Institut für Informationssysteme
Technische Universität Braunschweig

Information Retrieval and Web Search Engines

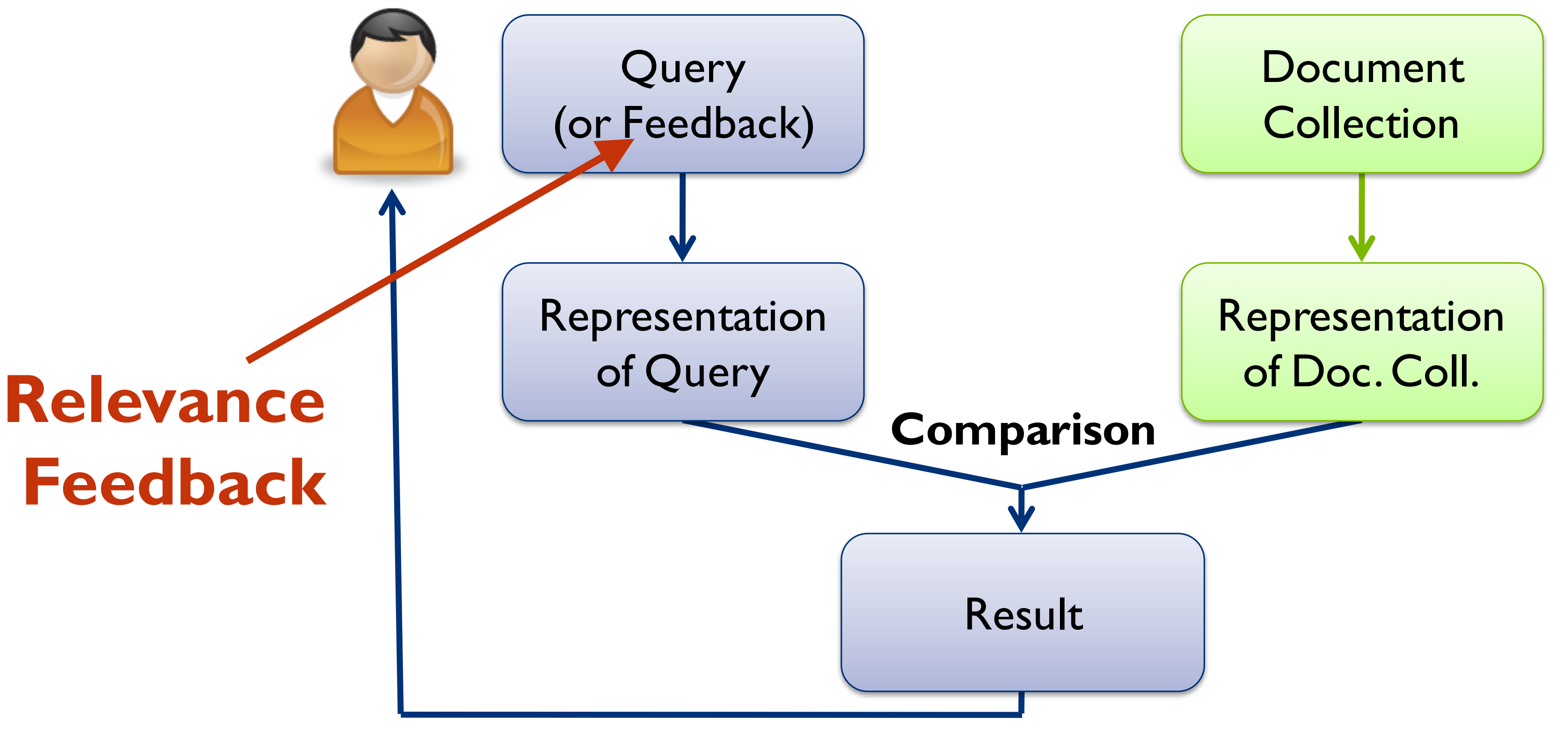
Lecture 8: Feedback and Classification

Wolf-Tilo Balke
Muhammad Usman

Institut für Informationssysteme
Technische Universität Braunschweig

Relevance Feedback

- Remember the **query process** from the first lecture:

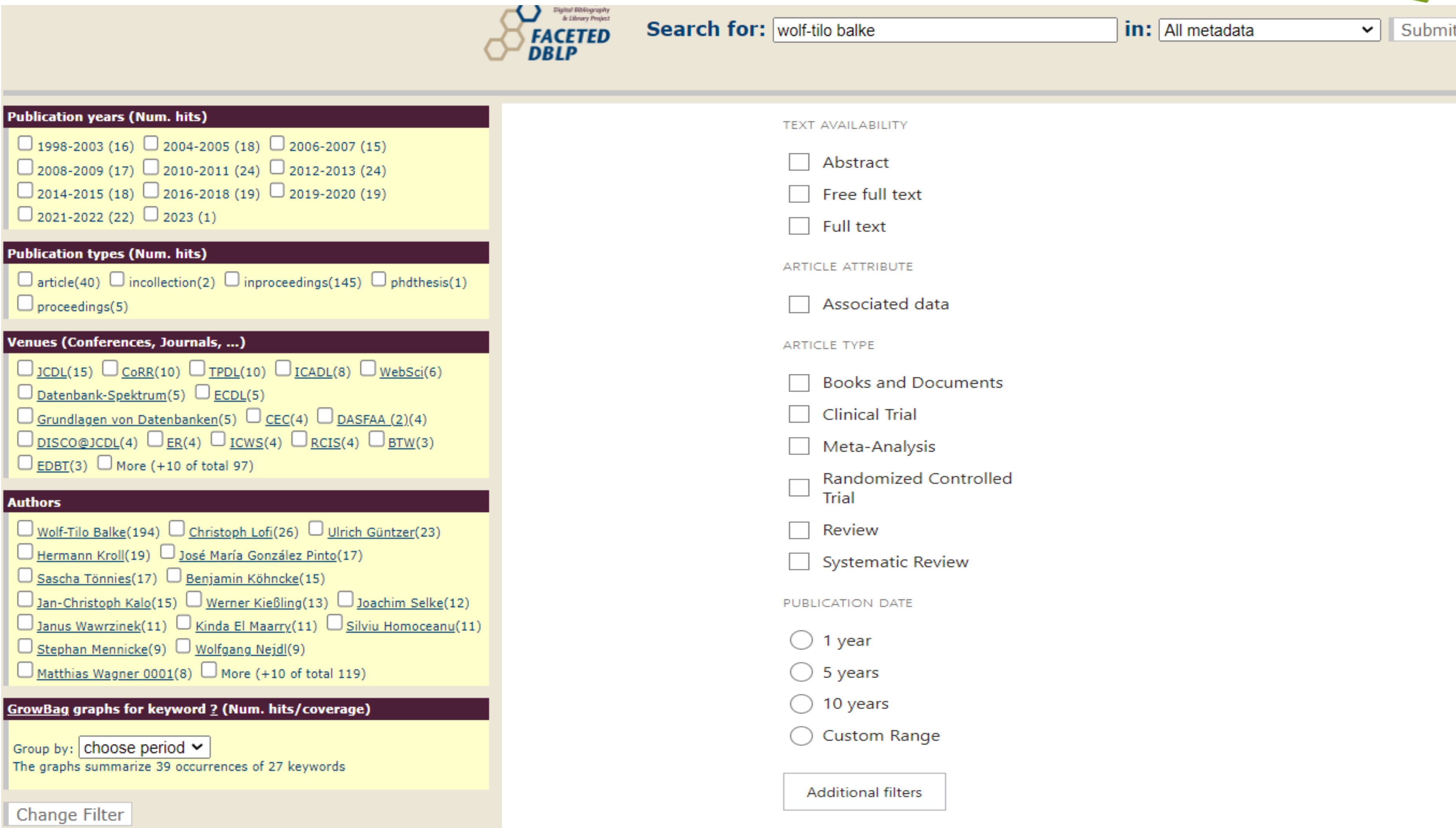


Result Improvement

- There are four main approaches to **result improvement:**
  - Manual modification of query (query refinement)
  - Browsing / "Find similar pages"
  - Faceted Search
  - Relevance feedback (RF)
- Manual modification requires **active user engagement**
- Browsing requires a **"good" clustering,** which is hard
- Relevance feedback is **much easier** to use
- Today, we consider two examples of relevance feedback:
  - **RF in probabilistic retrieval (BIR)**
  - **RF in vector space retrieval**

Faceted search

Detour



https://dblp.l3s.de/

https://pubmed.ncbi.nlm.nih.gov/

Implicit Relevance Feedback

Observing user behavior during normal interaction

- Eye tracking
  - Pupil dilation, eye fixation,…
- Mouse movements
- Reading time
  - Spend more time on relevant results
- User's history and queries.
- Clicks in result list
  - Click on third result but no click on first or second result implies that the first and second result are not relevant

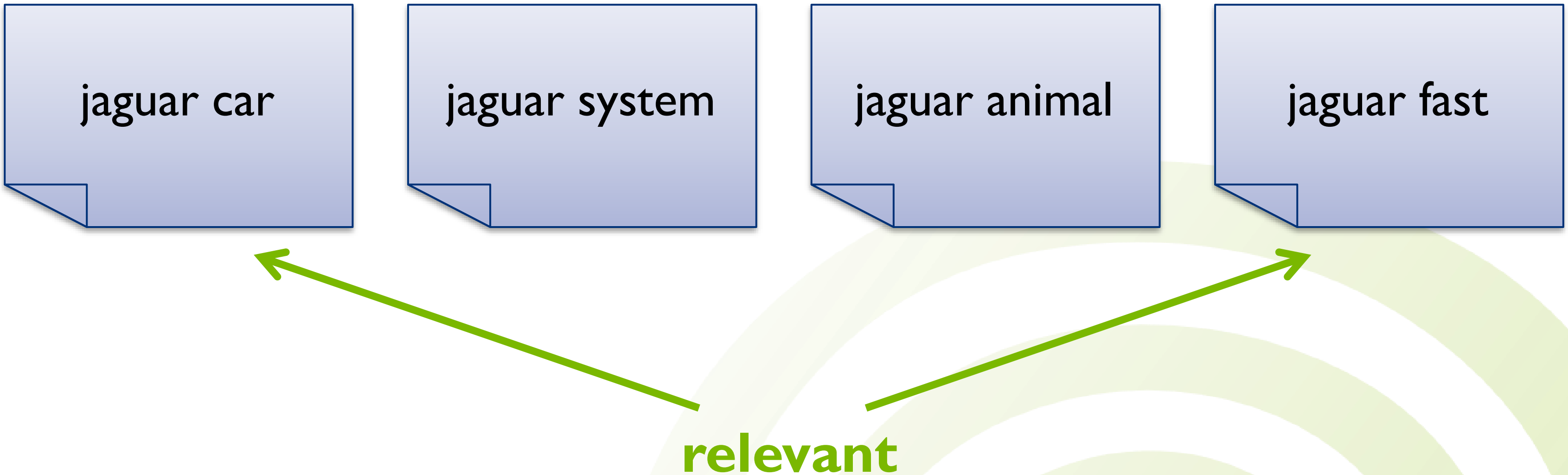
RF in Probabilistic Retrieval

- Remember the BIR retrieval model
  - We had to estimate $\Pr(D_i = 1 \mid D \in R_q)$:
    How many relevant documents contain term *i*?
  - We estimated it using **heuristics:** Choose 0.9!
- Better estimation: Exploit user feedback!
  - Show the user the current retrieval result (with 0.9 estimation)
  - Let him/her **label the relevant ones**
  - Determine the proportion of relevant documents containing term *i* by **counting**
- Use the new estimation to return a better result set
  - This process can be **repeated…**

RF in Probabilistic Retrieval

Example:

Query = “jaguar”



What’s $\Pr(D_{\text{car}} = 1 \mid D \in R_q)$?

$\rightarrow 1/2$

Pseudo Relevance Feedback

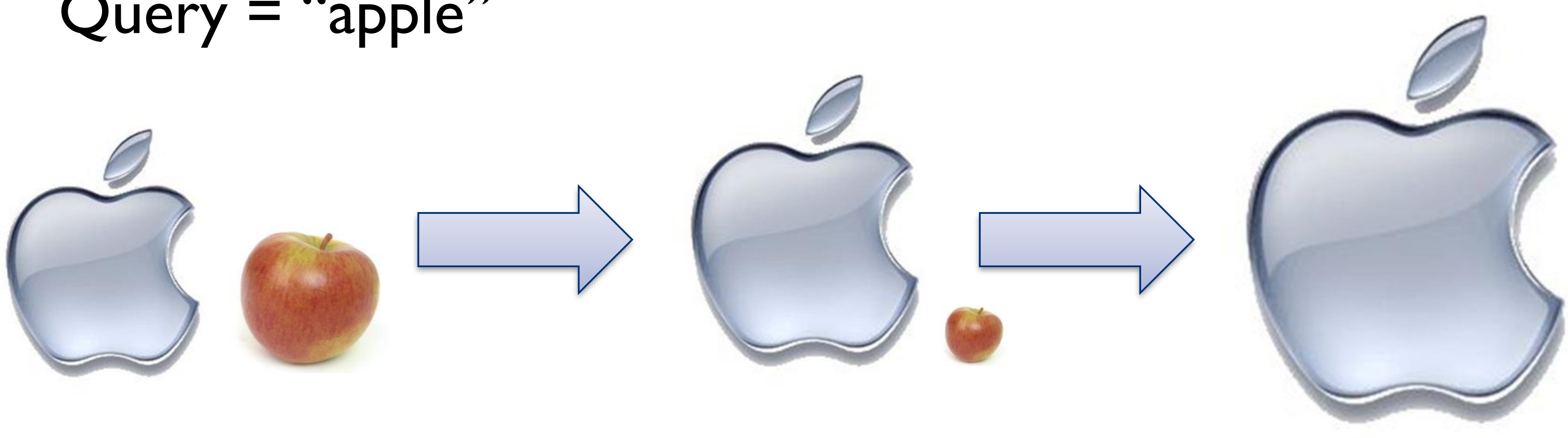
- Relevance feedback without asking the user? YES!
- The “manual” part of relevance feedback can be automated

- **Pseudo Relevance Feedback:**
  - Generate a result list for the user’s query
  - **Assumption: “The top *k* documents are relevant!”**
    - Usually true if *k* is small
  - Use this assumption for relevance feedback
  - **Repeat** this several times…

Pseudo Relevance Feedback

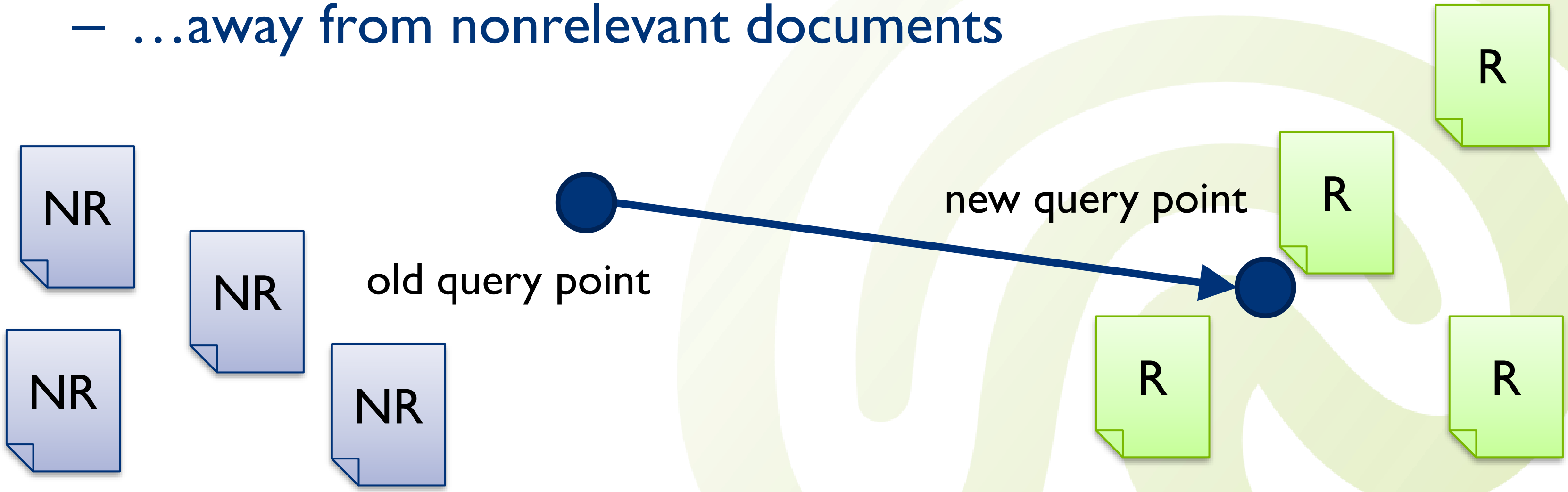
- **Pros:**
  - Works well on average
- **Cons:**
  - Can go horribly wrong for some queries: **Topic drift!**

- **Example of topic drift in pseudo RF:**
  Query = "apple"



RF in the Vector Space Model

- In the **vector space model,** relevance feedback is classically done using **Rocchio's algorithm** (Rocchio, 1971)
- **Idea:**
  Move the query point…
  - …into the direction of relevant documents, and
  - …away from nonrelevant documents



Rocchio's Algorithm

- **Theory:**
  - **The new query should…**
    - **…maximize cosine similarity to all relevant documents**
    - **…minimize cosine similarity to all nonrelevant documents**
  - Let C be the set of documents **returned** to the user
  - Let $C_+ \subseteq C$ be the set of documents rated as **relevant**
  - Let $C_- \subseteq C$ be the set of documents rated as **nonrelevant**
  - **Note:** $C_+ \cup C_- \subsetneq C$ could be true
  - **Task: Find the query point *q* that maximizes**

$$\frac{1}{|C_+|}\sum_{d \in C_+} \frac{q \cdot d}{\|q\| \cdot \|d\|} - \frac{1}{|C_-|}\sum_{d \in C_-} \frac{q \cdot d}{\|q\| \cdot \|d\|}$$

Cosine similarity

Rocchio's Algorithm

- To keep things simple, assume that both the query and all documents are **unit vectors**
  - Vector length does not really matter with cosine similarity
- Then the problem becomes:
  Maximize (in *q*)

$$\frac{1}{|C_+|} \sum_{d \in C_+} \sum_{i=1}^{m} q_i d_i - \frac{1}{|C_-|} \sum_{d \in C_-} \sum_{i=1}^{m} q_i d_i$$

  subject to |*q*| = 1

- This optimization problem can be solved using the method of **Lagrange multipliers**

Rocchio's Algorithm

- Maximize (in *q*)

$$\frac{1}{|C_+|}\sum_{d \in C_+}\sum_{i=1}^{m} q_i d_i - \frac{1}{|C_-|}\sum_{d \in C_-}\sum_{i=1}^{m} q_i d_i$$

  subject to $|q| = 1$

- **Observation** underlying Lagrange multipliers:
  Any maximum of the following expression (in *q*, *λ*) yields a maximum of the original expression:

$$\frac{1}{|C_+|}\sum_{d \in C_+}\sum_{i=1}^{m} q_i d_i - \frac{1}{|C_-|}\sum_{d \in C_-}\sum_{i=1}^{m} q_i d_i - \lambda\left(\sum_{i=1}^{m} q_i^2 - 1\right)$$

- $|q| = 1$ is enforced, since otherwise no maximum exists

Rocchio's Algorithm

$$\frac{1}{|C_+|}\sum_{d\in C_+}\sum_{i=1}^{m} q_i d_i - \frac{1}{|C_-|}\sum_{d\in C_-}\sum_{i=1}^{m} q_i d_i - \lambda\left(\sum_{i=1}^{m} q_i^2 - 1\right)$$

- How to find the maximum of this expression?
  Equate all **partial derivatives** (wrt. $q_1, \ldots, q_m, \lambda$) to zero!
  - Partial derivative with respect to q_j:

$$\frac{1}{|C_+|}\sum_{d\in C_+} d_j - \frac{1}{|C_-|}\sum_{d\in C_-} d_j - 2\lambda q_j \stackrel{!}{=} 0$$

  - Partial derivative with respect to λ:

$$1 - \sum_{i=1}^{m} q_i^2 \stackrel{!}{=} 0$$

Rocchio's Algorithm

$$\frac{1}{|C_+|}\sum_{d \in C_+} d_j - \frac{1}{|C_-|}\sum_{d \in C_-} d_j - 2\lambda q_j \stackrel{!}{=} 0 \qquad 1 - \sum_{i=1}^{m} q_i^2 \stackrel{!}{=} 0$$

- The first equation gives:

$$q \stackrel{!}{=} \frac{1}{2\lambda}\left(\frac{1}{|C_+|}\sum_{d \in C_+} d - \frac{1}{|C_-|}\sum_{d \in C_-} d\right)$$

  - Note that all possible choices for q only differ in their length
- The second equation just expresses the "length 1" constraint
  - Therefore, the choice of q having length 1 is the right one

Rocchio's Algorithm

- We arrive at:

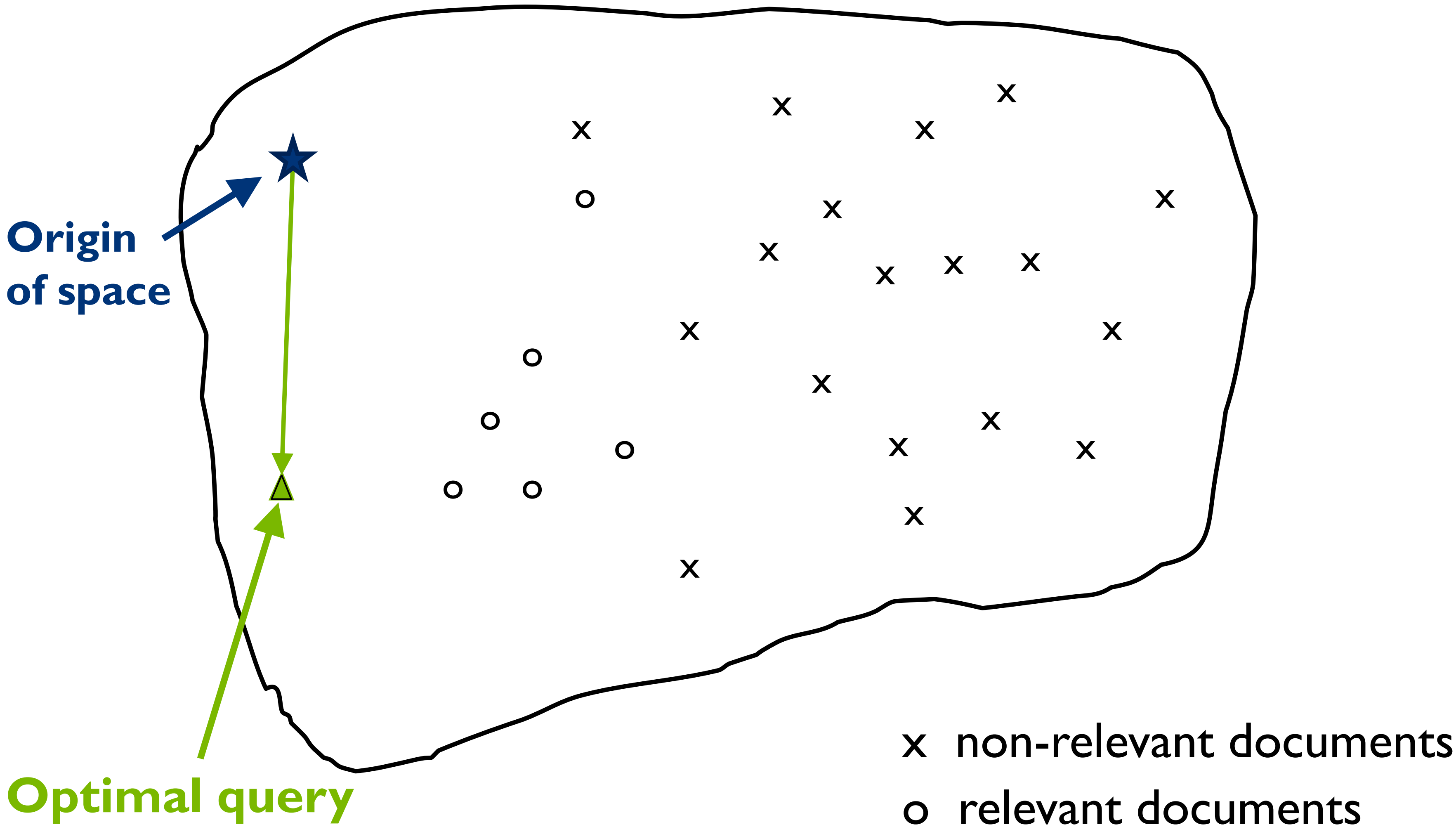
$$q_{\text{opt}}(\lambda) = \frac{1}{2\lambda}\left(\frac{1}{|C_+|}\sum_{d \in C_+} d - \frac{1}{|C_-|}\sum_{d \in C_-} d\right)$$

- Because of the constraint $|q| = 1$, the optimal solution points in the same direction as $q_{\text{opt}}(\lambda)$ but has unit length:

$$q_{\text{opt}} = \frac{q_{\text{opt}}(\lambda)}{\|q_{\text{opt}}(\lambda)\|}$$

- Note that q_{opt} is a scaled version of the **difference vector between C_+'s centroid and C_-'s centroid**

Rocchio's Algorithm



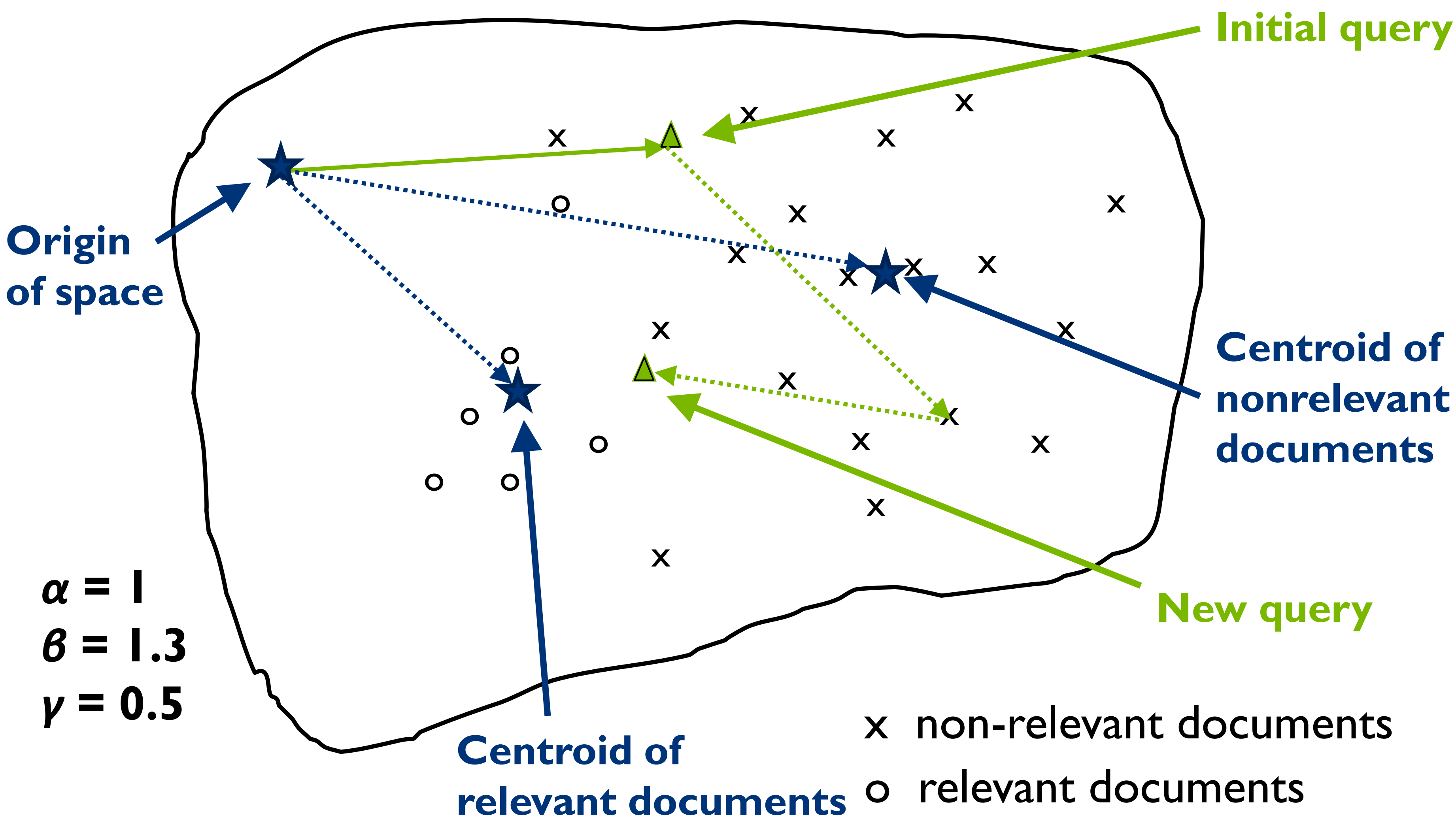
Rocchio's Algorithm

- **Problems:**
  - The user's judgments are biased by the initial result set
  - We cannot trust the user's judgments ultimately
- Therefore, in practice a modified approach is used
- **Idea: Modify the initial query vector!**

$$q_{opt} = \alpha q_0 + \beta \frac{1}{|C_+|} \sum_{d \in C_+} d - \gamma \frac{1}{|C_-|} \sum_{d \in C_-} d$$

  - q_0: Initial query
  - α, β, γ: Weighting factors

Rocchio's Algorithm



$\alpha = 1$
$\beta = 1.3$
$\gamma = 0.5$

Rocchio's Algorithm

$$q_{opt} = \alpha q_0 + \beta \frac{1}{|C_+|} \sum_{d \in C_+} d - \gamma \frac{1}{|C_-|} \sum_{d \in C_-} d$$

- How to choose α, β, and γ?
  - Only if we have a lot of judged documents, we want β and γ to be larger than α
  - Positive feedback usually is more valuable than negative feedback, so set $\beta > \gamma$
  - Reasonable values might be:
    - $\alpha = 1$
    - $\beta = 0.75$
    - $\gamma = 0.15$

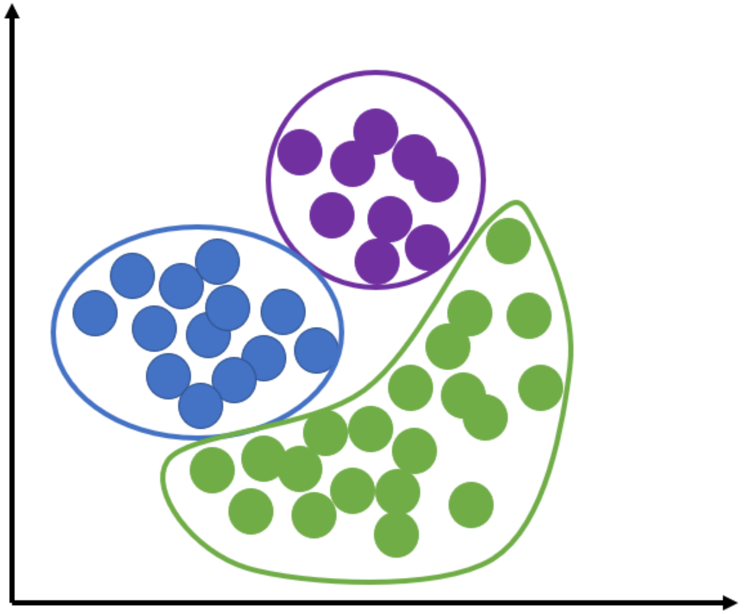
Relevance Feedback: Pros and Cons

- **Pros:**
  - Intuitive approach to automatic query refinement
  - Positive and negative feedback can be exploited
  - Pseudo relevance feedback can enhance result quality without any user interaction

- **Cons:**
  - Requires the initial query to be “good enough”
  - Relies on the cluster hypothesis:
    - Relevant documents are similar
    - Relevant documents are dissimilar from nonrelevant ones
  - Change of results often is hard to explain to the user

Feedback and Classification

1. Relevance Feedback
2. **Document Classification**



What's Document Classification?

- **Task:**
  Automatically assign a given document to one or more **categories,** based on its contents

- **Typical applications in IR:**
  - Spam detection
  - E-mail sorting (friends and family, job, study, …)
  - Detection of sexually explicit content
  - Domain-specific search (e.g. Google Scholar)
  - Language detection
  - Information filtering (standing queries)

Document Classification

- **General task:**
  Learn how to classify new documents
- **Supervised** document classification:
  - Some external mechanism (such as human feedback) provides a correctly classified **training set** of documents (and possibly some **explicit classification rules**)
- **Unsupervised** document classification:
  - **No training set** is available but a sample of unclassified docs
  - Exploits statistical properties of the data (e.g. clustering)
- **Semi-supervised** document classification:
  - A (usually small) training set as well as a set of unclassified documents is available

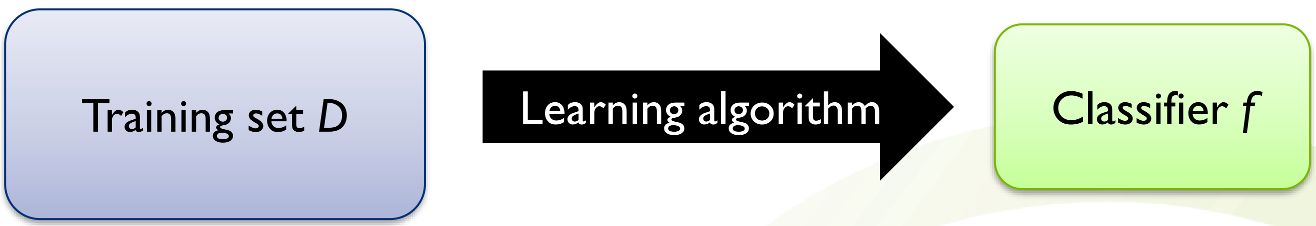
Supervised Classification

- We will focus on **supervised classification** here, which is the most common type

- Some **fundamental definitions:**
  - Let X be the **document space** (e.g. $\mathbb{R}^m$ in vector space retrieval)
  - Let $C = \{c_1, \ldots, c_r\}$ be a fixed set of **classes** (aka categories, labels)
  - Let D be a set of **training pairs** $(d, c) \in X \times C$ (training set)
- **Task** in supervised learning:
  - Using a learning algorithm, find a **classification function** (aka classifier) $\boldsymbol{f : X \rightarrow C}$**,** which maps documents to classes

Supervised Classification

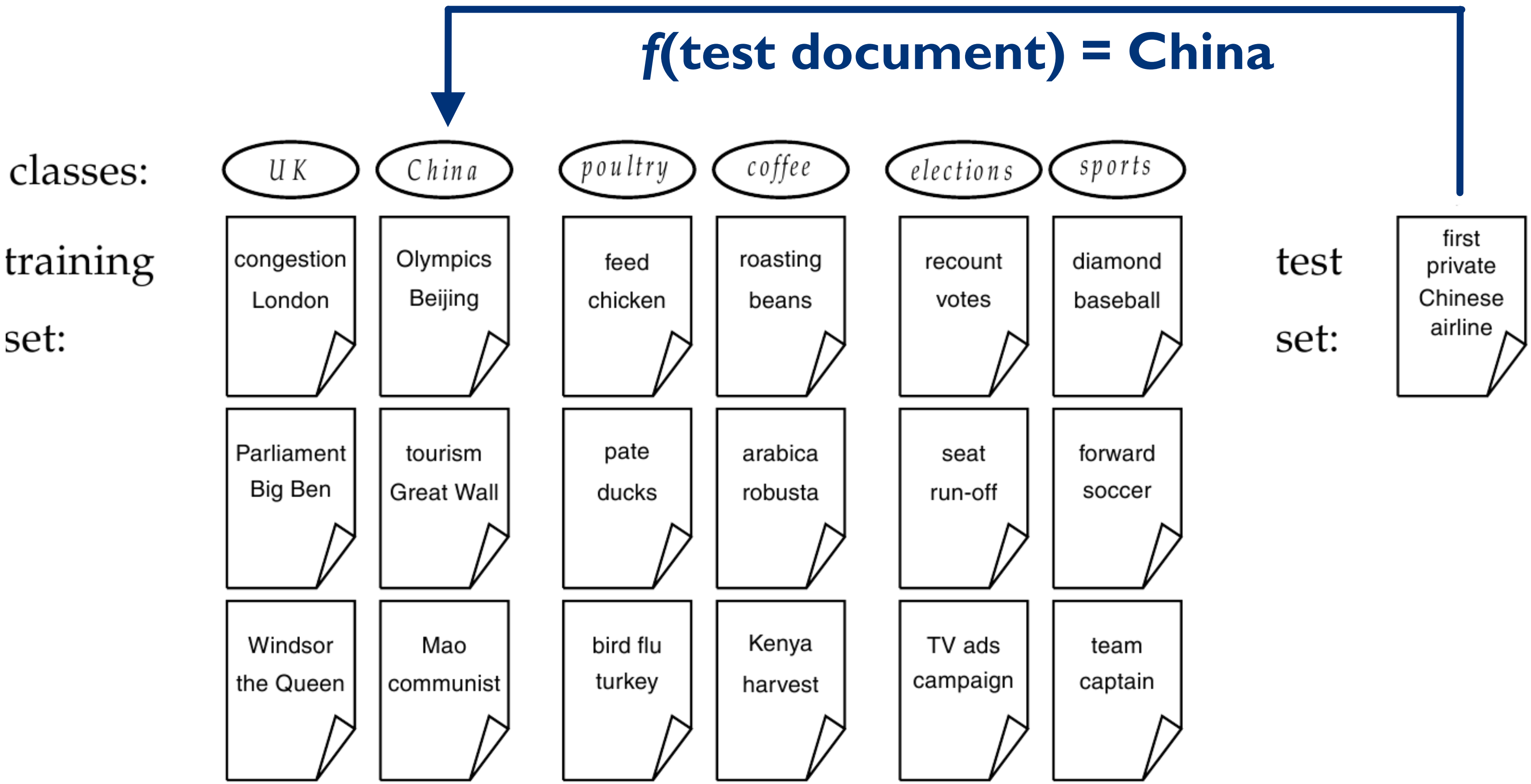
- The **learning algorithm** takes the training set D as input and returns the learned classification function f



- The quality of a learned classification function can be evaluated using a **test set,** which also consists of correctly labeled training pairs $(d, c) \in X \times C$
- Consequently, the training and test set should be similar (or from the same distribution)

Supervised Classification

Example from (Manning *et al.*, 2008):

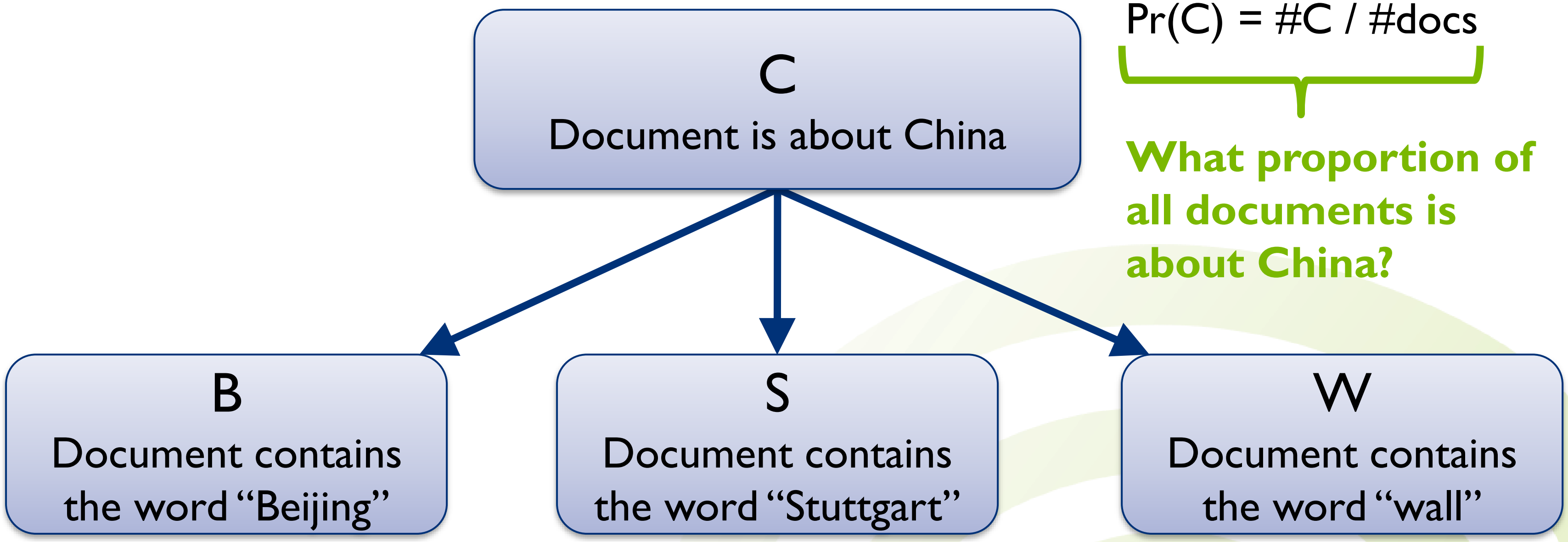


Supervised Classification

- There are several popular learning algorithms, which we will have a look at in this and the next lecture:
  - **Naïve Bayes:**
    A simple probabilistic approach
  - **Rocchio:**
    Classes are represented by centroids
  - **K-nearest neighbors:**
    Look at the nearest neighbors of a new document to determine class membership
  - **Support vector machines:**
    Use hyperplanes to cut the document space into slices; each slice corresponds to a class

Naïve Bayes

A simple **Bayesian network:**



Pr(C) = #C / #docs

What proportion of all documents is about China?

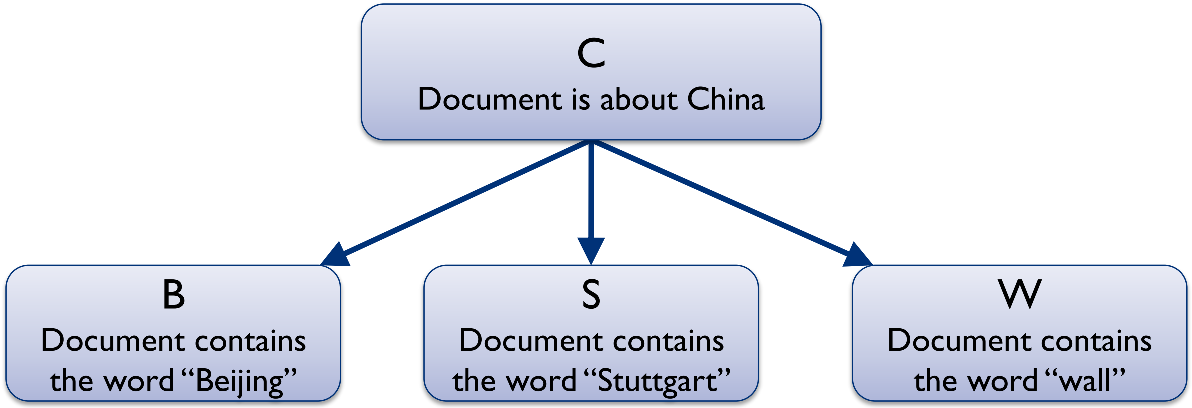
Pr(B) = #B / #docs
Pr(B|C) = #(B and C) / #C
Pr(B|¬C) = #(B and ¬C) / #(¬C)

Pr(S) = …
Pr(S|C) = …
Pr(S|¬C) = …

Pr(W) = …
Pr(W|C) = …
Pr(W|¬C) = …

All these probabilities can be estimated from the training set (possibly using smoothing)!

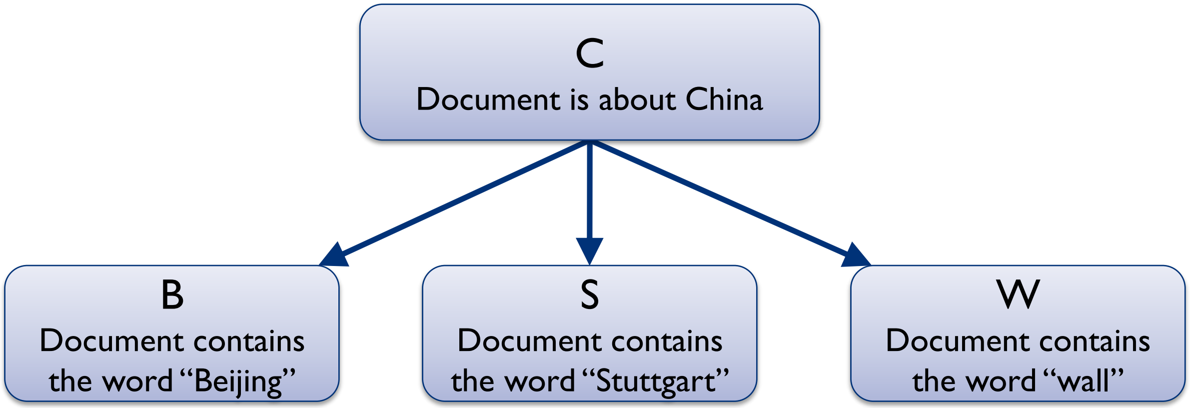
Naïve Bayes



- Classifying a new document:
  - We know whether each of the events B, S, and W occurred
  - **We want to find out whether event C is true**
- This can be done using **Bayes' Theorem:**

$$\Pr(A|B) = \frac{\Pr(A)}{\Pr(B)} \cdot \Pr(B|A)$$

Naïve Bayes



- Assume that the document to be classified contains the word "Beijing" but neither "Stuttgart" nor "wall"
- Consequently, **we want to find Pr(C | B, ¬S, ¬W)**
- **Bayes Theorem** yields:

$$\Pr(C|B, \neg S, \neg W) = \frac{\Pr(C)}{\Pr(B, \neg S, \neg W)} \cdot \Pr(B, \neg S, \neg W|C)$$

Naïve Bayes

$$\Pr(C \mid B, \neg S, \neg W) = \frac{\Pr(C)}{\Pr(B, \neg S, \neg W)} \cdot \Pr(B, \neg S, \neg W \mid C)$$

- In naïve Bayes (sometimes called **idiot Bayes**), **statistical independence** is assumed:

$$\Pr(C \mid B, \neg S, \neg W) = \frac{\Pr(C)}{\Pr(B) \cdot \Pr(\neg S) \cdot \Pr(\neg W)} \cdot \Pr(B \mid C) \cdot \Pr(\neg S \mid C) \cdot \Pr(\neg W \mid C)$$

- **How to classify a new document *d*?**
  - Estimate $\Pr(c \mid d)$, for any class $c \in C$
  - Assign *d* to the class having the highest probability

Naïve Bayes

- **Example** (from Manning *et al.*, 2008; modified):

	DocID	Words in document	Label "China"?
Training set	1	Chinese Beijing Japan	Yes
	2	Shanghai	Yes
	3	Chinese Beijing Tokyo	Yes
	4	Tokyo Japan	No
Test set	5	Chinese Tokyo Japan	?

  – Estimation for Pr(China): 3/4
  – Estimation for Pr(Chinese | China): 2/3
  – Estimation for Pr(Tokyo | China): 1/3
  – Estimation for Pr(Japan | China): 1/3
  – Estimation for Pr(¬Shanghai | China): 2/3
  – Estimation for Pr(¬Beijing | China): 1/3

Naïve Bayes

	DocID	Words in document	Label "China"?
Training set	1	Chinese Beijing Japan	Yes
	2	Shanghai	Yes
	3	Chinese Beijing Tokyo	Yes
	4	Tokyo Japan	No
Test set	5	Chinese Tokyo Japan	?

- Pr(China | Chinese, Tokyo, Japan, ¬Shanghai, ¬Beijing)

$$= 3/4 \cdot \frac{2/3 \cdot 1/3 \cdot 1/3 \cdot 2/3 \cdot 1/3}{1/2 \cdot 1/2 \cdot 1/2 \cdot 3/4 \cdot 1/2} = 64/243 \approx 0.26$$

- Pr(¬China | Chinese, Tokyo, Japan, ¬Shanghai, ¬Beijing)

$$= 1/4 \cdot \frac{0/1 \cdot 1/1 \cdot 1/1 \cdot 1/1 \cdot 1/1}{1/2 \cdot 1/2 \cdot 1/2 \cdot 3/4 \cdot 1/2} = 0$$

- **Since Pr(China | …) > Pr(¬China | …), let's classify doc 5 as "China"**

Naïve Bayes

	DocID	Words in document	Label "China"?
Training set	1	Chinese Beijing Japan	Yes
	2	Shanghai	Yes
	3	Chinese Beijing Tokyo	Yes
	4	Tokyo Japan	No
Test set	5	Chinese Tokyo Japan	?

Pr(China | Chinese, Tokyo, Japan, ¬Shanghai, ¬Beijing) = 0.26

Pr(¬China | Chinese, Tokyo, Japan, ¬Shanghai, ¬Beijing) = 0

- Well, obviously, we need some **smoothing** here…
  - For example, estimate Pr(Chinese | ¬China) by a linear blend of

$$\frac{\#(\text{“Chinese” and “}\neg\text{China”})}{\#(\text{“}\neg\text{China”})} \quad \text{and} \quad \frac{\#(\text{“Chinese”})}{\#\text{documents}}$$

  - From now on, we estimate Pr(Chinese | ¬China) by $0.8 \cdot 0 + 0.2 \cdot 1/2 = 0.1$
    - We do the same for all other probabilities (using weights 0.8 and 0.2)

Naïve Bayes

	DocID	Words in document	Label "China"?
Training set	1	Chinese Beijing Japan	Yes
	2	Shanghai	Yes
	3	Chinese Beijing Tokyo	Yes
	4	Tokyo Japan	No
Test set	5	Chinese Tokyo Japan	?

Using the **smoothed estimates,** we get the following:

- Pr(China | Chinese, Tokyo, Japan, ¬Shanghai, ¬Beijing)

$$= 3/4 \cdot \frac{19/30 \cdot 11/30 \cdot 11/30 \cdot 41/60 \cdot 11/30}{1/2 \cdot 1/2 \cdot 1/2 \cdot 3/4 \cdot 1/2} \approx 0.34$$

- Pr(¬China | Chinese, Tokyo, Japan, ¬Shanghai, ¬Beijing)

$$= 1/4 \cdot \frac{1/10 \cdot 9/10 \cdot 9/10 \cdot 19/20 \cdot 9/10}{1/2 \cdot 1/2 \cdot 1/2 \cdot 3/4 \cdot 1/2} \approx 0.37$$

- **Since Pr(China | …) < Pr(¬China | …), let's classify doc 5 as "¬China"**

Naïve Bayes

	DocID	Words in document	Label "China"?
Training set	1	Chinese Beijing Japan	Yes
	2	Shanghai	Yes
	3	Chinese Beijing Tokyo	Yes
	4	Tokyo Japan	No
Test set	5	Chinese Tokyo Japan	?

$$\Pr(\text{China} \mid \text{Chinese, Tokyo, Japan, } \neg\text{Shanghai, } \neg\text{Beijing}) \approx 0.34$$

$$\Pr(\neg\text{China} \mid \text{Chinese, Tokyo, Japan, } \neg\text{Shanghai, } \neg\text{Beijing}) \approx 0.37$$

- Why don't these probabilities sum up to 1?
  - We assumed independence but it does not hold in the data
  - This is true even without smoothing
  - Example:
    - Pr(Chinese, Beijing | China) = 2/3
    - Pr(Chinese | China) · Pr(Beijing | China) = 2/3 · 2/3 = 4/9 ≠ 2/3
- **Conclusion:** Naïve Bayes is just a heuristic, but an effective one

Naïve Bayes

	DocID	Words in document	Label "China"?
Training set	1	Chinese Beijing Japan	Yes
	2	Shanghai	Yes
	3	Chinese Beijing Tokyo	Yes
	4	Tokyo Japan	No
Test set	5	Chinese Tokyo Japan	?

Typically, when using naïve Bayes, one considers **only positive events,** that is, only probabilities of terms that actually occur in the document:

- Pr(China | Chinese, Tokyo, Japan)

$$= 3/4 \cdot \frac{19/30 \cdot 11/30 \cdot 11/30}{1/2 \cdot 1/2 \cdot 1/2} \approx 0.51$$

- Pr(¬China | Chinese, Tokyo, Japan)

$$= 1/4 \cdot \frac{1/10 \cdot 9/10 \cdot 9/10}{1/2 \cdot 1/2 \cdot 1/2} \approx 0.65$$

- **Since Pr(China | …) < Pr(¬China | …), let's classify doc 5 as "¬China"**

Extensions of Naïve Bayes

- There are many ways to extend naïve Bayes…

- Account for number of occurrences
- Use better smoothing techniques for estimations
- Do not assume independence
- Restrict model to the "most indicative" terms
- Extend model to handle more than two classes
- …

Rocchio

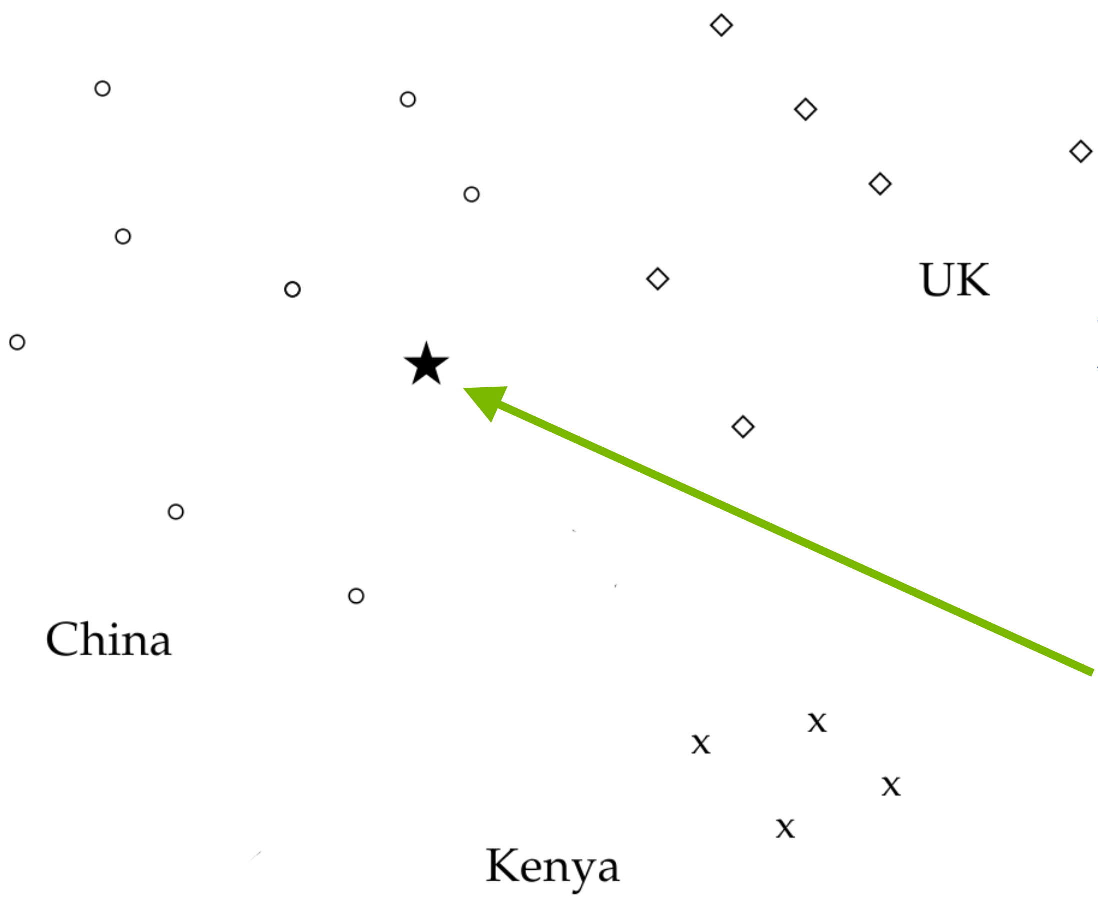
- **Rocchio classification**
  - Requires a **vector space representation** of documents
  - Divides the space into regions centered on **centroids**

- Rocchio relies on the **contiguity hypothesis:**

"Documents in the same class
form a **contiguous region** and
regions of different classes **do not overlap**"

Rocchio

Example (from Manning *et al.*, 2008):

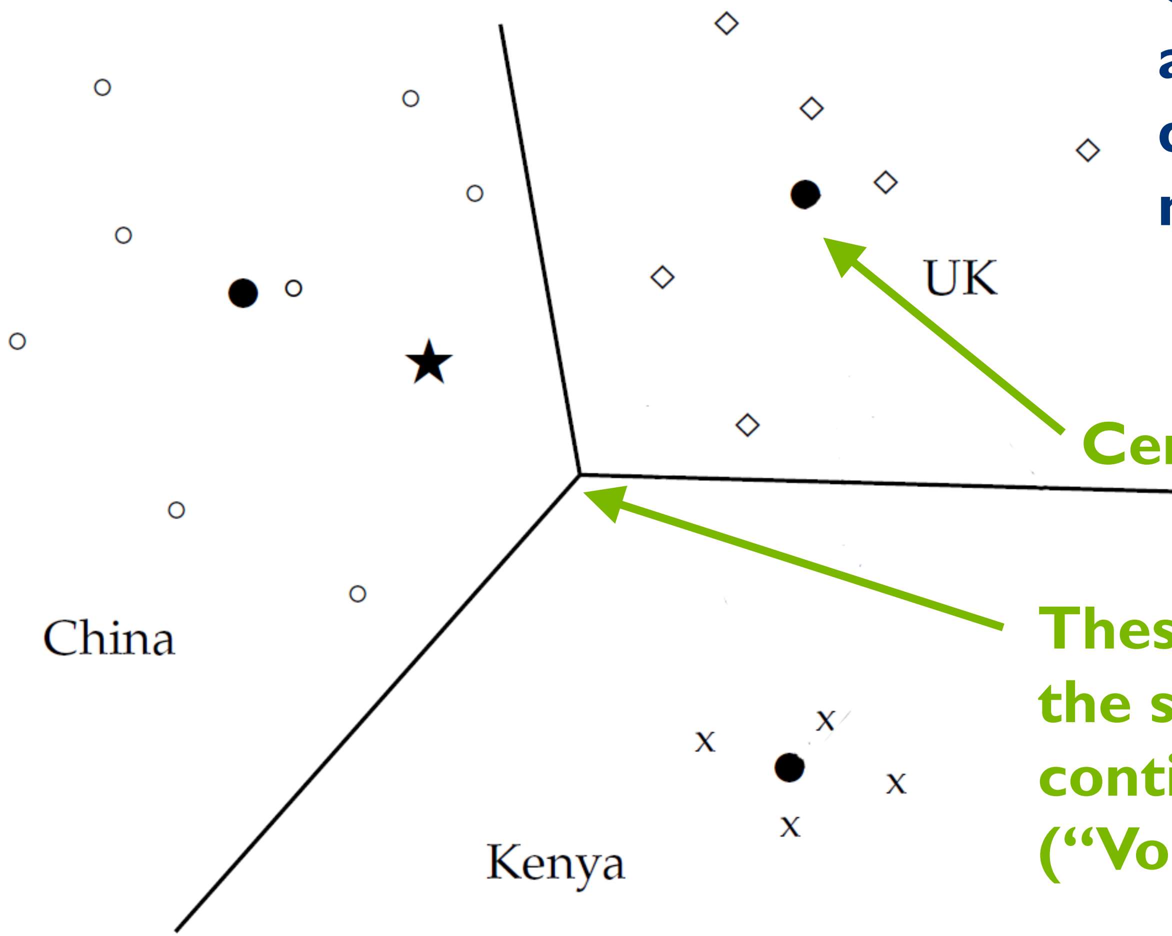


A training set with 3 classes: China, UK, and Kenya

New document to be classified

Rocchio

Rocchio classification:



Compute centroids and assign new documents to their nearest centroid

Centroid of class "UK"

These lines divide the space into contiguous regions ("Voronoi tessellation")

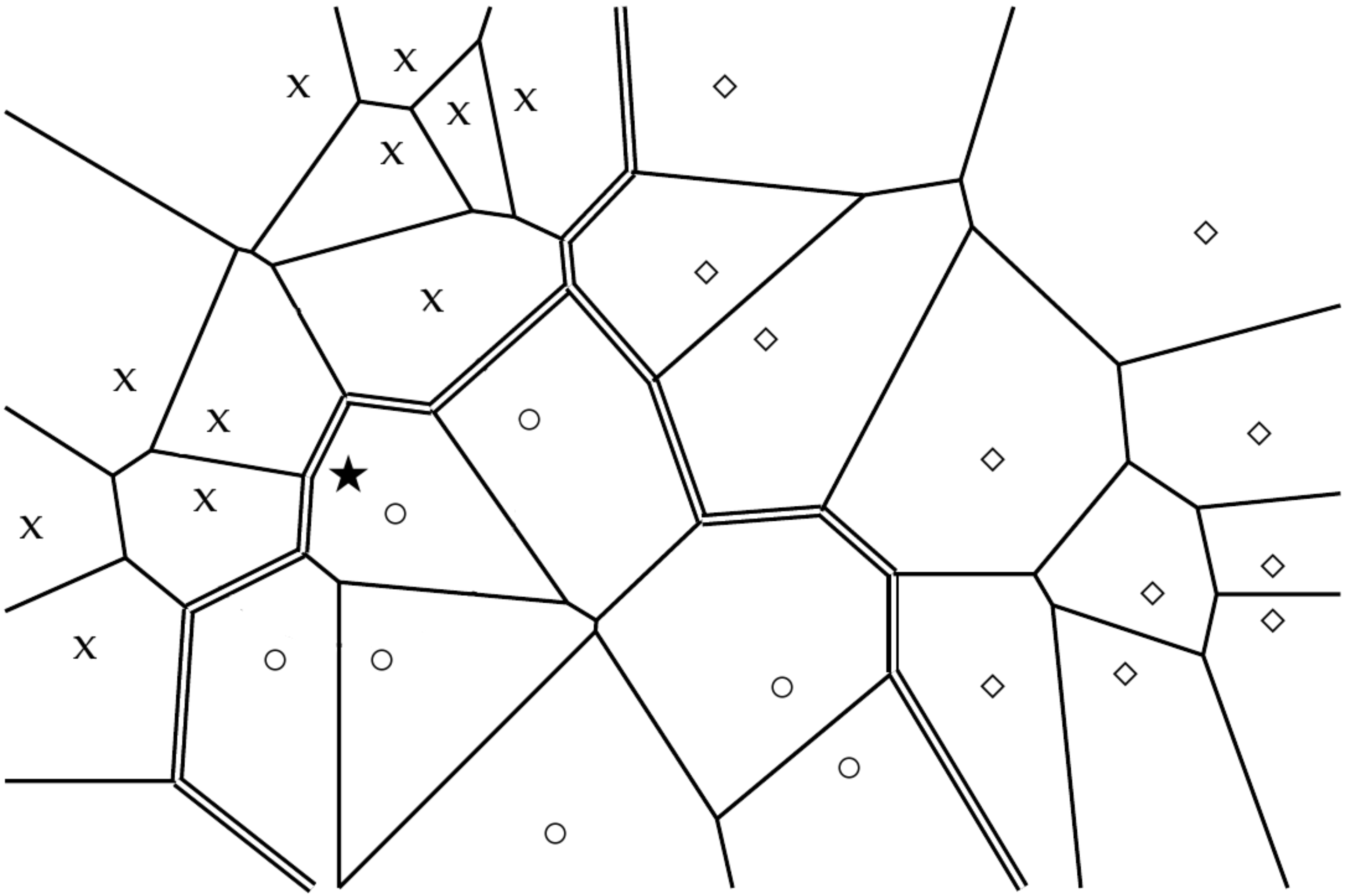
K-Nearest Neighbors

- Unlike Rocchio, **k-nearest neighbor** classification (kNN) uses **class boundaries based on individual documents** (instead of centroids of classes)
- Each new documents gets assigned to the **majority class of its *k* closest neighbors,** where *k* is a parameter
- For $k = 1$, the classes correspond to the **Voronoi tessellation** of the training set
- Clearly, kNN for $k > 1$ is more robust than kNN for $k = 1$

K-Nearest Neighbors

Example (from Manning *et al.*, 2008):

k = 1



K-Nearest Neighbors

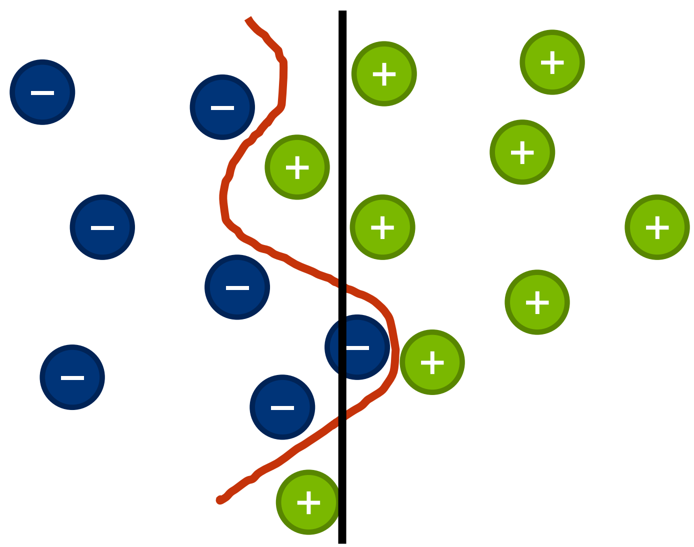
- We can also **weight the "votes"** of the *k* nearest neighbors by their **cosine similarity**
- The **score** of class *c* with respect to some document to be classified *d* then is:

$$\sum_{\substack{d' \in NN_k(d), \\ \text{class}(d')=c}} \frac{d \cdot d'}{\|d\| \cdot \|d'\|}$$

  - $NN_k(d)$: The set of the *k* nearest neighbors of *d* in the training set
  - class(*d'*): The class of training document *d'*
- Every document to be classified gets assigned to the class having the highest score

Support Vector Machines

- Another very important classifier:
  - Support vector machines
  - Highly effective but more complicated to explain
  - Next lecture…



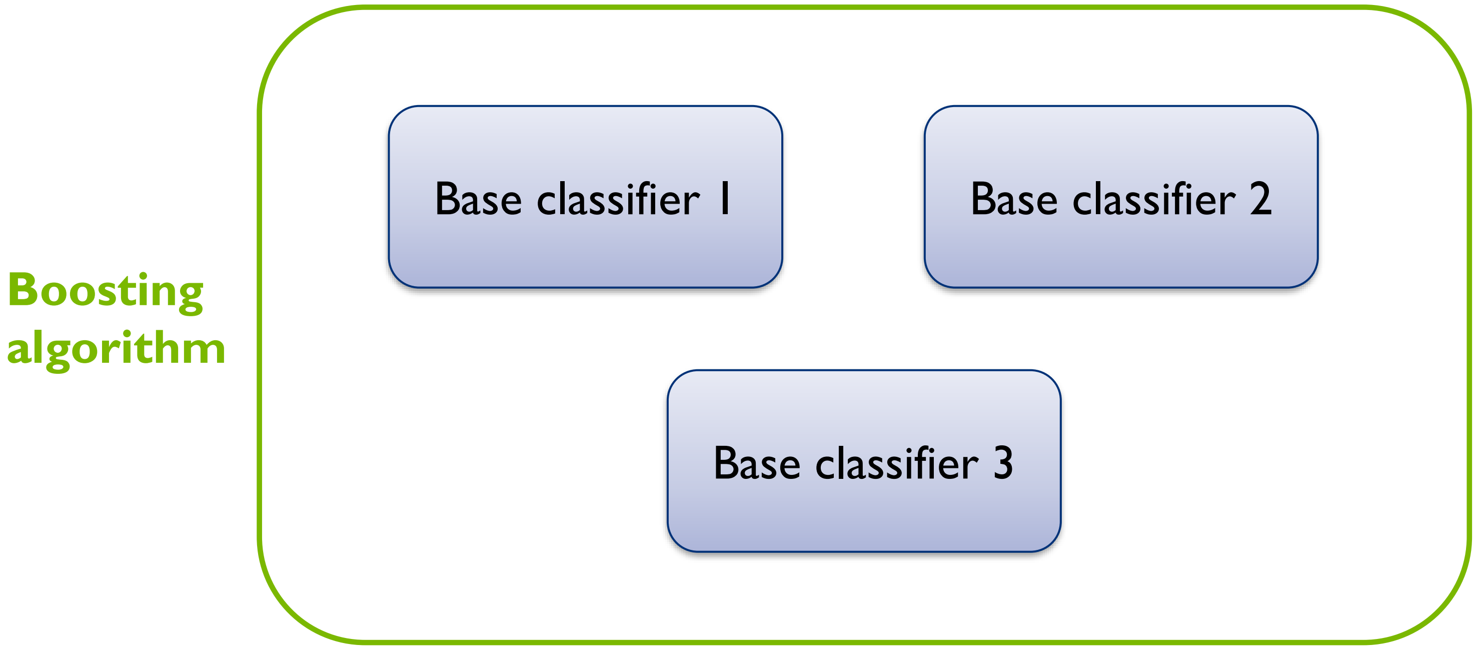
Boosting

- Each different classification algorithm comes with individual strengths and weaknesses
  - “There ain’t no such thing as a free lunch”
- For hard classification problems, the usual classifiers tend to be **weak learners**
  - Weak learner = only slightly better than random guessing

- Question:
  - Can a set of weak learners create a single **strong learner?**
- Answer: **YES!**
  - **Boosting algorithms** do the trick!

Boosting

- Boosting algorithms are **meta-algorithms**
  - Basically, a boosting algorithm is a **blueprint** of how to combine a set of "real" classification algorithms to yield a single combined (and hopefully better) classifier



Boosting

- **Naïve approach** to boosting: **Majority vote!**
  1. Train base classifiers independently on the training set
  2. For each new object to be classified, independently ask each base classifier and return the answer given by the majority

- Problems:
  - Does only work if the majority is right very often
  - Each base algorithm cannot take advantage of its individual strengths
  - Should expert votes have the same weight as any other vote?

Boosting

- Better approach: **Adaptive boosting**
  1. Train the **first base classifier** on the training set
  2. Check which training examples cannot be explained by the first case classifier's underlying model ("errors")
  3. Assign a **weight** to each training example
     - Low weight = Example fits perfectly into the first classifier's model
     - High weight = Example fits hardly into the first classifier's model
  4. Train the **second base classifier** on the weighted training set
     - Fitting training example with high weights is more important than fitting those with low weights
  5. **Reweight** as in step (3)
  6. **Repeat** the steps (4) and (5) for all remaining base classifiers

Boosting

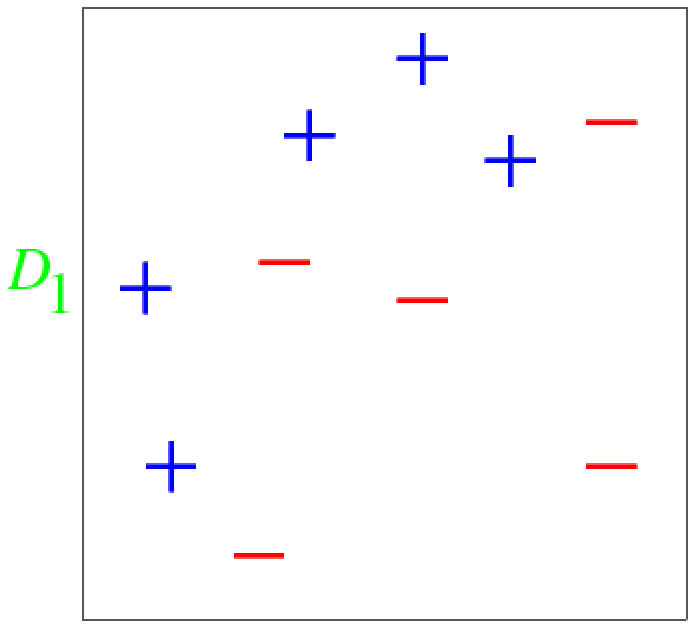
- **Adaptive boosting (continued)**
  - In addition, assign an **importance weight** to each base classifier, depending on how many training examples fit its model
    - High importance if errors occur only on training examples with low weight
    - Low importance if errors occur on training examples with high weight
  - How does the **combined classifier** work?
    1. Classify the new example with each base classifier
    2. Use **majority vote** but weight the individual classifier's answers by their **importance weights;** also incorporate each classifier's confidence if this information is available
  - Typically, the importance weights and the weights of the individual training examples are chosen to be **balanced,** such that **the weighted majority now is right very often**

Boosting

- Why is adaptive boosting better than "pure" majority vote?
  - Later weak learners focus more on those training examples previous weak learners had problems with
  - Individual weaknesses can be compensated
  - Individual strengths can be exploited

Boosting: Example

- Toy example:

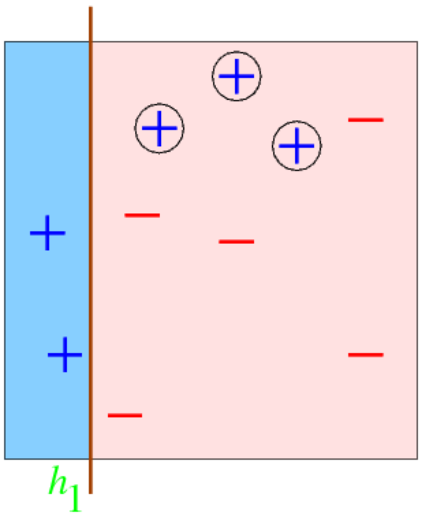


Taken from Freund/Schapire: A Tutorial on Boosting

Boosting: Example

Detour

- Round 1:

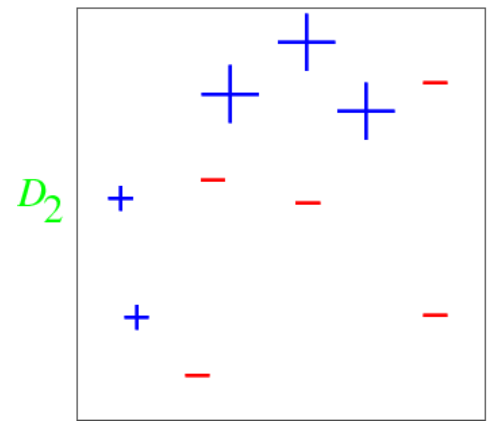


Model of classifier 1

$\varepsilon_1 = 0.30$

$\alpha_1 = 0.42$

$$\alpha \equiv 0.5 \times ln\left|\frac{1-err}{err}\right|$$

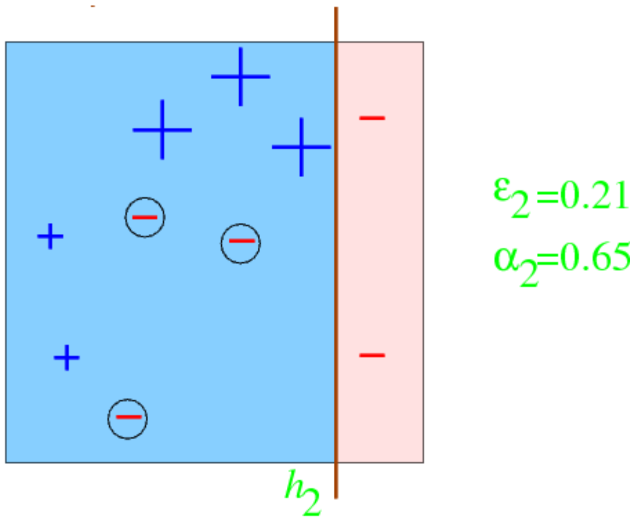


Reweighted training data

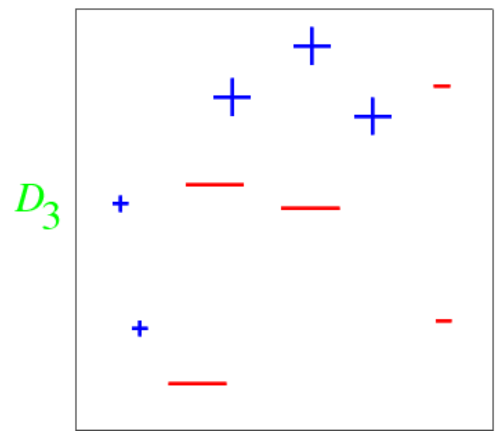
Taken from Freund/Schapire: A Tutorial on Boosting

Boosting: Example

- Round 2:



Model of classifier 2



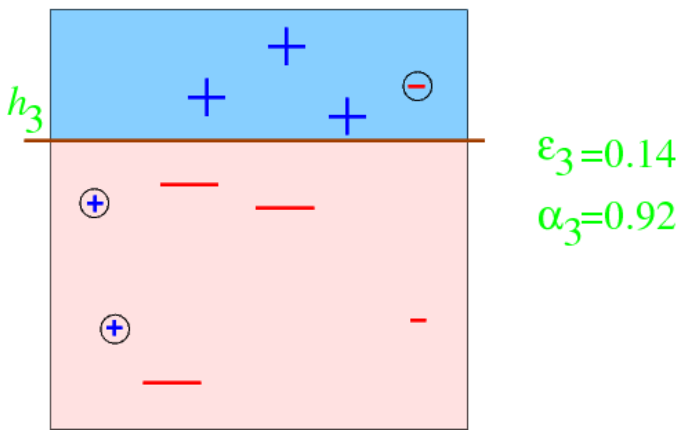
Reweighted training data

Taken from Freund/Schapire: A Tutorial on Boosting

Boosting: Example

Detour

- Round 3:



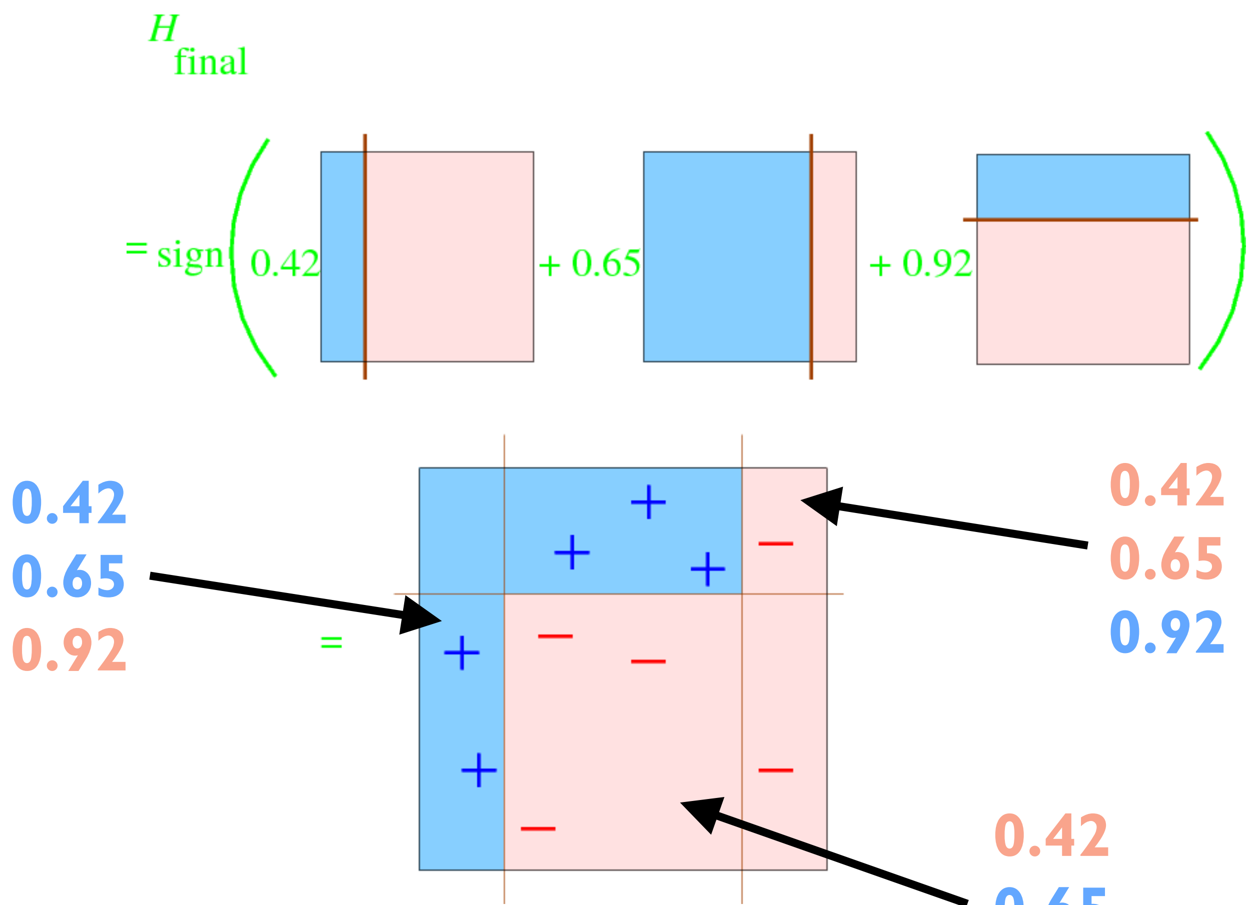
Model of classifier 3

Taken from Freund/Schapire: A Tutorial on Boosting

Boosting: Example

Detour

- Combined classifier:



Taken from Freund/Schapire: A Tutorial on Boosting

Next Lecture

- Support vector machines
- The bias–variance tradeoff (overfitting)