Lecture 8: Feedback and Classification

1. Relevance Feedback
2. Document Classification
• Remember the **query process** from the first lecture:
There are three 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:
  – http://dblp.l3s.de

• Relevance feedback in IR:
  – http://demo.zites.net/search

• Relevance feedback in image search:
  – http://amazon.ece.utexas.edu/~qasim/cires.htm
Implicit Relevance Feedback

• Surf Canyon:
  – http://www.surfcanyon.com

• Other ways to get implicit relevance feedback:
  – Eye tracking
  – Mouse movements
  – 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
• 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**…
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 (2)

• **Pros:**
  – Works well on average

• **Cons:**
  – Can go horribly wrong for some queries: *Topic drift!*

• **Example of topic drift in pseudo RF:**
  Query = “apple”
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
• **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
    \[
    \frac{1}{|C_+|} \sum_{d \in C_+} \frac{q \cdot d}{\|q\| \cdot \|d\|} - \frac{1}{|C_-|} \sum_{d \in C_-} \frac{q \cdot d}{\|q\| \cdot \|d\|}
    \]
Rocchio’s Algorithm (2)

- 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**
Maximize (in $q$) \[
\frac{1}{|C_+|} \sum_{d \in C_+} \sum_{i=1}^{m} q_id_i - \frac{1}{|C_-|} \sum_{d \in C_-} \sum_{i=1}^{m} q_id_i
\] subject to $|q| = 1$

Observation underlying Lagrange multipliers: Any maximum of the following expression (in $q, \lambda$) yields a maximum of the original expression:

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

$|q| = 1$ is enforced, since otherwise no maximum exists.
• 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 \( q_j \):

\[
\frac{1}{|C_+|} \sum_{d \in C_+} d_j - \frac{1}{|C_-|} \sum_{d \in C_-} d_j - 2\lambda q_j = 0
\]

- Partial derivative with respect to \( \lambda \):

\[
1 - \sum_{i=1}^{m} q_i^2 = 0
\]
Rocchio’s Algorithm (5)

\[
\frac{1}{|C_+|} \sum_{d \in C_+} d_j - \frac{1}{|C_-|} \sum_{d \in C_-} d_j - 2\lambda q_j^2 = 0 \quad \text{and} \quad 1 - \sum_{i=1}^{m} q_i^2 = 0
\]

- The first equation gives:

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

- 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
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 \(q_{\text{opt}}\) is a scaled version of the difference vector between \(C_+\)’s centroid and \(C_-\)’s centroid.
Rocchio’s Algorithm (7)

Origin of space

Optimal query

x non-relevant documents
○ relevant documents
• **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
– \( \alpha, \beta, \gamma \): Weighting factors
Rocchio’s Algorithm (9)

Initial query

Centroid of nonrelevant documents

Origin of space

New query

Centroid of relevant documents

$\alpha = 1$

$\beta = 1.3$

$\gamma = 0.5$

x non-relevant documents

o relevant documents
Rocchio’s Algorithm (10)

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

• How to choose \( \alpha \), \( \beta \), and \( \gamma \)?
  – Only if we have a lot of judged documents, we want \( \beta \) and \( \gamma \) to be larger than \( \alpha \)
  – 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
Lecture 8: 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)
• **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
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) $f : X \to C$, which maps documents to classes
• 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 (3)

Example from (Manning et al., 2008):

\[ f(\text{test document}) = \text{China} \]

<table>
<thead>
<tr>
<th>classes:</th>
<th>UK</th>
<th>China</th>
<th>poultry</th>
<th>coffee</th>
<th>elections</th>
<th>sports</th>
</tr>
</thead>
<tbody>
<tr>
<td>training</td>
<td>congestion</td>
<td>Olympics</td>
<td>feed chicken</td>
<td>roasting</td>
<td>recount</td>
<td>diamond</td>
</tr>
<tr>
<td>set:</td>
<td>London</td>
<td>Beijing</td>
<td>chicken</td>
<td>beans</td>
<td>votes</td>
<td>baseball</td>
</tr>
<tr>
<td></td>
<td>Parliament</td>
<td>tourism</td>
<td>pate ducks</td>
<td>arabica</td>
<td>seat</td>
<td>forward</td>
</tr>
<tr>
<td></td>
<td>Big Ben</td>
<td>Great Wall</td>
<td>ducks</td>
<td>robusta</td>
<td>run-off</td>
<td>soccer</td>
</tr>
<tr>
<td></td>
<td>Windsor</td>
<td>Mao</td>
<td>bird flu</td>
<td>Kenya</td>
<td>TV ads</td>
<td>team</td>
</tr>
<tr>
<td></td>
<td>the Queen</td>
<td>communist</td>
<td>turkey</td>
<td>harvest</td>
<td>campaign</td>
<td>captain</td>
</tr>
</tbody>
</table>

| test set: | first private | Chinese      |
|           | airline      |              |
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
A simple **Bayesian network:**

Pr(C) = #C / #docs

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) = …

What proportion of all documents is about China?

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

\[
\Pr(A \mid B) = \frac{\Pr(A)}{\Pr(B)} \cdot \Pr(B \mid A)
\]
• Assume that the document to be classified contains the word “Beijing” but neither “Stuttgart” nor “wall”
• Consequently, we want to find $Pr(C \mid B, \neg S, \neg W)$
• **Bayes Theorem** yields:

$$Pr(C \mid B, \neg S, \neg W) = \frac{Pr(C)}{Pr(B, \neg S, \neg W)} \cdot Pr(B, \neg S, \neg W \mid C)$$
**Naïve Bayes (4)**

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

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

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

- **How to classify a new document** \(d\)?
  - Estimate \(Pr(c | d)\), for any class \(c \in C\)
  - Assign \(d\) to the class having the highest probability
**Naïve Bayes (5)**

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

<table>
<thead>
<tr>
<th>DocID</th>
<th>Words in document</th>
<th>Label “China”?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training set</td>
<td>1  Chinese Beijing Japan</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>2  Shanghai</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>3  Chinese Beijing Tokyo</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>4  Tokyo Japan</td>
<td>No</td>
</tr>
<tr>
<td>Test set</td>
<td>5  Chinese Tokyo Japan</td>
<td>?</td>
</tr>
</tbody>
</table>

- 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 (6)

<table>
<thead>
<tr>
<th>DocID</th>
<th>Words in document</th>
<th>Label “China”?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training set</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Chinese Beijing Japan</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>Shanghai</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>Chinese Beijing Tokyo</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>Tokyo Japan</td>
<td>No</td>
</tr>
<tr>
<td>Test set</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Chinese Tokyo Japan</td>
<td>?</td>
</tr>
</tbody>
</table>

- \( \Pr(\text{China} \mid \text{Chinese, Tokyo, Japan, } \neg\text{Shanghai, } \neg\text{Beijing}) \)
  
  \[
  = \frac{3}{4} \cdot \frac{2/3 \cdot 1/3 \cdot 1/3 \cdot 2/3 \cdot 1/3}{1/2 \cdot 1/2 \cdot 1/2 \cdot 3/4 \cdot 1/2} = \frac{64}{243} \approx 0.26
  \]

- \( \Pr(\neg\text{China} \mid \text{Chinese, Tokyo, Japan, } \neg\text{Shanghai, } \neg\text{Beijing}) \)
  
  \[
  = \frac{1}{4} \cdot \frac{0/1 \cdot 1/1 \cdot 1/1 \cdot 1/1 \cdot 1/1}{1/2 \cdot 1/2 \cdot 1/2 \cdot 3/4 \cdot 1/2} = 0
  \]

- Since \( \Pr(\text{China} \mid \ldots) > \Pr(\neg\text{China} \mid \ldots) \), let’s classify doc 5 as “China”
\textbf{Naïve Bayes (7)}

<table>
<thead>
<tr>
<th>DocID</th>
<th>Words in document</th>
<th>Label “China”?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training set</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Chinese Beijing Japan</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>Shanghai</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>Chinese Beijing Tokyo</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>Tokyo Japan Chinese</td>
<td>No</td>
</tr>
<tr>
<td>Test set</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Chinese Tokyo Japan</td>
<td>?</td>
</tr>
</tbody>
</table>

\[
\Pr(\text{China} \mid \text{Chinese, Tokyo, Japan, } \neg\text{Shanghai, } \neg\text{Beijing}) = 0.26 \\
\Pr(\neg\text{China} \mid \text{Chinese, Tokyo, Japan, } \neg\text{Shanghai, } \neg\text{Beijing}) = 0
\]

• Well, obviously, we need some \textbf{smoothing} here…
  
  – For example, estimate \( \Pr(\text{Chinese} \mid \neg\text{China}) \) by a linear blend of

  \[
  \frac{\#(\text{“Chinese” and “¬China”})}{\#(“¬China”)} \quad \text{and} \quad \frac{\#(\text{“Chinese”})}{\#\text{documents}}
  \]

  – From now on, we estimate \( \Pr(\text{Chinese} \mid \text{China}) \) by \( 0.8 \cdot 0 + 0.2 \cdot 3/4 = 0.15 \)
  
  • We do the same for all other probabilities (using weights 0.8 and 0.2)
Using the smoothed estimates, we get the following:

- \( \Pr(\text{China} \mid \text{Chinese, Tokyo, Japan, ¬Shanghai, ¬Beijing}) \)

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

- \( \Pr(\neg\text{China} \mid \text{Chinese, Tokyo, Japan, ¬Shanghai, ¬Beijing}) \)

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

- Since \( \Pr(\text{China} \mid \ldots) < \Pr(\neg\text{China} \mid \ldots) \), let’s classify doc 5 as “¬China”
Naïve Bayes (9)

<table>
<thead>
<tr>
<th>DocID</th>
<th>Words in document</th>
<th>Label “China”?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training set</td>
<td>1 Chinese Beijing Japan</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>2 Shanghai</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>3 Chinese Beijing Tokyo</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>4 Tokyo Japan</td>
<td>No</td>
</tr>
<tr>
<td>Test set</td>
<td>5 Chinese Tokyo Japan</td>
<td>?</td>
</tr>
</tbody>
</table>

Pr(China | Chinese, Tokyo, Japan, ¬Shanghai, ¬Beijing) ≈ 0.34
Pr(¬China | Chinese, Tokyo, Japan, ¬Shanghai, ¬Beijing) ≈ 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
Typically, when using naïve Bayes, one considers only positive events, that is, only probabilities of terms that actually occur in the document:

- $\Pr(\text{China} | \text{Chinese, Tokyo, Japan})$

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

- $\Pr(\neg \text{China} | \text{Chinese, Tokyo, Japan})$

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

- Since $\Pr(\text{China} | \ldots) < \Pr(\neg \text{China} | \ldots)$, let’s classify doc 5 as “$\neg \text{China}$”
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 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**”
Example (from Manning et al., 2008):

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

New document to be classified
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”)

Information Retrieval and Web Search Engines — Wolf-Tilo Balke and Joachim Selke — Technische Universität Braunschweig
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.
Example (from Manning et al., 2008):

\[ k = 1 \]
• 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:

\[
\sum_{d' \in \text{NN}_k(d), \text{class}(d') = c} \frac{d \cdot d'}{\|d\| \cdot \|d'\|}
\]

  – **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
• Another very important classifier:
  – Support vector machines
  – Highly effective but more complicated to explain
  – Next week …
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 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
• **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?
• **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
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**
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
• Toy example:

Taken from Freund/Schapire: A Tutorial on Boosting
• Round 1:

Model of classifier 1

\[ \epsilon_1 = 0.30 \]
\[ \alpha_1 = 0.42 \]

Reweighted training data

Taken from Freund/Schapire: A Tutorial on Boosting
• Round 2:

![Model of classifier 2](image)

\[ h_2 \]

\[ \varepsilon_2 = 0.21 \]
\[ \alpha_2 = 0.65 \]

![Reweighted training data](image)

\[ D_3 \]

Taken from Freund/Schapire: A Tutorial on Boosting
• Round 3:

Model of classifier 3

Taken from Freund/Schapire: A Tutorial on Boosting
• Combined classifier:

\[ H_{\text{final}} = \text{sign}(0.42 + 0.65 + 0.92) \]

Taken from Freund/Schapire: A Tutorial on Boosting
- Support vector machines
- The bias–variance tradeoff (overfitting)