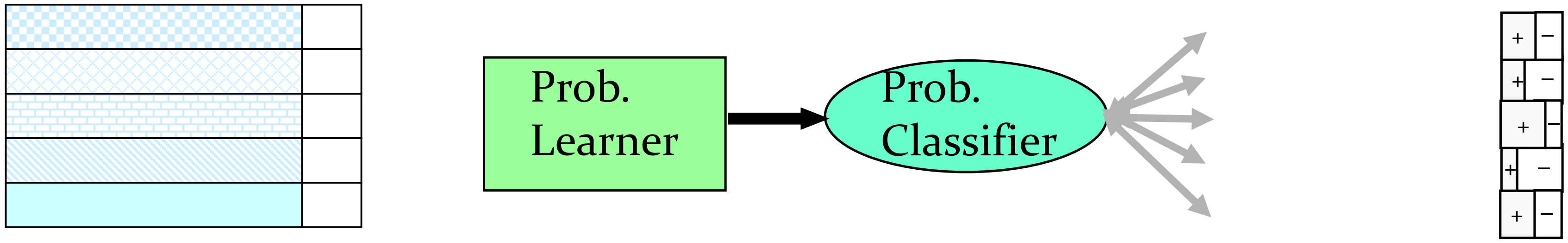
Expectation Maximization (EM)

- **Initialize:** Produce a probabilistic classifier



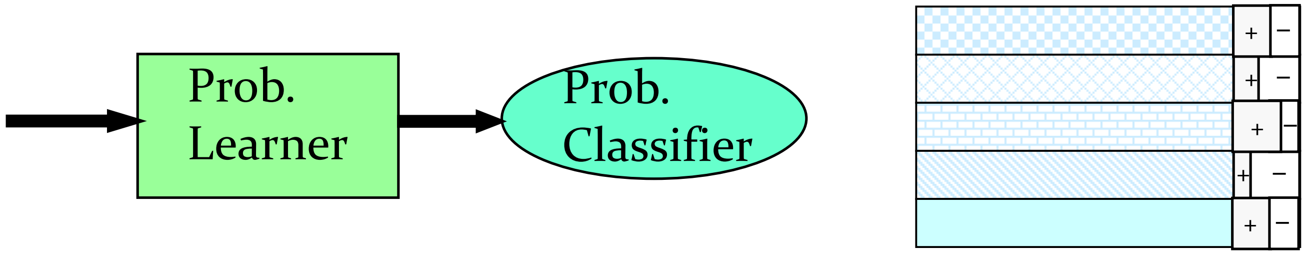
Expectation Maximization (EM)

- **E Step:** Relabel unlabled data using the trained classifier



Expectation Maximization (EM)

- **M Step:** Retrain classifier on relabeled data



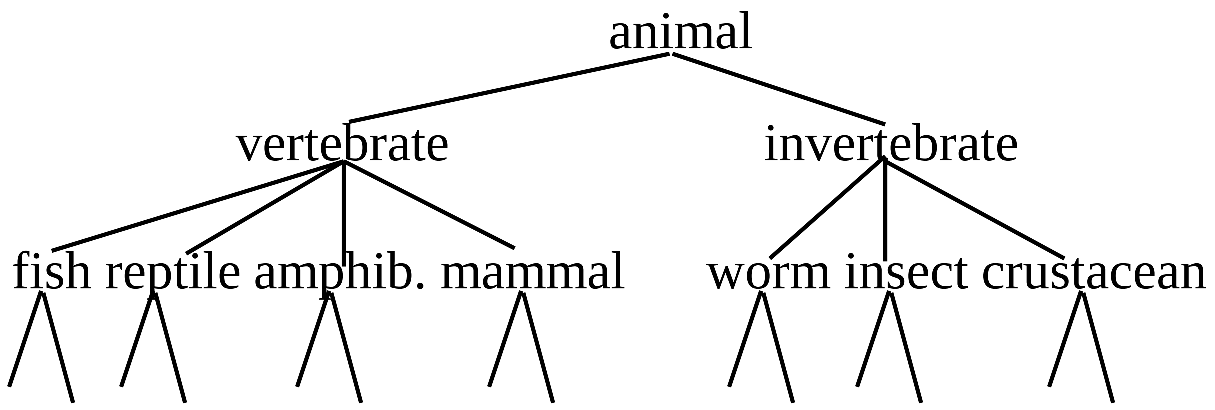
Continue EM iterations until probabilistic labels on unlabeled data converge.

Semi-Supervised Learning

- **Use EM in a *semi-supervised* mode by training EM on both labeled and unlabeled data.**
  - Train initial probabilistic model on user-labeled subset of data instead of randomly labeled unsupervised data.
  - Labels of user-labeled examples are "frozen" and never relabeled during EM iterations.
  - Labels of unsupervised data are constantly probabilistically relabeled by EM.

Clustering Examples
Hirachical Clustering

Build a tree-based hierarchical taxonomy (*dendrogram*) from a set of documents.

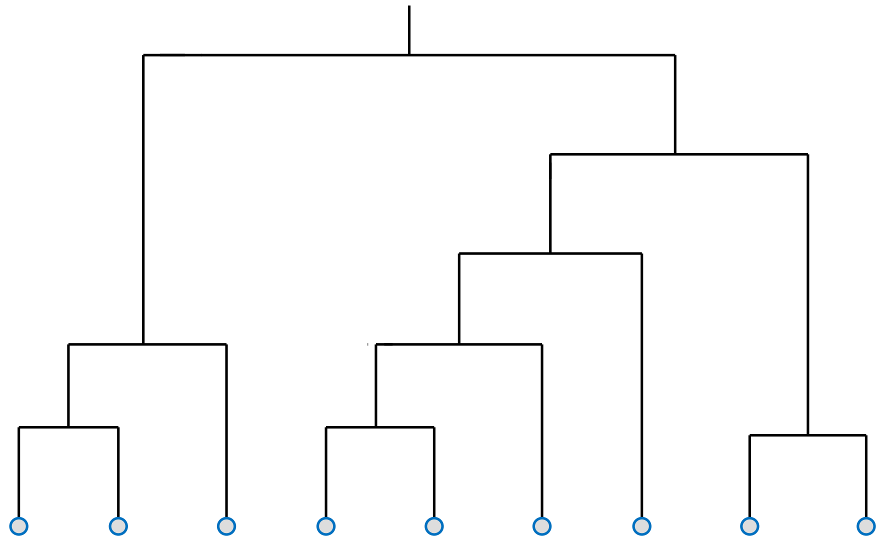


One approach: recursive application of a partitional clustering algorithm.

Clustering Examples Hirachical Clustering

Clustering obtained by cutting the dendrogram at a desired level:

Each connected component forms a cluster.



Clustering Examples
Hirachical Agglomerative Clustering (HAC)

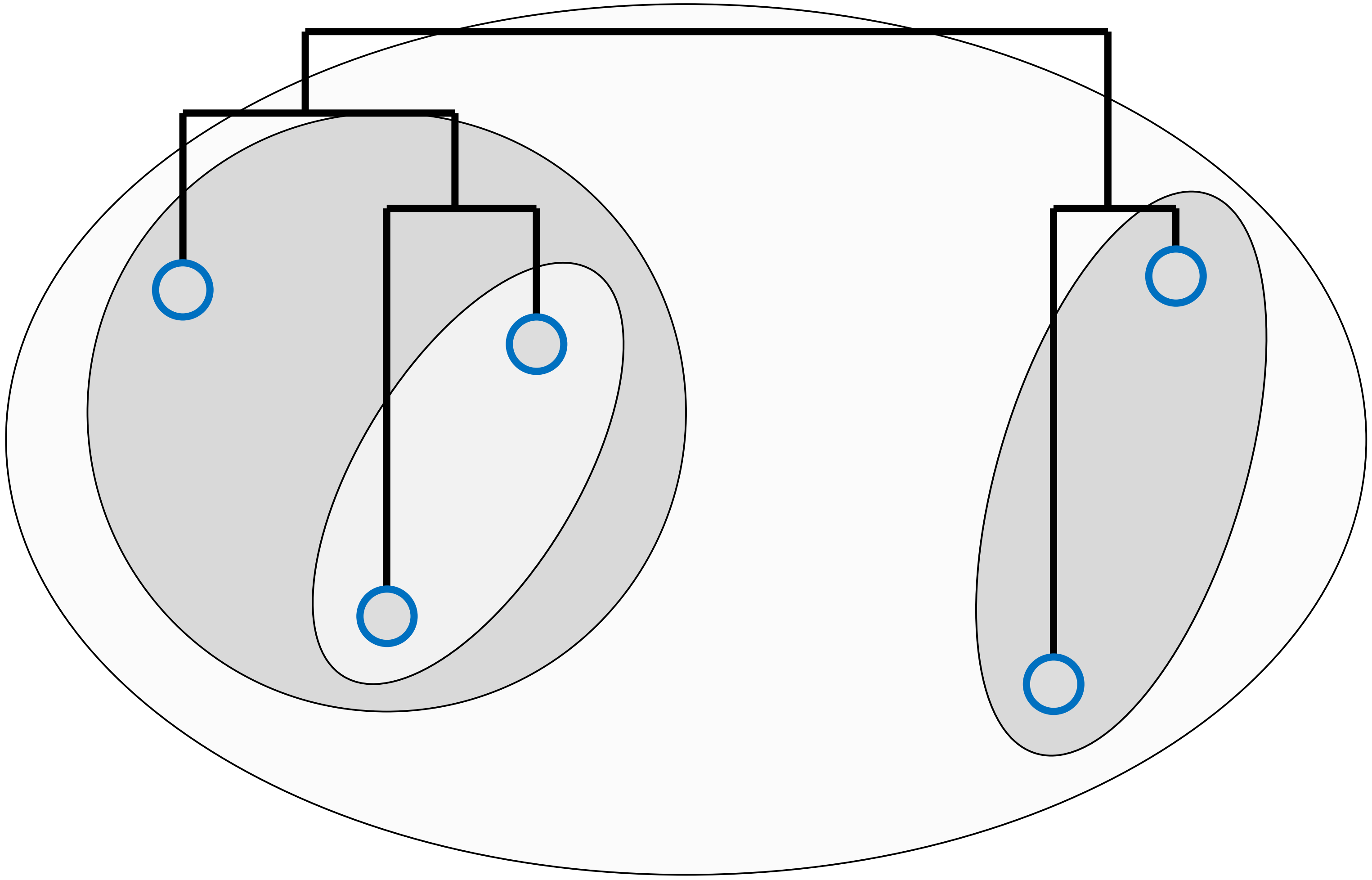
- **Starts with each doc in a separate cluster**
  - Then repeatedly joins the ***closest pair*** of clusters, until there is only one cluster.
- **The history of merging forms a binary tree or hierarchy.**

1. Start with all instances in their own cluster.

2. Until there is only one cluster:

   - Among the current clusters, determine the two clusters, c_i and c_j, that are most similar.
   - Replace c_i and c_j with a single cluster $c_i \cup c_j$

Clustering Examples
Hirachical Clustering



Clustering Examples Hirachical Clustering

- **Cluster Similarity:** Many variants to defining closest pair of clusters
  - **Single-link**
    - Similarity of the most cosine-similar (single-link)
  - **Complete-link**
    - Similarity of the "furthest" points, the least cosine-similar
  - **Centroid**
    - Clusters whose centroids (centers of gravity) are the most cosine-similar
  - **Average-link**
    - Average cosine between pairs of elements

Clustering Examples
Buckshot Algorithm

- **Combines HAC and K-Means clustering**

1. First randomly take a sample of instances of size $\sqrt{m}$
2. Run group-average HAC on this sample, which takes only O(m) time.
3. Use the results of HAC as initial seeds for K-means.

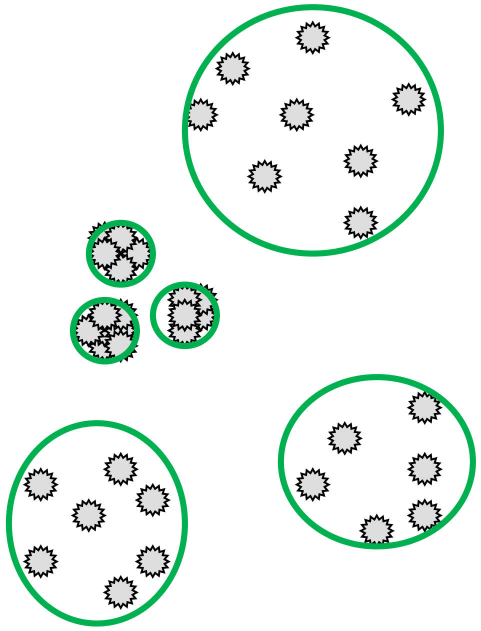
- **Overall algorithm is O(m) and avoids problems of bad seed selection.**

What is a good clustering?

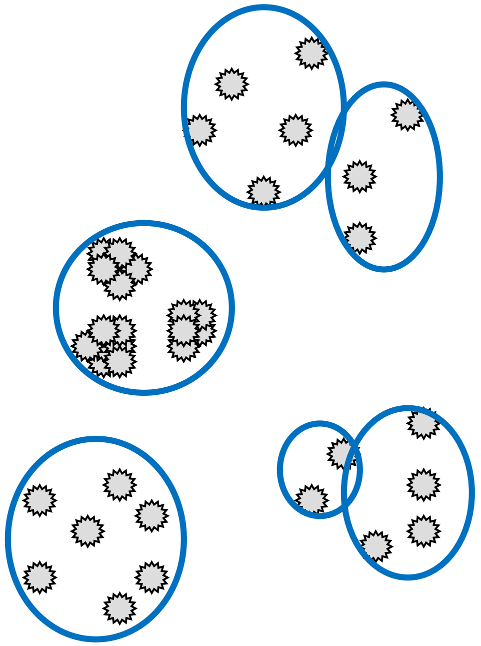
Internal criterion: A good clustering will produce high quality clusters in which:

1. the ***intra-class*** (that is, intra-cluster) similarity is high
2. the ***inter-class*** similarity is low

The measured quality of a clustering depends on both the document representation and the similarity measure used



Intuitively "correct" clustering



HAC-generated clusters

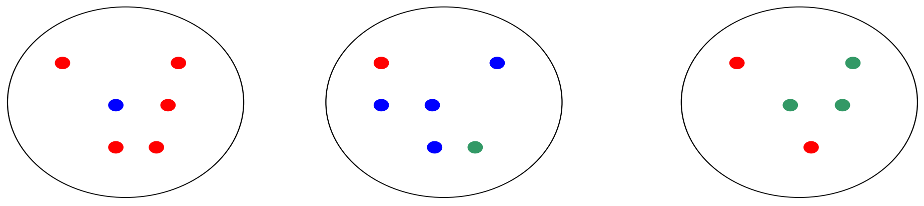
External Evaluation of Cluster Quality

- Simple measure: ***purity***, the ratio between the dominant class in the cluster and the size of cluster

$$Purity(C) = \frac{1}{m} \max_j (m_j) \quad j \in C$$

m number of objects

- Others are ***entropy of classes in clusters*** (or mutual information between classes and clusters)



Cluster I Cluster II Cluster III

Cluster I:	Purity = 1/6 (max(5, 1, 0)) = 5/6
Cluster II:	Purity = 1/6 (max(1, 4, 1)) = 4/6
Cluster III:	Purity = 1/5 (max(2, 0, 3)) = 3/5

Rand Index measures between pair decisions

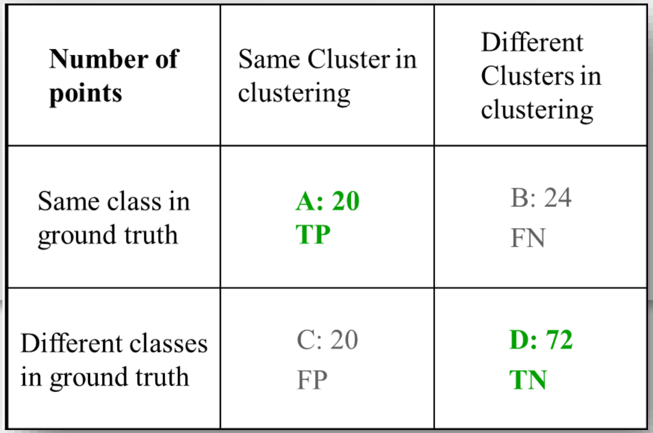
Number of points	Same Cluster in clustering	Different Clusters in clustering
Same class in ground truth	**A: 20 TP**	B: 24 FN
Different classes in ground truth	C: 20 FP	**D: 72 TN**

RI = 0.68

$$Rand\ \ Index = RI = \frac{A + D}{A + B + C + D}$$

The Rand Index measures the percentage of decisions that are correct.

Rand index and Cluster F-measure



Number of points	Same Cluster in clustering	Different Clusters in clustering
Same class in ground truth	**A: 20 TP**	B: 24 FN
Different classes in ground truth	C: 20 FP	**D: 72 TN**

Rand Index - Accuracy $$RI = \frac{A+D}{A+B+C+D}$$ ... percentage of decisions that are correct.

Precision $$P = \frac{A}{A+C}$$ ... fraction of retrieved documents that are relevant to the find

Recall $$R = \frac{A}{A+B}$$ ... fraction of the documents that are relevant to the query that are successfully retrieved

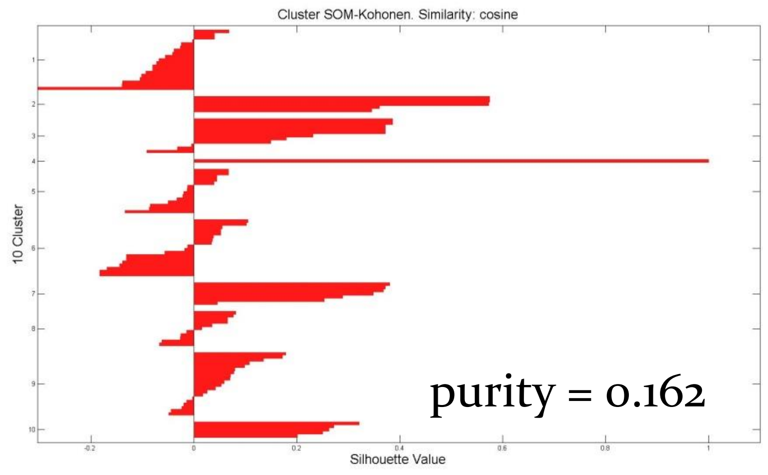
F-measure $$F = 2\frac{PR}{P+R}$$ ... can be interpreted as a weighted average of the precision and recall

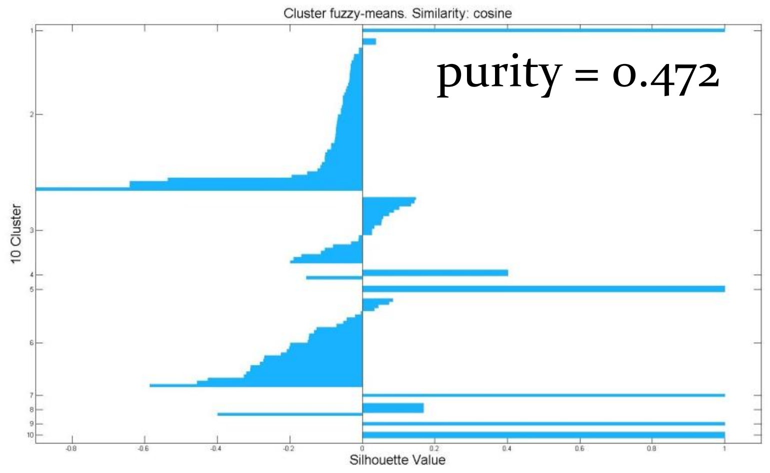
Silhouette Values

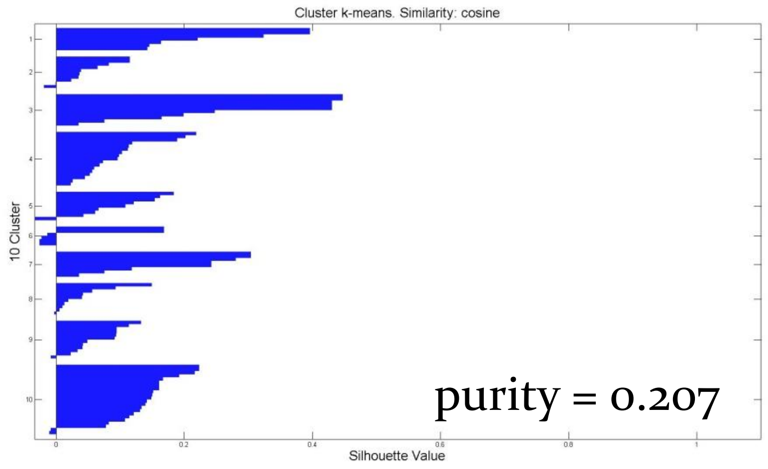
- **Good clusters have the property that cluster members are close to each other and far from members of other clusters.**

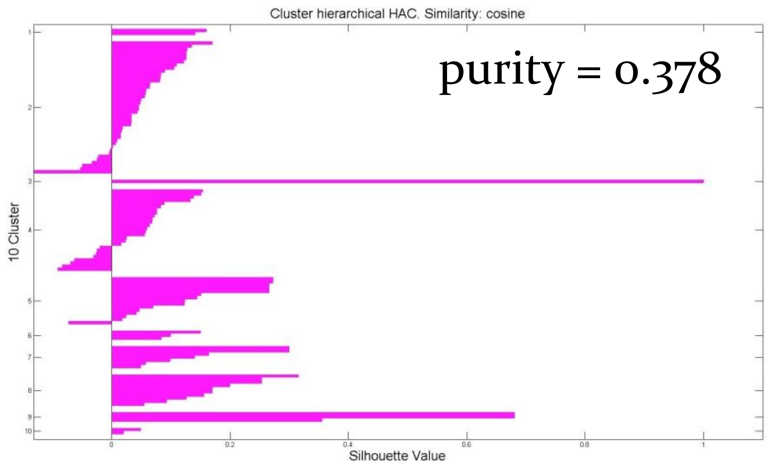
- Calculate the average distance of each cluster member within its cluster. This is its *dissimilarity* from its cluster
- Silhouette values range from -1 to 1
  - 0… a record is right on the boundary of two clusters.
  - 1… it is identical to the other records in its own cluster.
  - -1… the record is more similar to the records of its neighboring cluster than to other members of its own cluster

Cluster Validation Example









Classification

Given:

- A description of an instance $d \in X$

  X is the *instance language* or *instance space*.

  (How to represent text documents. Usually some type of high-dimensional space)

- A fixed set of classes: $C = \{c_1, c_2, \ldots, c_K\}$

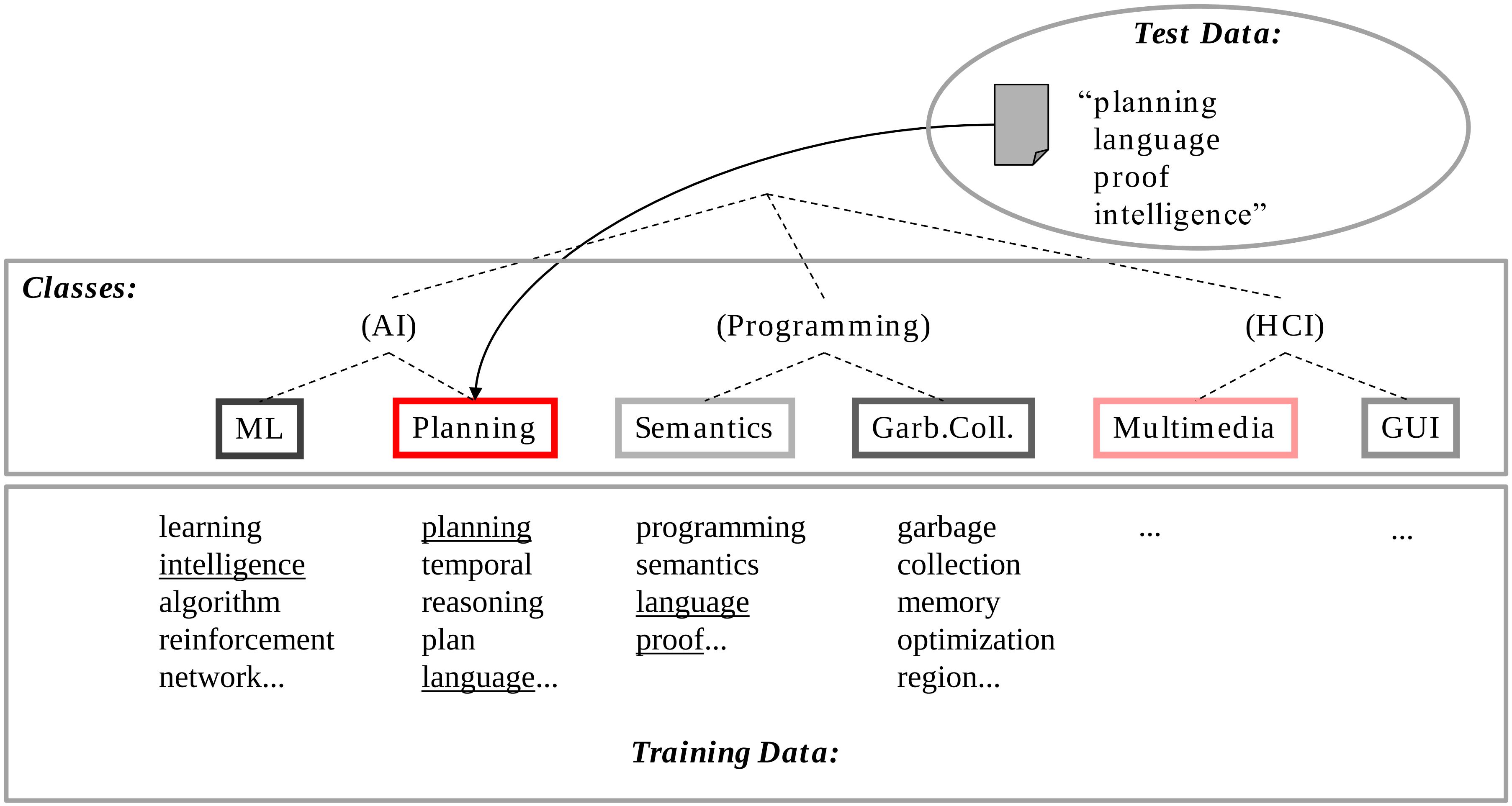
Determine:

- The category (class) of d : $\gamma(d) \in C$

  where $\gamma(d)$ is a *classification function* whose domain is X and whose range is C.

  (How to build classification functions ("classifiers").)

Document Classification



Classification Methods

- **Manual classification**
  - Used by the original Yahoo! , Looksmart, PubMed
  - Very accurate when job is done by experts and consistent when the problem size and team is small
  - Difficult and expensive to scale - we need automatic classification methods for big problems

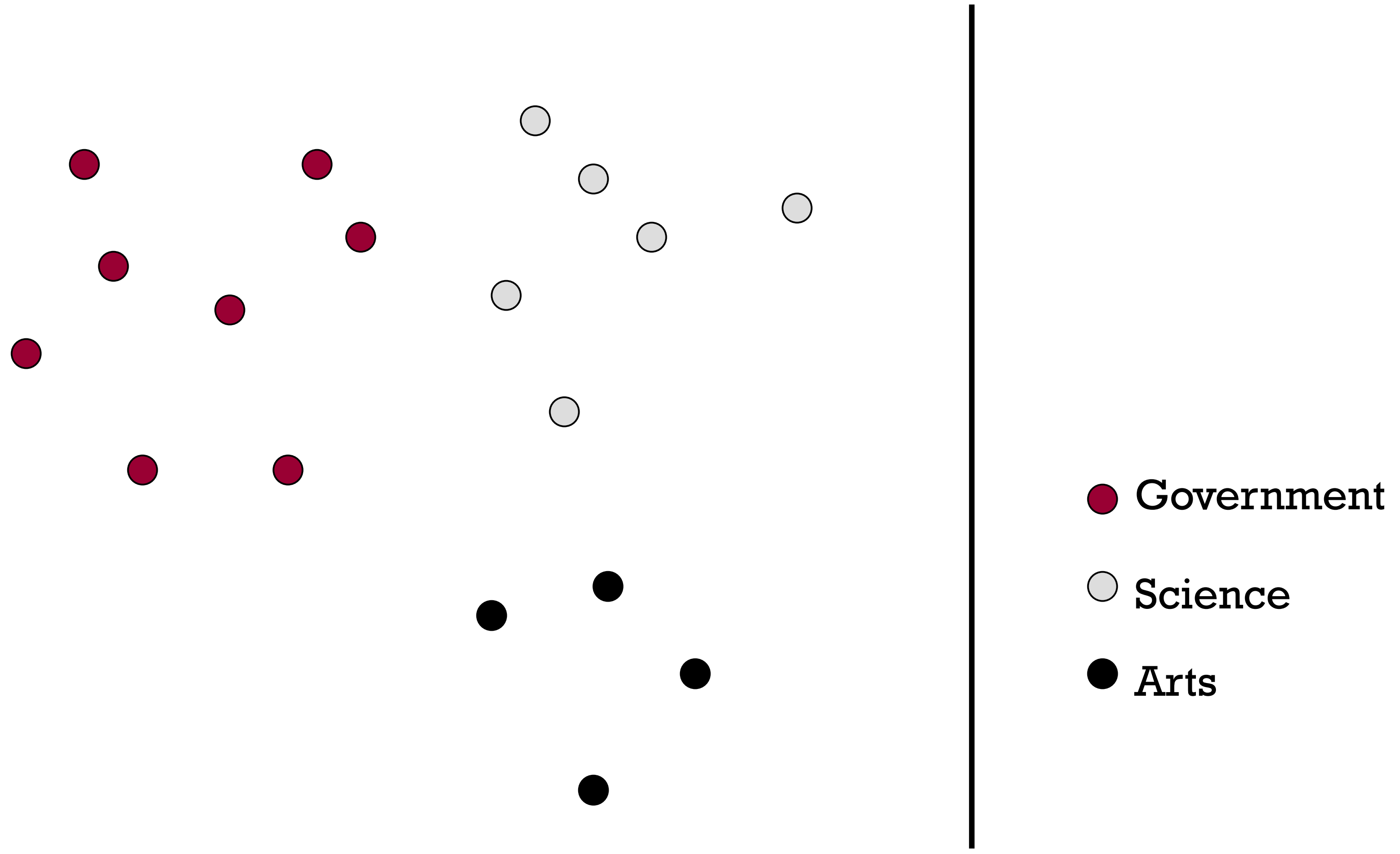
- **Automatic document classification**
  - Hand-coded rule-based systems
    - It is what Google Alerts is doing- widely deployed in government and enterprise
    - assign category if document contains a given boolean combination of words
    - Standing queries: Commercial systems have complex query languages (everything in IR query languages + score accumulators)
    - Accuracy is often very high if a rule has been carefully refined over time by a subject expert but building and maintaining these rules is expensive

- Supervised learning of a document-label assignment function
  - Many systems partly rely on machine learning
  - k-Nearest Neighbors (simple, powerful)
  - Naïve Bayes (simple, common method)
  - Support-vector machines (new, more powerful)
  - Many commercial systems use a mixture of methods

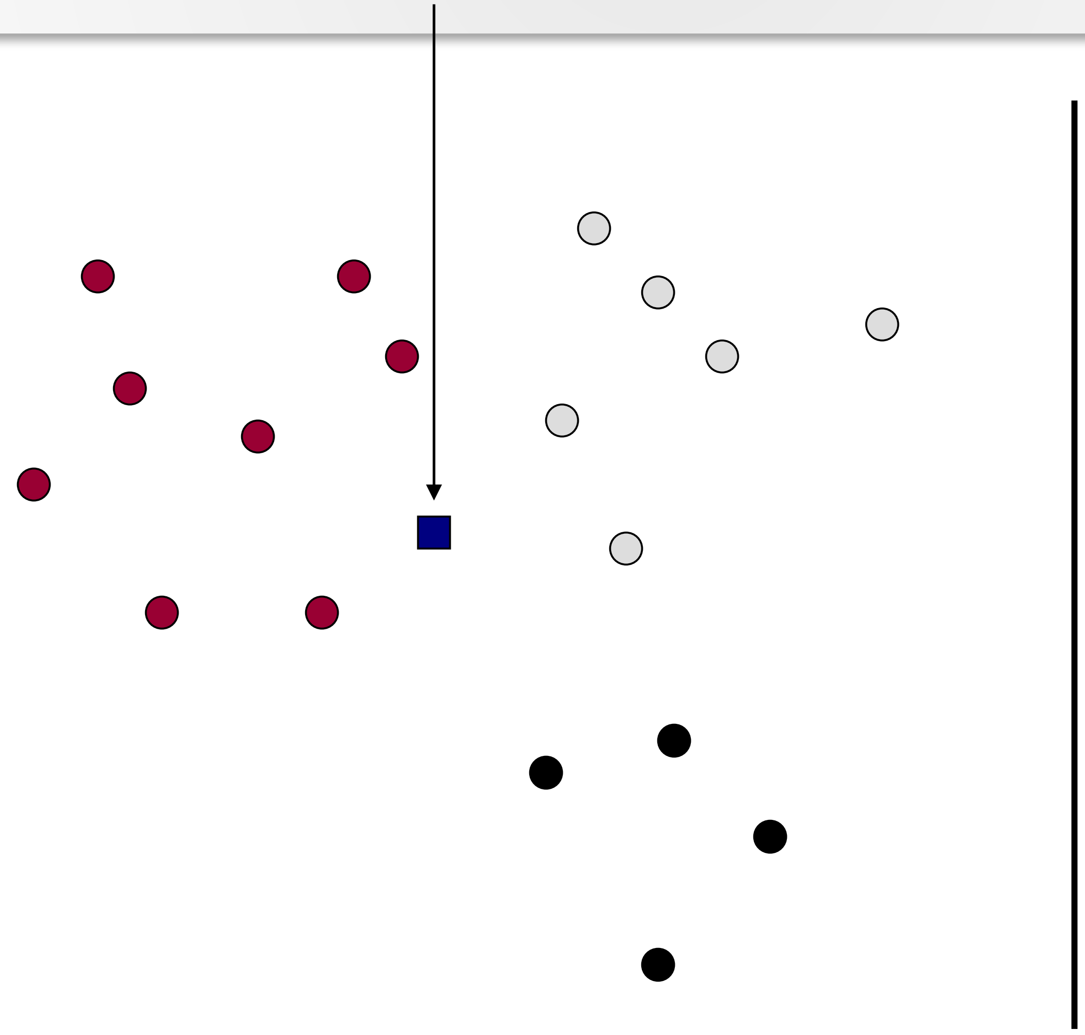
Classification using Vector Spaces

- The ***training set*** is a set of documents, each labeled with its class (e.g., topic)
- In vector space classification, this set corresponds to a ***labeled set of points*** (or, equivalently, vectors) in the vector space
- Premise 1: Documents in the same class form a contiguous region of space
- Premise 2: Documents from different classes do not overlap
- We define surfaces to delineate classes in the space

Documents in a Vector Space



Test Document of what Class?

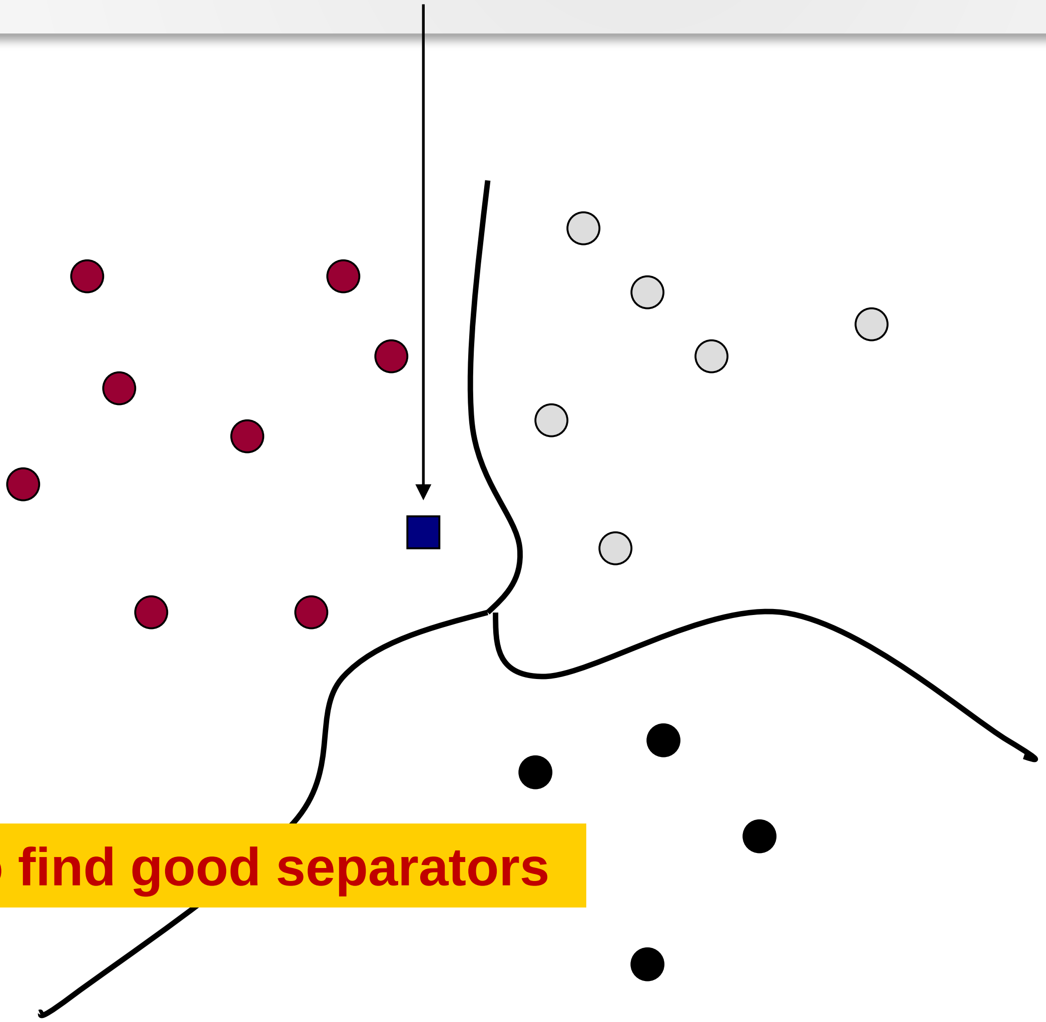


● Government

○ Science

● Arts

Test Document



Is this similarity hypothesis true in general?

● Government

○ Science

● Arts

how to find good separators

Classification Examples: Rocchio Method

- Use standard ***tf-idf*** weighted vectors to represent text documents
- For training documents in each category, compute a prototype vector by summing the vectors of the training documents in the category.
- Prototype = centroid of members of class

$$\vec{\mu}(c) = \frac{1}{|D_c|} \sum_{d \in D_c} \vec{v}(d)$$

where D_c is the set of all documents that belong to class c and $v(d)$ is the vector space representation of d.

- Assign test documents to the class with the closest prototype vector (centroid) based on distance metric for example cosine similarity.

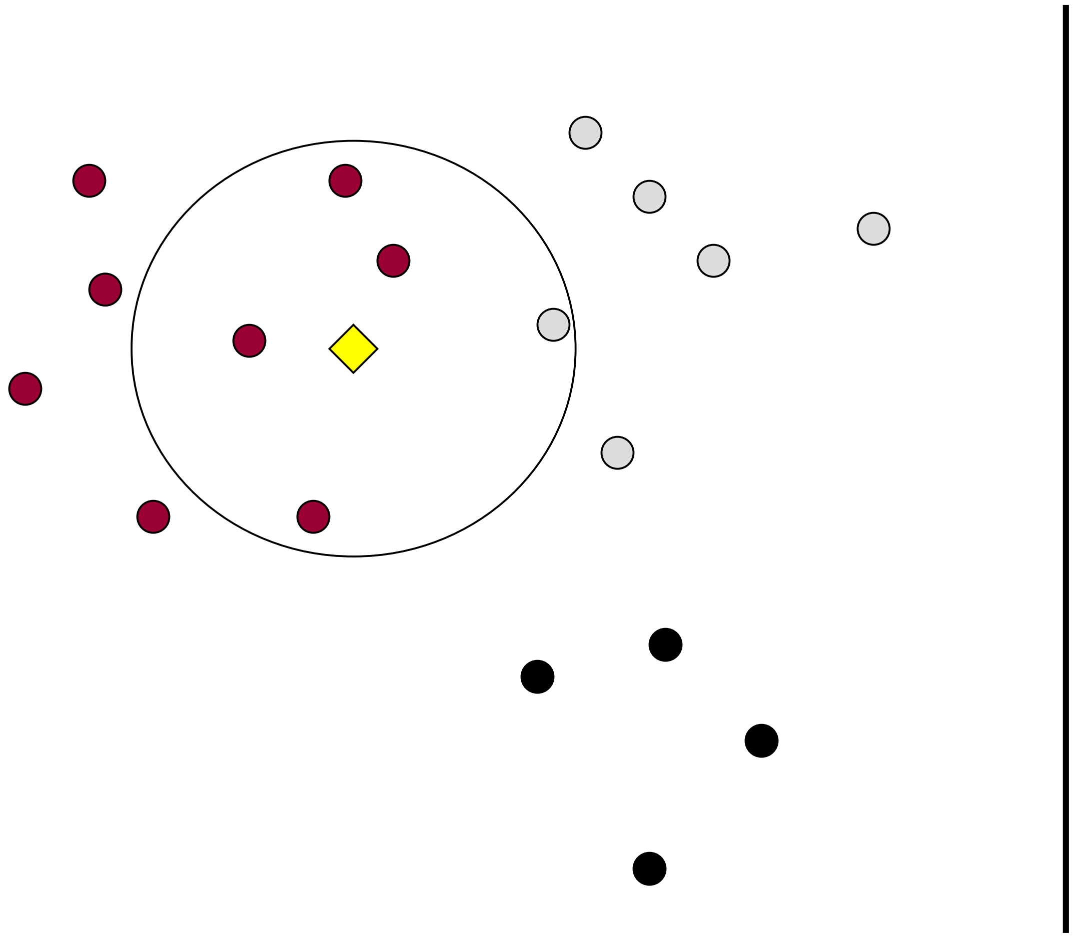
Rocchio forms a simple representation for each class: the centroid Classification is based on similarity (distance) to the prototype/centroid. It does not guarantee that classifications are consistent with the given training data. It is little used outside text classification.

Classification Examples: k Nearest Neighbor (kNN)

To classify a document *d* into class *c*:

1. Define *k*-neighborhood *N* as *k* nearest neighbors of *d*
2. Count number of documents *i* in N that belong to *c*
3. Estimate probability $P(c|d)$ as i/k
4. Choose as class $argmax_c\, P(c|d)$ [= **majority class**]

Classification Examples:
k Nearest Neighbor (k=5, 5NN)



P(science| ◇)?

● Government

○ Science

● Arts

Classification Examples: kNN Learning Algorithm

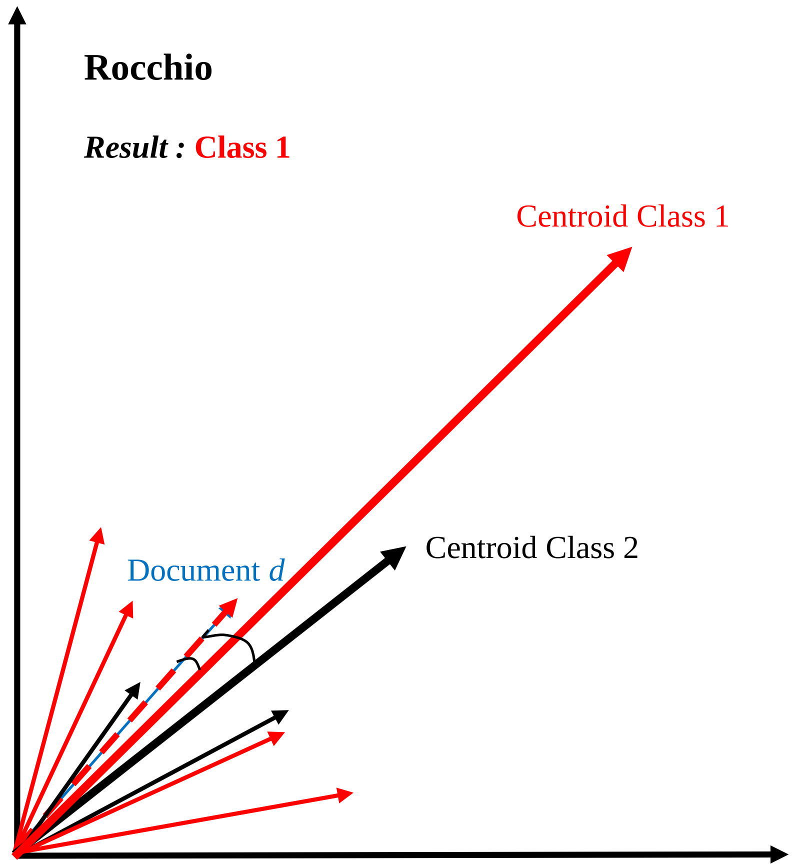
- Learning is just storing the representations of the training examples in *D*.
- Testing instance *x* (under 1NN):
  1. Compute similarity between *x* and all examples in *D*.
  2. Assign *x* the category of the most similar example in *D*.

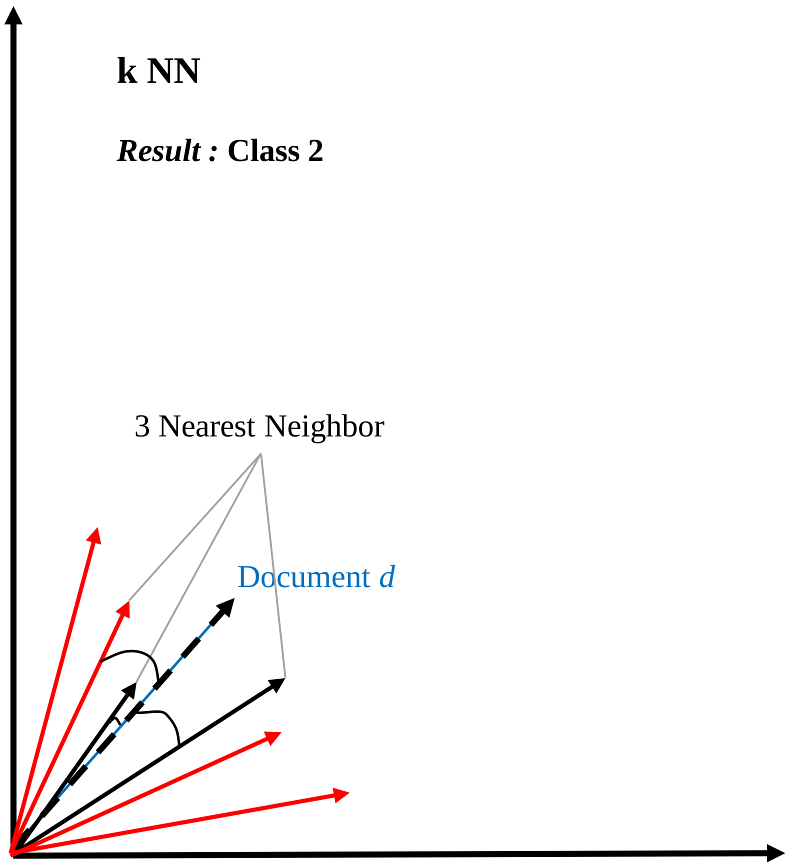
- Does not explicitly compute a generalization or category prototypes (like centroids).

- Also called:
  - Case-based learning
  - Memory-based learning
  - Lazy learning
- Rationale of kNN: neighborhood hypothesis

Classification Examples: kNN vs. Rocchio

Nearest Neighbor tends to handle polymorphic categories better than Rocchio/Naïve Bayes.





Probabilistic in IR: Bayesian Methods

- Learning and classification methods based on probability theory.
- Bayes theorem plays a critical role in probabilistic learning and classification. It describes the **calculation of conditional probabilities** in probability theory.
- Builds a generative model that approximates how data is produced
- Uses **prior probability** of each category given no information about an item.
- Categorization produces a **posterior probability** distribution over the possible categories given a description of an item.
- **Bayes' Rule for text classification**
  For a document *d* and a class *c*

$$P(c,d) = P(c \cap d) = P(c \mid d)P(d) = P(d \mid c)P(c)$$

Prior

$$P(c \mid d) = \frac{P(d \mid c)P(c)}{P(d)}$$

Posterior

(read: the conditional probability of class *c* given document *d* = ...)

See also introductory example: http://en.wikipedia.org/wiki/Bayes'_theorem

Naïve Bayes Classifiers

Task:

Classify a new instance d based on a tuple of attribute values $d = \left(x_1, x_2, \ldots, x_n\right)$ into one of the classes $c_j \in C$

$$c_{MAP} = \underset{c_j \in C}{\boldsymbol{argmax}}\, P(c_j \mid x_1, x_2, \ldots, x_n) = \underset{c_j \in C}{\boldsymbol{argmax}}\, \frac{P(x_1, x_2, \ldots, x_n \mid c_j) P(c_j)}{P(x_1, x_2, \ldots, x_n)}$$

$$c_{MAP} = \underset{c_j \in C}{\boldsymbol{argmax}}\, P(x_1, x_2, \ldots, x_n \mid c_j) P(c_j)$$

MAP is "maximum a posteriori" = most likely class

Conditional Independence Assumption:

features detect term presence and are independent of each other given the class:

$$P(x_1, \ldots, x_n \mid c) = P(x_1 \mid c) \bullet P(x_2 \mid c) \bullet \cdots \bullet P(x_n \mid c)$$

$$\boxed{c_{MAP} = \boldsymbol{argmax}_c P(c) \prod_i P(x_i \mid c)}$$