Machine Learning for Information Discovery

Thorsten Joachims

Cornell University
Department of Computer Science

(Supervised) Machine Learning

GENERAL:

Input:

- training examples
- design space

Training:

- automatically find the solution in design space that works well on the training data

Prediction:

- predict well on new examples

EXAMPLE: Text Retrieval

Input:

- queries with relevance judgments
- parameters of retrieval function

Training:

- find parameters so that many relevant documents are ranked highly

Prediction:

- rank relevant documents high also for new queries

Common Machine Learning Tasks in ID

- **Text Retrieval**
  - provide good rankings for a query
  - use machine learning on relevance judgments to optimize ranking function
- **Text Classification**
  - classify documents by their semantic content
  - use machine learning and classified documents to learn classification rules
- **Information Extraction**
  - learn to extract particular attributes from a document
  - use machine learning to identify where in the text the information is located
- **Topic Detection and Tracking**
  - find and track new topics in a stream of documents

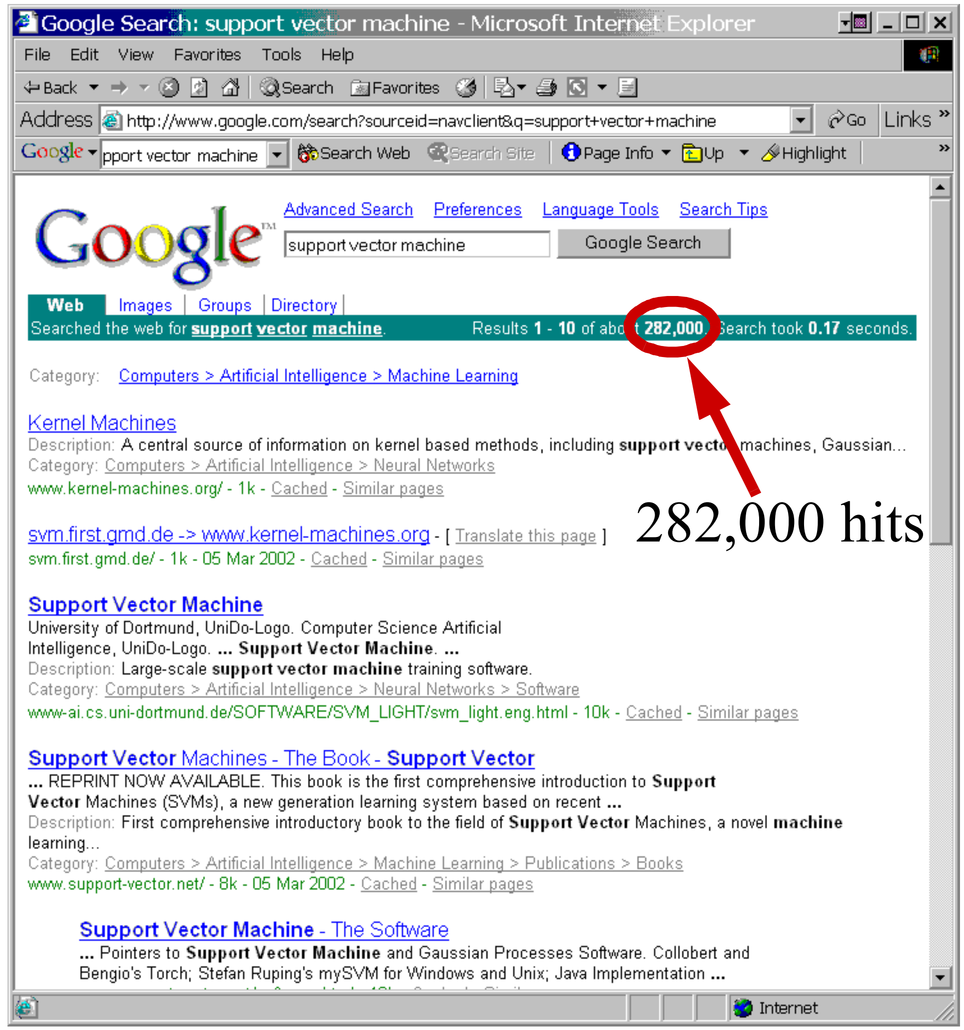
Text Retrieval

Query:

- "Support Vector Machine"

Goal:

- "rank the documents I want high in the list"



Text Classification

E.D. And F. MAN TO BUY INTO HONG KONG FIRM

The U.K. Based commodity house E.D. And F. Man Ltd and Singapores Yeo Hiap Seng Ltd jointly announced that Man will buy a substantial stake in Yeos 71.1 pct held unit, Yeo Hiap Seng Enterprises Ltd. Man will develop the locally listed soft drinks manufacturer into a securities and commodities brokerage arm and will rename the firm Man Pacific (Holdings) Ltd.

About a corporate acquisition?

YES

NO

Information Extraction

Why Use Machine Learning?

Approach 1: Just do everything manually!

- pretty mind numbing
- too expensive (e.g. Reuters 11,000 stories per day, 90 indexers)
- does not scale

Approach 2: Construct automatic rules manually!

- humans are not really good at it (e.g. constructing classification rules)
- no expert is available (e.g. rules for filtering my email)
- its just too expensive to do by hand (e.g. ArXiv classification, personal retrieval functions)

Approach 3: Construct automatic rules via machine learning!

- training data is cheap and plenty (e.g. clickthrough)
- can be done on an (pretty much) arbitrary level of granularity
- works well without expert interventions

Text Classification

E.D. And F. MAN TO BUY INTO HONG KONG FIRM

The U.K. Based commodity house E.D. And F. Man Ltd and Singapores Yeo Hiap Seng Ltd jointly announced that Man will buy a substantial stake in Yeos 71.1 pct held unit, Yeo Hiap Seng Enterprises Ltd. Man will develop the locally listed soft drinks manufacturer into a securities and commodities brokerage arm and will rename the firm Man Pacific (Holdings) Ltd.

About a corporate acquisition?

YES

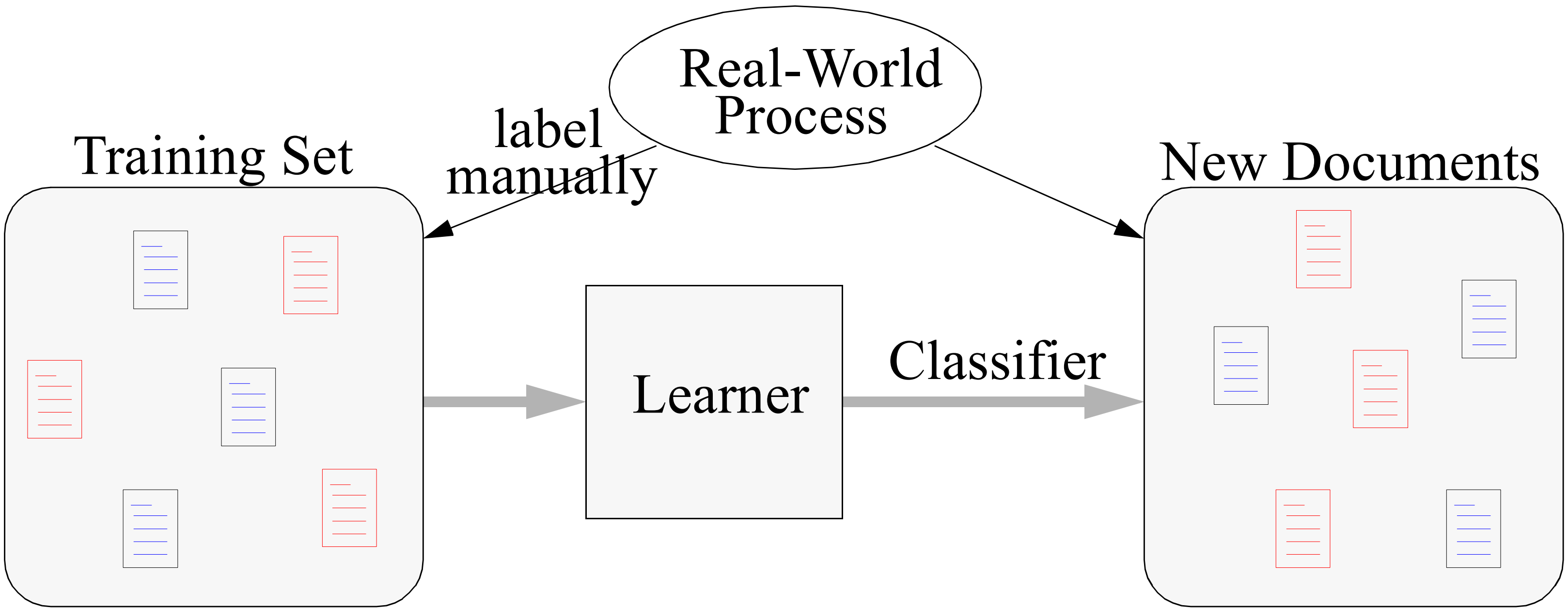
NO

Tasks and Applications

Text-Classification Task	Application
Text Routing	Help-Desk Support: Who is an appropriate expert for a particular problem?
Information Filtering	Information Agents: Which news articles are interesting to a particular person?
Relevance Feedback	Information Retrieval: What are other documents relevant for a particular query?
Text Categorization	Knowledge Management: Organizing a document database by semantic categories.

Hand-coding text classifiers is costly or even impractical!

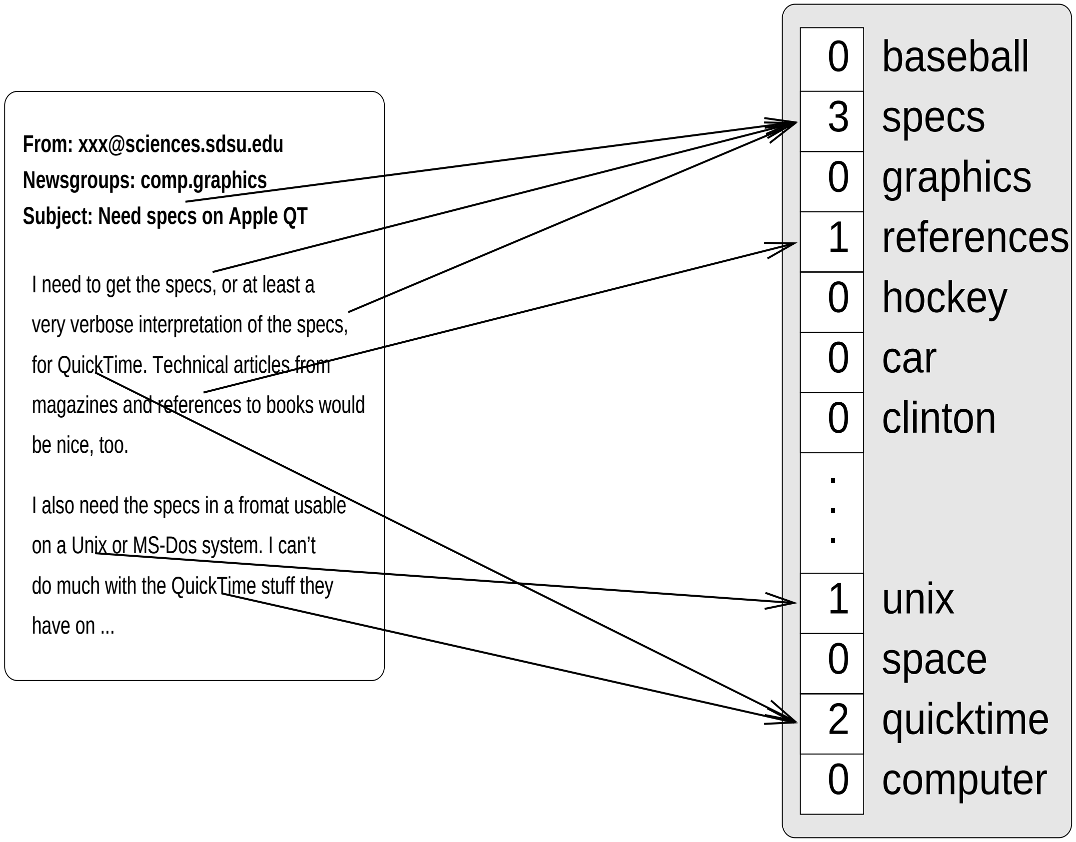
Learning Text Classifiers



Goal:

- Learner uses training set to find classifier with low prediction error.

Representing Text as Attribute Vectors



Attributes: Words (Word-Stems)

Values: Occurrence-Frequencies

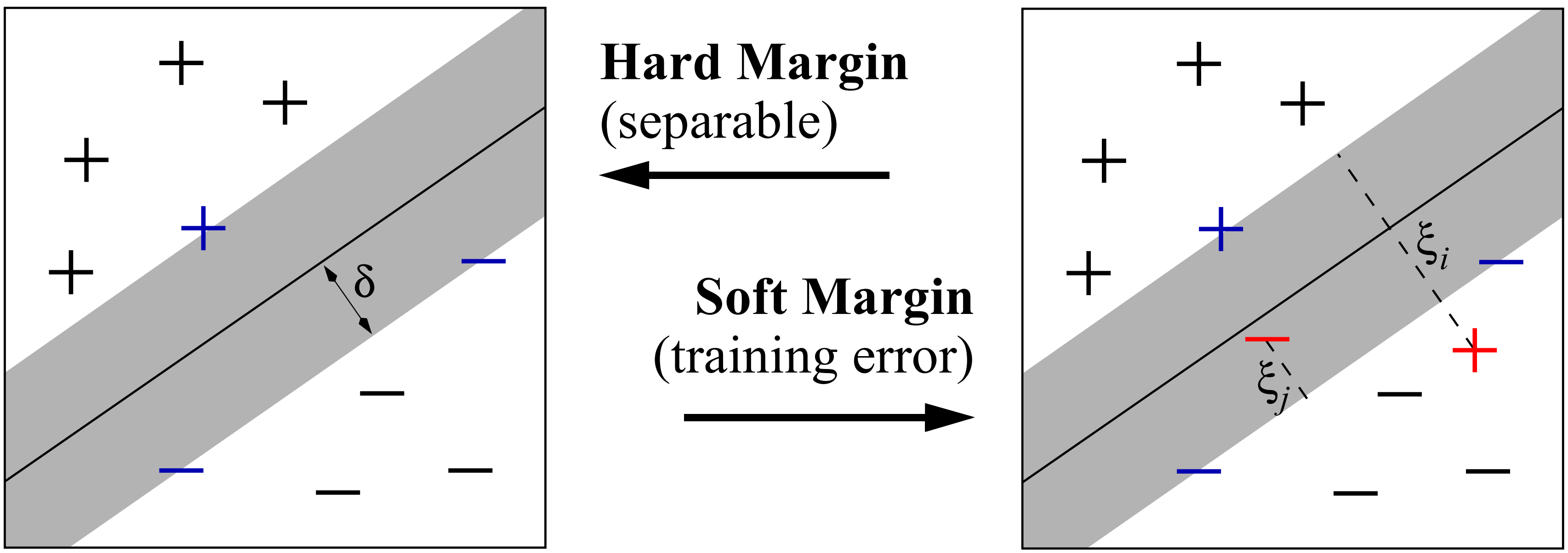
==> The ordering of words is ignored!

Support Vector Machines

Training Examples: $(\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n) \quad \vec{x}_i \in \Re^N \quad y_i \in \{1, -1\}$

Hypothesis Space: $h(\vec{x}) = \mathrm{sgn}\left[\vec{w} \cdot \vec{x} + b\right]$ with $\vec{w} = \sum \alpha_i y_i \vec{x}_i$

Training: Find hyperplane $\langle \vec{w}, b \rangle$ with minimal $\frac{1}{\delta^2} + C \sum_{i=1}^{n} \xi_i$



Experimental Results

Reuters Newswire
- 90 categories
- 9603 training doc.
- 3299 test doc.
- ~27000 features

WebKB Collection
- 4 categories
- 4183 training doc.
- 226 test doc.
- ~38000 features

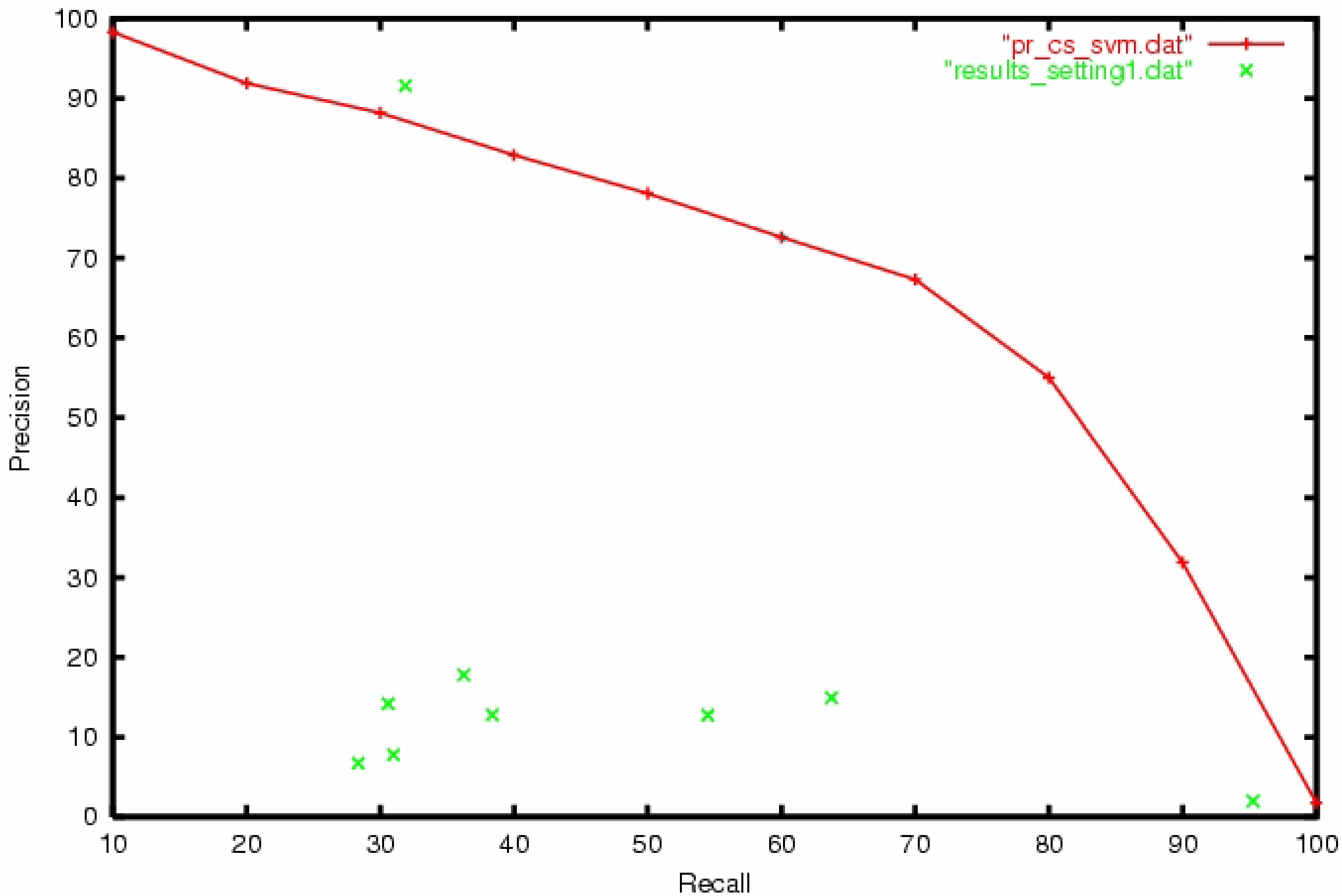
Ohsumed MeSH
- 20 categories
- 10000 training doc.
- 10000 test doc.
- ~38000 features

microaveraged precision/recall breakeven-point [0..100]	**Reuters**	**WebKB**	**Ohsumed**
Naive Bayes	72.3	82.0	62.4
Rocchio Algorithm	79.9	74.1	61.5
C4.5 Decision Tree	79.4	79.1	56.7
k-Nearest Neighbors	82.6	80.5	63.4
SVM	**87.5**	**90.3**	**71.6**

Table from [Joachims, 2002]

Humans vs. Machine Learning

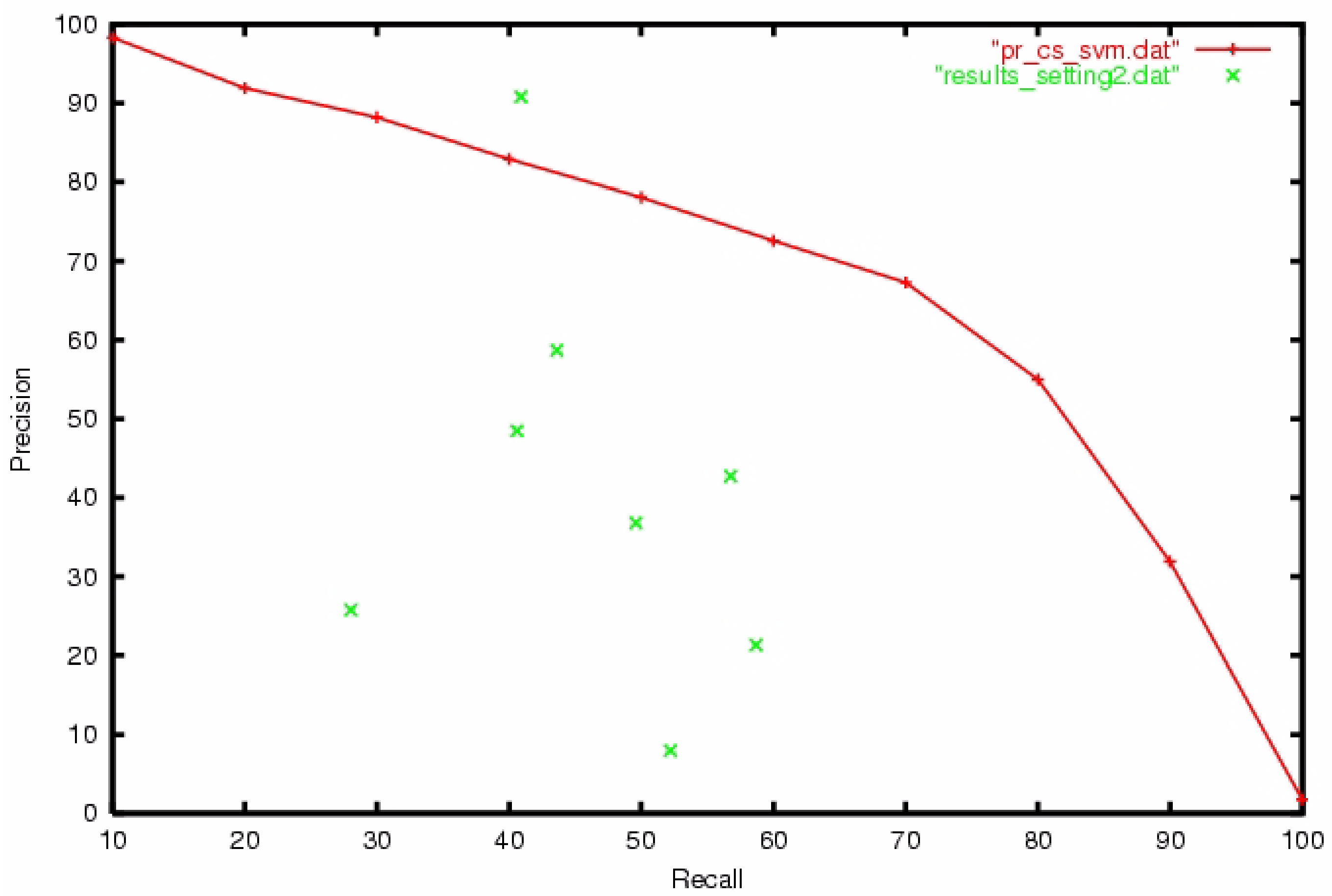
Task: Write query that retrieves all CS documents in ArXiv.org!



Data: 29,890 training examples / 32,487 test examples (relevant:=in_CS)

Humans vs. Machine Learning (Setting 2)

Task: Improve query using the training data!



Data: 29,890 training examples / 32,487 test examples (relevant:=in_CS)

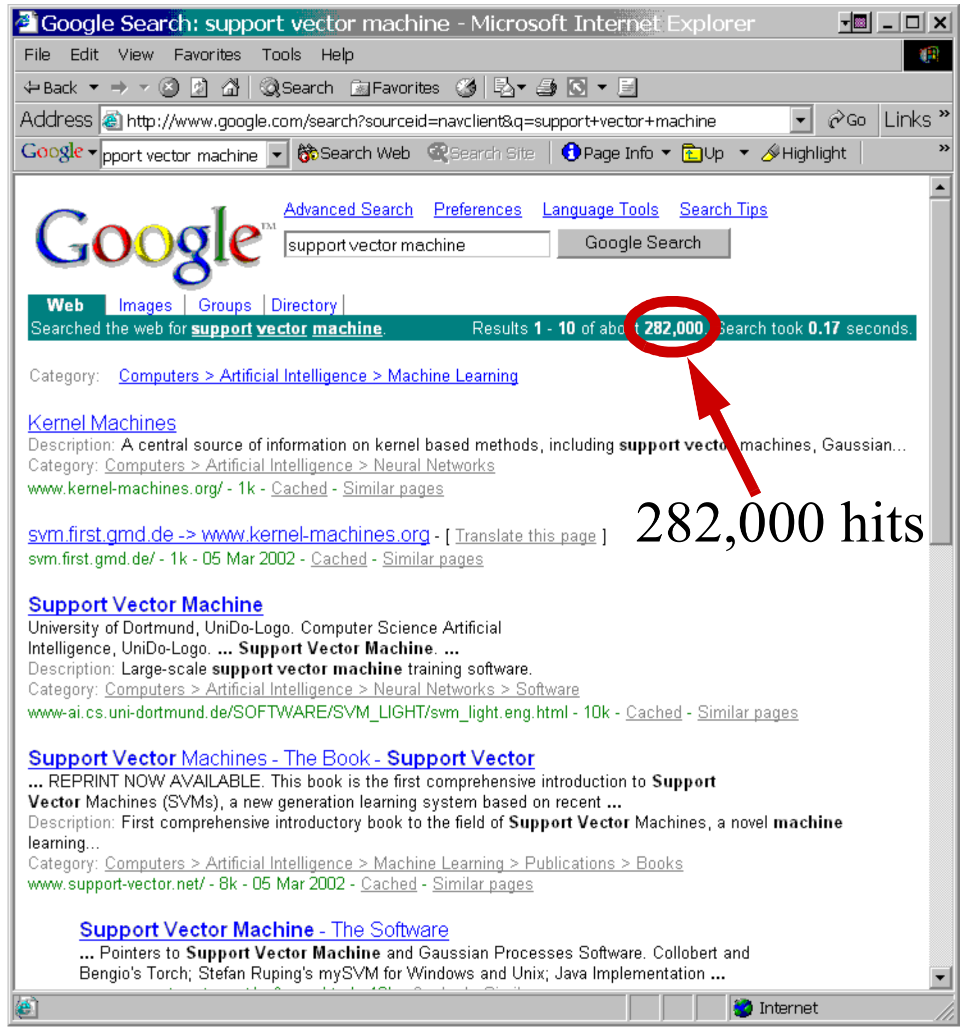
What is a Good Retrieval Function?

Query:

- "Support Vector Machine"

Goal:

- "rank the documents I want high in the list"



Training Examples from Clickthrough

Assumption: If a user skips a link *a* and clicks on a link *b* ranked lower, then the user preference reflects *rank(b) < rank(a)*.

Example: *(3 < 2) and (7 < 2), (7 < 4), (7 < 5), (7 < 6)*

Ranking Presented to User:

1. Kernel Machines
   http://svm.first.gmd.de/
2. Support Vector Machine
   http://jbolivar.freeservers.com/
3. SVM-Light Support Vector Machine
   http://ais.gmd.de/~thorsten/svm light/
4. An Introduction to Support Vector Machines
   http://www.support-vector.net/
5. Support Vector Machine and Kernel ... References
   http://svm.research.bell-labs.com/SVMrefs.html
6. Archives of SUPPORT-VECTOR-MACHINES ...
   http://www.jiscmail.ac.uk/lists/SUPPORT...
7. Lucent Technologies: SVM demo applet
   http://svm.research.bell-labs.com/SVT/SVMsvt.html
8. Royal Holloway Support Vector Machine
   http://svm.dcs.rhbnc.ac.uk/

Training Examples from Clickthrough

Assumption: If a user skips a link *a* and clicks on a link *b* ranked lower, then the user preference reflects *rank(b) < rank(a)*.

Example: *(3 < 2) and (7 < 2), (7 < 4), (7 < 5), (7 < 6)*

Ranking Presented to User:

1. Kernel Machines
   http://svm.first.gmd.de/
2. Support Vector Machine
   http://jbolivar.freeservers.com/
3. SVM-Light Support Vector Machine
   http://ais.gmd.de/~thorsten/svm light/
4. An Introduction to Support Vector Machines
   http://www.support-vector.net/
5. Support Vector Machine and Kernel ... References
   http://svm.research.bell-labs.com/SVMrefs.html
6. Archives of SUPPORT-VECTOR-MACHINES ...
   http://www.jiscmail.ac.uk/lists/SUPPORT...
7. Lucent Technologies: SVM demo applet
   http://svm.research.bell-labs.com/SVT/SVMsvt.html
8. Royal Holloway Support Vector Machine
   http://svm.dcs.rhbnc.ac.uk/

Learning to Rank

Assume:

- distribution of queries P(Q)
- distribution of target rankings for query P(R | Q)

Given:

- collection D of m documents
- i.i.d. training sample $(q_1, r_1), \ldots, (q_n, r_n)$

Design:

- set of ranking functions F, with elements f:$Q \rightarrow P^{D \times D}$ (weak ordering)
- loss function $l(r_a, r_b)$
- learning algorithm

Goal:

- find $f^\circ \in F$ with minimal

$$R_P(f) = \int l(f(q), r) dP(q, r)$$

A Loss Function for Rankings

For two orderings r_a and r_b, a pair $d_i \neq d_j$ is

- *concordant*, if r_a and r_b agree in their ordering
  P = number of concordant pairs
- *discordant*, if r_a and r_b disagree in their ordering
  Q = number of discordant pairs

Loss function: [Kemeny & Snell, 62], [Wong et al, 88], [Cohen et al, 1999], [Crammer & Singer, 01], [Herbrich et al., 98] ...

$$l(r_a, r_b) = Q$$

Example:

$$r_a = (a, c, d, b, e, f, g, h)$$

$$r_b = (a, b, c, d, e, f, g, h)$$

=> discordant pairs *(c,b), (d,b)* => $l(r_a, r_b) = 2$

A Loss Function for Rankings

For two orderings r_a and r_b, a pair $d_i \neq d_j$ is

- *concordant*, if r_a and r_b agree in their ordering
  P = number of concordant pairs
- *discordant*, if r_a and r_b disagree in their ordering
  Q = number of discordant pairs

Loss function: [Kemeny & Snell, 62], [Wong et al, 88], [Cohen et al, 1999], [Crammer & Singer, 01], [Herbrich et al., 98] ...

$$l(r_a, r_b) = Q$$

Example:

$$r_a = (a, c, d, b, e, f, g, h)$$
$$r_b = (a, b, c, d, e, f, g, h)$$

=> discordant pairs *(c,b), (d,b)* => $l(r_a, r_b) = 2$

A Loss Function for Rankings

For two orderings r_a and r_b, a pair $d_i \neq d_j$ is

- *concordant*, if r_a and r_b agree in their ordering
  P = number of concordant pairs
- *discordant*, if r_a and r_b disagree in their ordering
  Q = number of discordant pairs

Loss function: [Kemeny & Snell, 62], [Wong et al, 88], [Cohen et al, 1999], [Crammer & Singer, 01], [Herbrich et al., 98] ...

$$l(r_a, r_b) = Q$$

Example:

$$r_a = (a, c, \underbrace{d, b}, e, f, g, h)$$

$$r_b = (a, \underbrace{b, c, d}, e, f, g, h)$$

=> discordant pairs *(c,b), (d,b)* => $l(r_a, r_b) = 2$

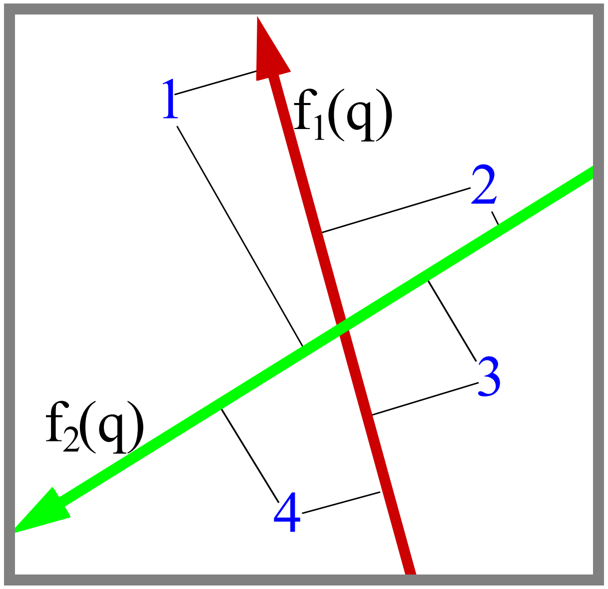
What does the Retrieval Function Look Like?

Sort documents d_i by their "retrieval status value" rsv(q,d_i) with query q [Fuhr, 89]:

$$
\begin{aligned}
\mathrm{rsv}(q,d_i) = \quad & w_1 * \#(\text{of query words in title of } d_i) \\
& + w_2 * \#(\text{of query words in H1 headlines of } d_i) \\
& \ldots \\
& + w_N * \mathrm{PageRank}(d_i) \\
= \; & \vec{w}\ \Phi(q,d_i).
\end{aligned}
$$

Select F as:

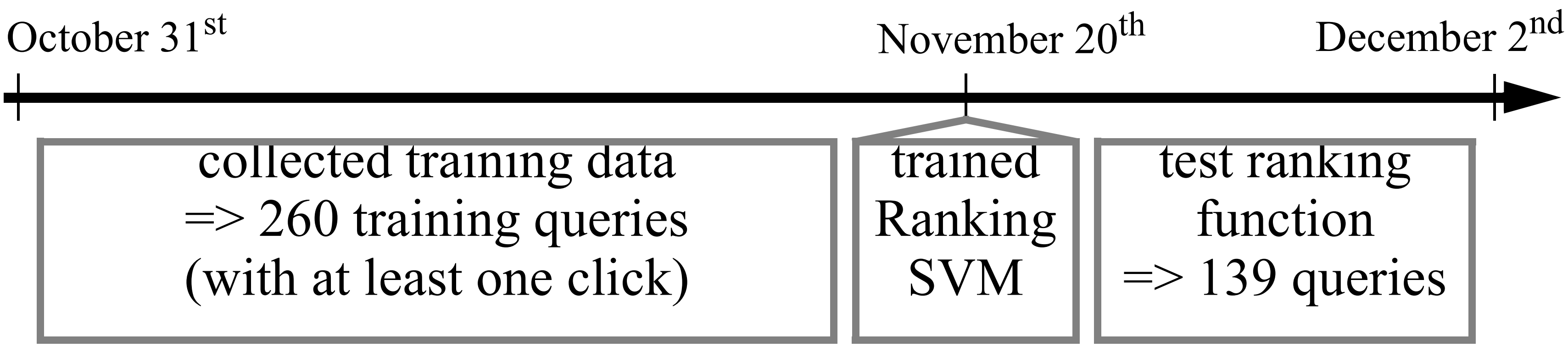
$$
d_i > d_j \\
\Leftrightarrow \\
(d_i, d_j) \in f_{\vec{w}}(q) \\
\Leftrightarrow \\
\vec{w}\Phi(q, d_i) > \vec{w}\Phi(q, d_j)
$$



Experiment

Experiment Setup:

- meta-search engine (Google, MSNSearch, Altavista, Hotbot, Excite)
- approx. 20 users
- machine learning students and researchers from University of Dortmund AI Unit (Prof. Morik)
- asked to use system as any other search engine
- display title and URL of document



Query/Document Match Features $\Phi(q,d)$

Rank in other search engine:

- Google, MSNSearch, Altavista, Hotbot, Excite

Query/Content Match:

- cosine between URL-words and query
- cosine between title-words and query
- query contains domain-name

Popularity-Attributes:

- length of URL in characters
- country code of URL
- domain of URL
- word "home" appears in title
- URL contains "tilde"
- URL as an atom

Experiment: Learning vs. Google/MSNSearch

Ranking A	Ranking B	A better	B better	Tie	Total
Learned	Google	29	13	27	69
Learned	MSNSearch	18	4	7	29
Learned	Toprank	21	9	11	41

~20 users, as of 2nd of December

Toprank: rank by increasing mimium rank over all 5 search engines

=> Result: Learned > Google
Learned > MSNSearch
Learned > Toprank

Learned Weights

weight	feature
0.60	cosine between query and abstract
0.48	ranked in top 10 from Google
0.24	cosine between query and the words in the URL
0.24	document was ranked at rank 1 by exactly one of the 5 search engines
...	
0.17	country code of URL is ".de"
0.16	ranked top 1 by HotBot
...	
-0.15	country code of URL is ".fi"
-0.17	length of URL in characters
-0.32	not ranked in top 10 by any of the 5 search engines
-0.38	not ranked top 1 by any of the 5 search engines

Summary

Why and when is it good to use ML?

- humans are not really good at it (e.g. constructing classification rules)
- training data is cheap and plenty (e.g. clickthrough)
- no expert is available (e.g. rules for filtering my email)
- its just too expensive to do by hand (e.g. ArXiv classification, personal retrieval functions)

Further Info:

- Demo retrieval system for Cornell
  => Striver: http://www.cs.cornell.edu/~tj/striver
- CS478: Introduction to Machine Learning (Spring 03)
- CS678: Advanced Topics in Machine Learning (Spring 03)
- CS574: Language Technologies (currently)