RuSSIR

Russian Summer School in Information Retrieval 2007

Machine Learning for Web Related Problems

Pavel Dmitriev, Mikhail Bilenko

Microsoft® Research

Y!™

Tutorial Outline

- **Introduction to ML**
- **ML against SPAM**
- **ML for Information Extraction**

- **ML in Web Search**
- **ML in the News**
- **ML for Finding Compound Documents**

Tutorial Outline

- **Introduction to ML**
- **ML against SPAM**
- **ML for Information Extraction**

- ML in Web Search
- ML in the News
- ML for Finding Compound Documents

Part 1

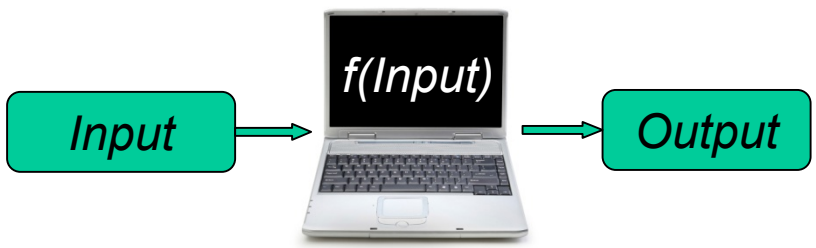
Machine Learning: Introduction

Pavel Dmitriev, Mikhail Bilenko

Some slides provided by Radford Neal

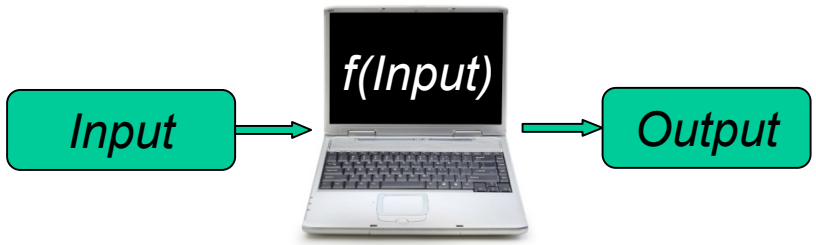
Machine Learning: Motivation

- **Any computational task:**



- **For many tasks, programming *f* manually is impossible**
  - Knowledge Engineering Bottleneck: *f* is too complex
    - Handwriting recognition, medical diagnosis
  - Need for adaptation: *f* changes at runtime
    - Spam filtering, AI in games
  - Knowledge discovery/data mining: *f* is undefined without input
    - Market basket analysis (discovery of correlations)

Machine Learning: Motivation



- **Solution:** search through function space F to find optimum f^* from F
- **Example:** make f depend on parameters, write the program to *learn* parameter values from examples, feedback, data, …
  - $f(Input) \equiv f_{\Lambda}(Input)$ where $\Lambda=\{\lambda_1, ..., \lambda_n\}$
- **Machine learning:** algorithms that use training data to identify optimum f^* for a given task

Outline

- **Definition of Machine Learning**
- **Standard Problems**
- **Canonical Settings**
- **Bayesian and Optimization Approaches to ML**

ML: Classic Definition

- **[Mitchell 1997]:** An algorithm that improves on task *T*, with respect to performance measure *P*, based on experience *E*
- **T: Spam detection**
  - P: Number of false negatives (spam in Inbox) and false positives (good messages in Junk)
  - E: Lots of email, some of it labeled as spam
- **T: Ranking for web search**
  - P: Some ranking metric (e.g., Mean Average Precision)
  - E: A training set of queries and top documents for them
- **T: Financial prediction**
  - P: Net worth (actually not that simple: constraints on volatility,…)
  - E: Market/trading history

Task Examples

Task	Input	Output
Ranking for web search and advertising	**Query**	**Ranking of relevant web pages and ads**
Webpage classification	**Webpage**	**Topical category of page**
Spam detection	**Email message**	**Spam vs. Not Spam**
Medical diagnosis	**Medical history, tests, images, …**	**Disease vs. No disease** **Request for more tests…**
Information extraction	**A document**	**Extracted entities (names, dates, locations, brands,…)**
Financial prediction	**Market data**	**Buy vs. Sell vs. Hold**
News clustering	**News articles**	**Grouping of articles**
Speech recognition	**Audio signal**	**Sequence of words**
Recommending	**User ratings, history of viewed/purchased items**	**Recommendations of novel items (to buy, to read, to watch…)**

ML = Improve on task *T*, wrt perf. measure *P*, based on experience *E*

Performance Measures

- **Performance measure computes error on *distinct test set***
- **Error costs can be imbalanced, multi-dimensional, indirect**
  - Webpage classification: % of correctly classified pages
  - Spam: % of false positives, % of false negatives
  - Medical diagnosis: % of false positives, % of false negatives (!)
  - Financial prediction: earnings, risk, earnings wrt market, …
  - Recommending: uptake, ratings, customer churn, …
- **"Prime directive" of machine learning experiments:** test data is *held-out* (not seen during training); multiple samples used for statistical testing

ML = Improve on task *T*, wrt perf. measure *P*, based on experience *E*

Experience – Training Data

- **Classic assumption: training instances are independently sampled from some underlying distribution**
  - IID: independently and identically distributed
- **Assumption is false when instances are related**
  - Examples:
    - Webpage classification: links tend to connect same-category pages
    - Spam: multiple emails from same address are all spam/not spam
  - Collective/Relational learning methods account for the connections
- **Active learning: learner can construct/request labels for most informative examples**
  - Typically "least certain" / "hard" / "near miss" examples

ML = Improve on task *T*, wrt perf. measure *P*, based on experience *E*

Outline

- Definition of Machine Learning
- **Standard Problems**
- Canonical Settings
- Bayesian and Optimization Approaches to ML

Standard ML Problems: Classification

- **Input: instance *x***
  - Vector/sequence/graph/set/etc.
  - May include continuous/discrete/ordinal/etc. attributes
- **Output: discrete $y \in \{1,..,k\}$ (set of labels)**
  - Example: binary classification – diabetes
    - Input: $x = \{blood\_pressure, heart\_rate, smoking, \ldots\}$
    - Output: $y = true$ or $y = false$
- **Training data: set of instances with true labels $\{(x_i, y_i)\}_{i=1..n}$**
- **Performance measure: error rate**
  - Averaged over all categories: $\Sigma[y \neq y']$ where y' is prediction
  - Weighted for cost-sensitive classification (spam, medical, …)

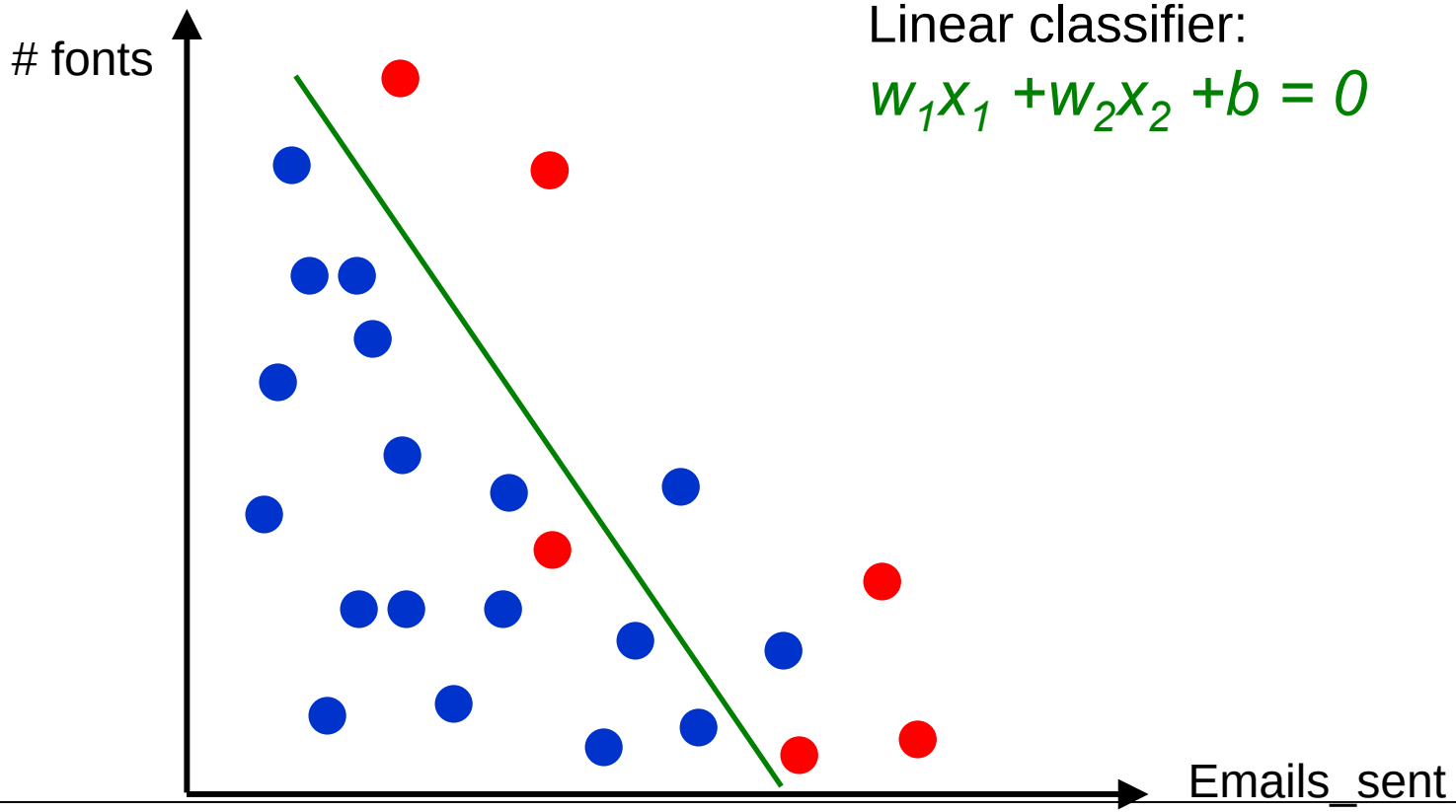
Standard ML Problems: Classification

- **Most common case: $x = [x_i]_{i=1..d}$ is a d-dimensional vector of real values**
  - Linear classifier: $y=1$ if $(w_1x_1 + w_2x_2 + \ldots + w_dx_d) > 0$
  - Rule-based classifier: $y=1$ if $(x_1>0)$ && $(x_2=3)$ && $(x_7<5)$
  - Nearest-neighbor: $y=y'$, (x',y') is training example closest to x

- **Learning as search in function space:**
  - Linear: space of all possible values for $w_1 \ldots w_k$
  - Rule-based: all possible rules (boolean formulas)
  - Nearest-neighbor: all possible distance metrics (defining "closest")

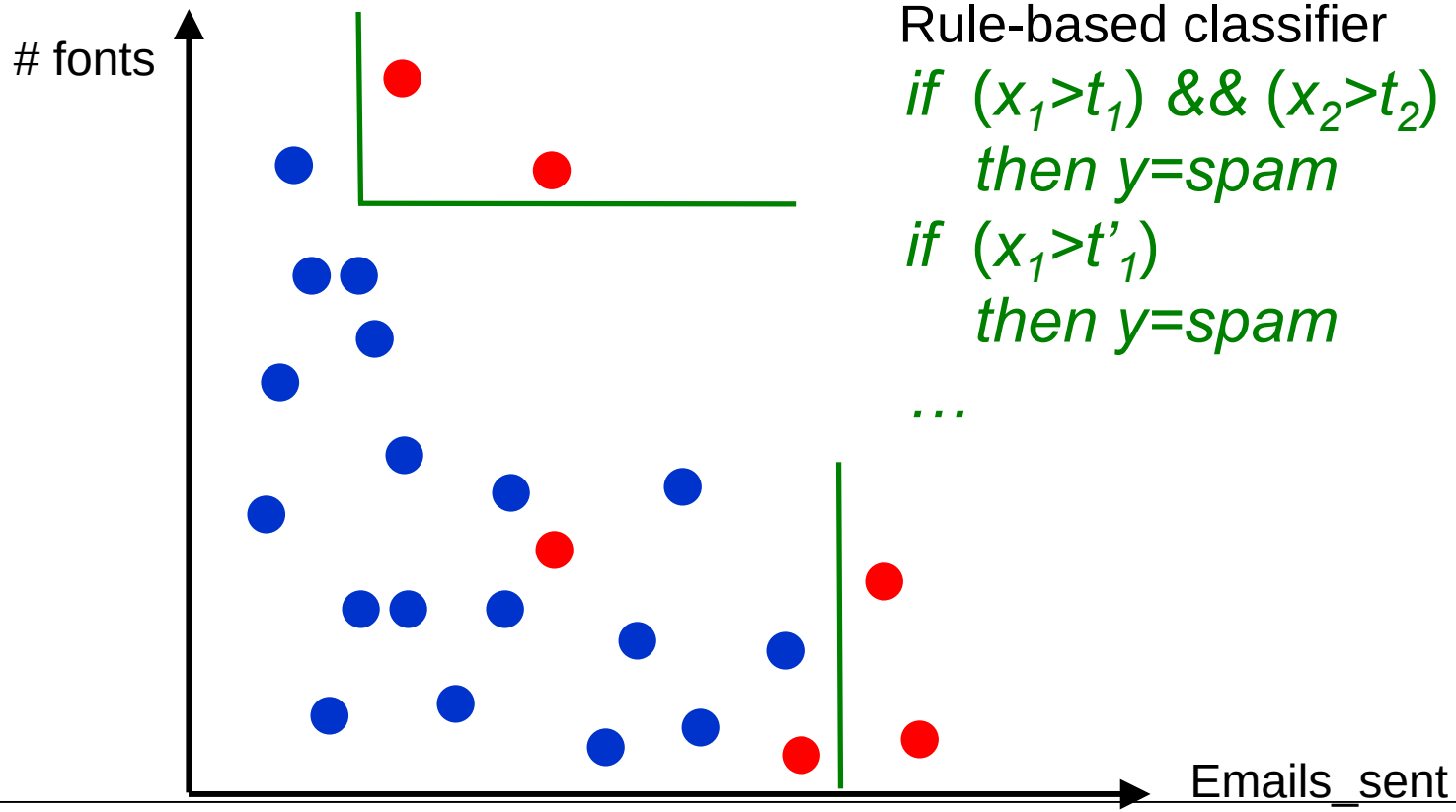
Classification Example

- **Learning for spam:**
  - x=[number_of_fonts_used, emails_sent_by_author]
  - y = spam/not_spam



Classification Example

- **Learning for spam:**
  - x=[number_of_fonts_used, emails_sent_by_author]
  - y = spam/not_spam



Rule-based classifier

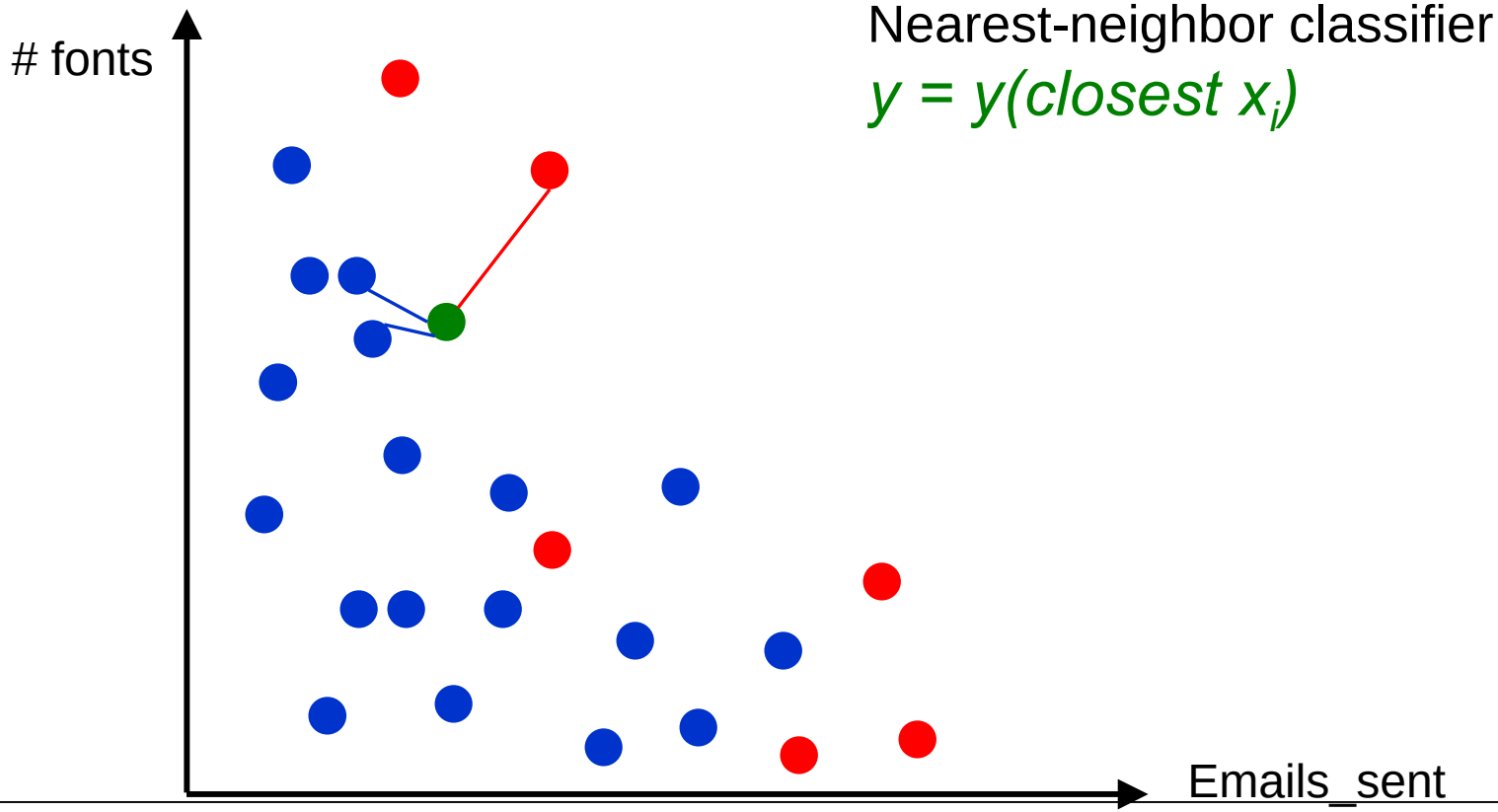
if $(x_1 > t_1)$ && $(x_2 > t_2)$
then y=spam

if $(x_1 > t'_1)$
then y=spam

…

Classification Example

- **Learning for spam:**
  - x=[number_of_fonts_used, emails_sent_by_author]
  - y = spam/not_spam



Nearest-neighbor classifier

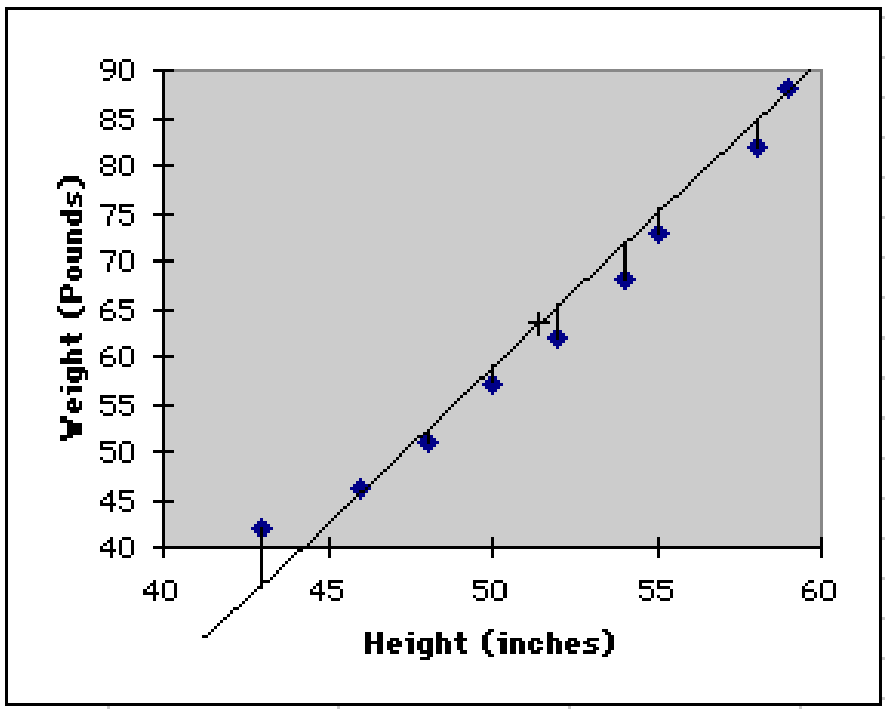
$y = y(\text{closest } x_i)$

Standard ML Problems: Regression

- **Input: instance *x***
  - Vector/sequence/graph/set/etc.
  - May include continuous/discrete/ordinal/etc. attributes
- **Output: continuous *y* (numeric output)**
- **Example: web traffic prediction (e.g., news site)**
  - Input: $x = \{day\_of\_week, today\_traffic, breaking\_news, \ldots\}$
  - Output: $y = traffic\_tomorrow$
- **Training data: set of instances with true output $\{(x_i,y_i)\}_{i=1..n}$**
- **Performance measure: error rate**
  - L2-norm: $\Sigma(y-y')^2$
  - L1-norm: $\Sigma|y-y'|$

Regression Example

- **Dataset:**
  - x = heights of boys in Birmingham, England
  - y = their weights

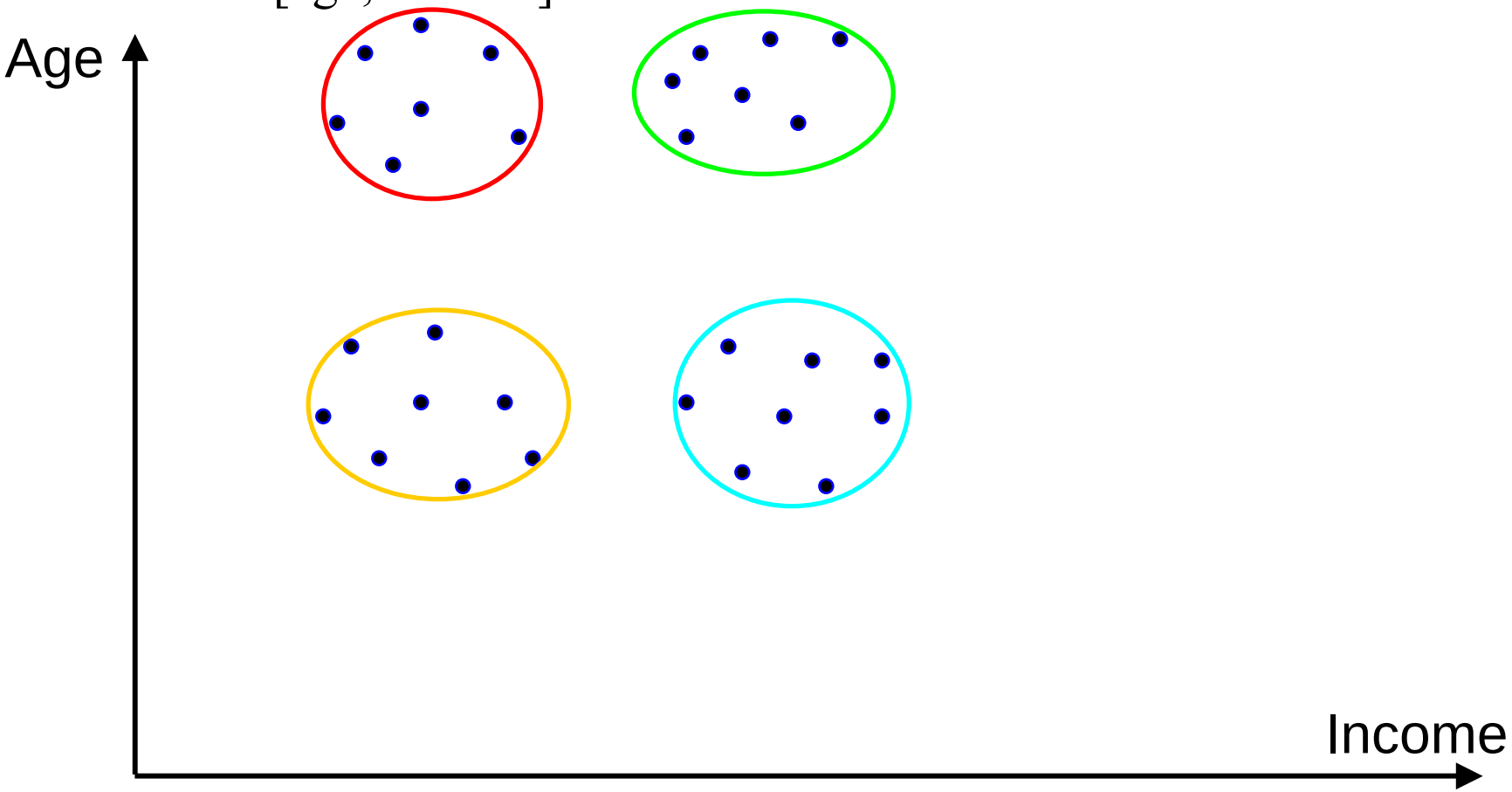


Standard ML Problems: Clustering

- **Input: set of instances $x=\{x_i\}_{i=1..n}$**
- **Output: partitioning the dataset into k non-overlapping clusters $X=\{X_1,\ldots, X_k\}$**
  - Example: news clustering (part 5 of this tutorial)
- **Training data: none!**
- **Performance measures:**
  - Recall: given a true clustering, compute the proportion of correct clusters
  - Pairwise accuracy: given a true clustering, compute the proportions of same-cluster and different-cluster pairs that have been correctly placed in the same/different clusters

Clustering Example

- **Identifying social groups:**
  - x=[age, income]



Bias in Machine Learning

- **Learning is search in some (very large!) function space:**
  - Classification: search for weights/rules/distances
  - Regression: search for weights
  - Clustering: search for a partitioning
- **Bias: strategy for search in function space**
- **Language bias: only consider some class of functions among all possible functions (e.g. only disjunctive rules)**
- **Search bias: search criteria besides agreement with data**
  - Simplicity, e.g., in rule-based learning, prefer a minimum number of rules that agree with data (Occam's Rasor)
  - Closeness to prior knowledge: penalize weights that are too large / too small / too different from priors

Outline

- Definition of Machine Learning
- Standard Problems
- **Canonical Settings**
- Bayesian and Optimization Approaches to ML

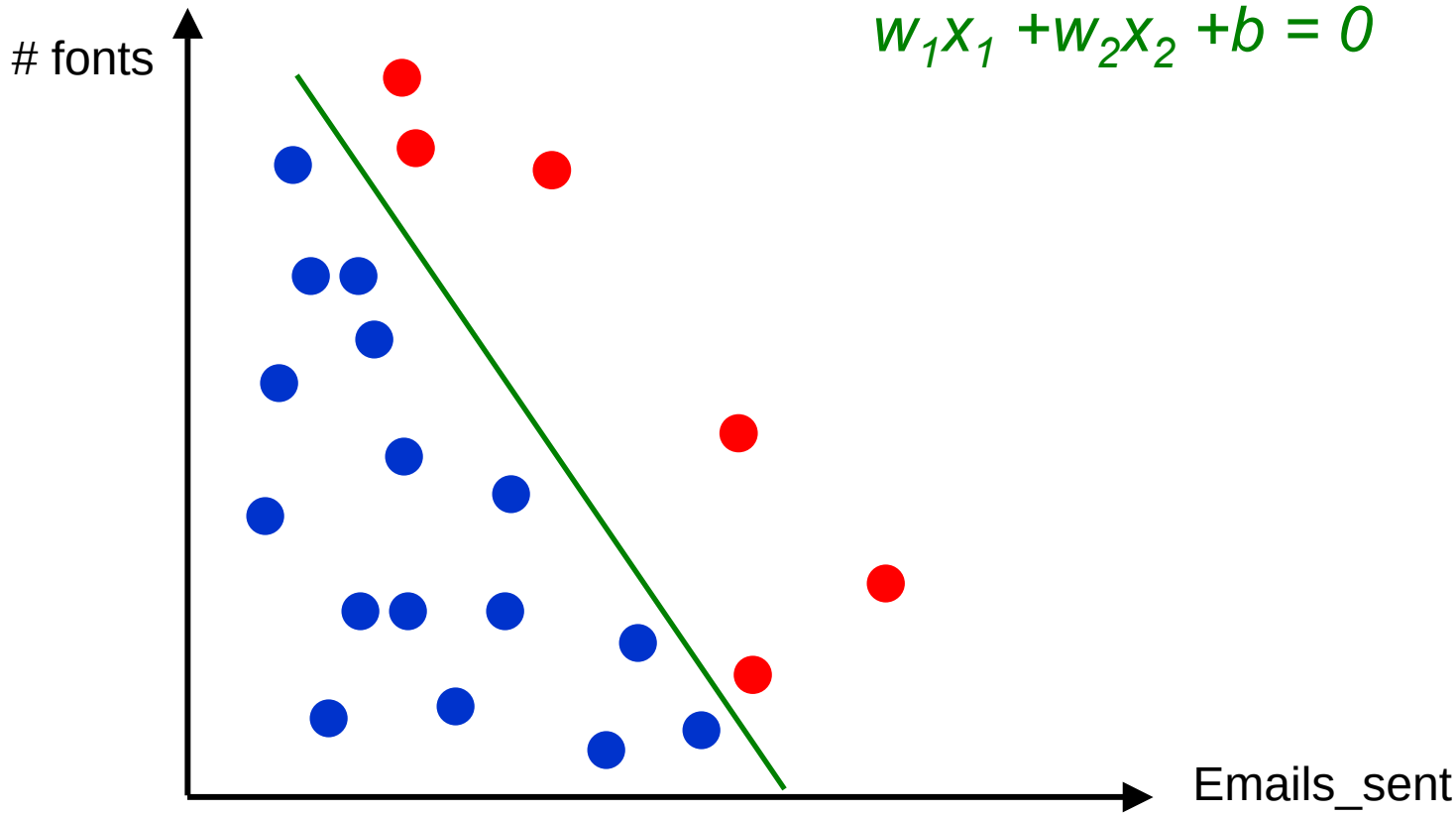
ML: Canonical Settings

- ***Unsupervised Learning*: given input data, find features/structure/a model that provides insights**
  - Topic detection, Clustering, Hierarchy Learning
- ***Supervised Learning*: given examples of input data and corresponding output data, find a function that correctly predicts outputs for future inputs**
  - Classification, Regression
- ***Semi-supervised Learning*: above tasks, given a mixture of labeled an unlabeled data**
  - Transductive Classification: know future inputs at training time
  - Semi-supervised Clustering: unlabeled data accompanied by labeled data (same/different cluster pairs or cluster labels)

Transductive Classification

Linear classifier:

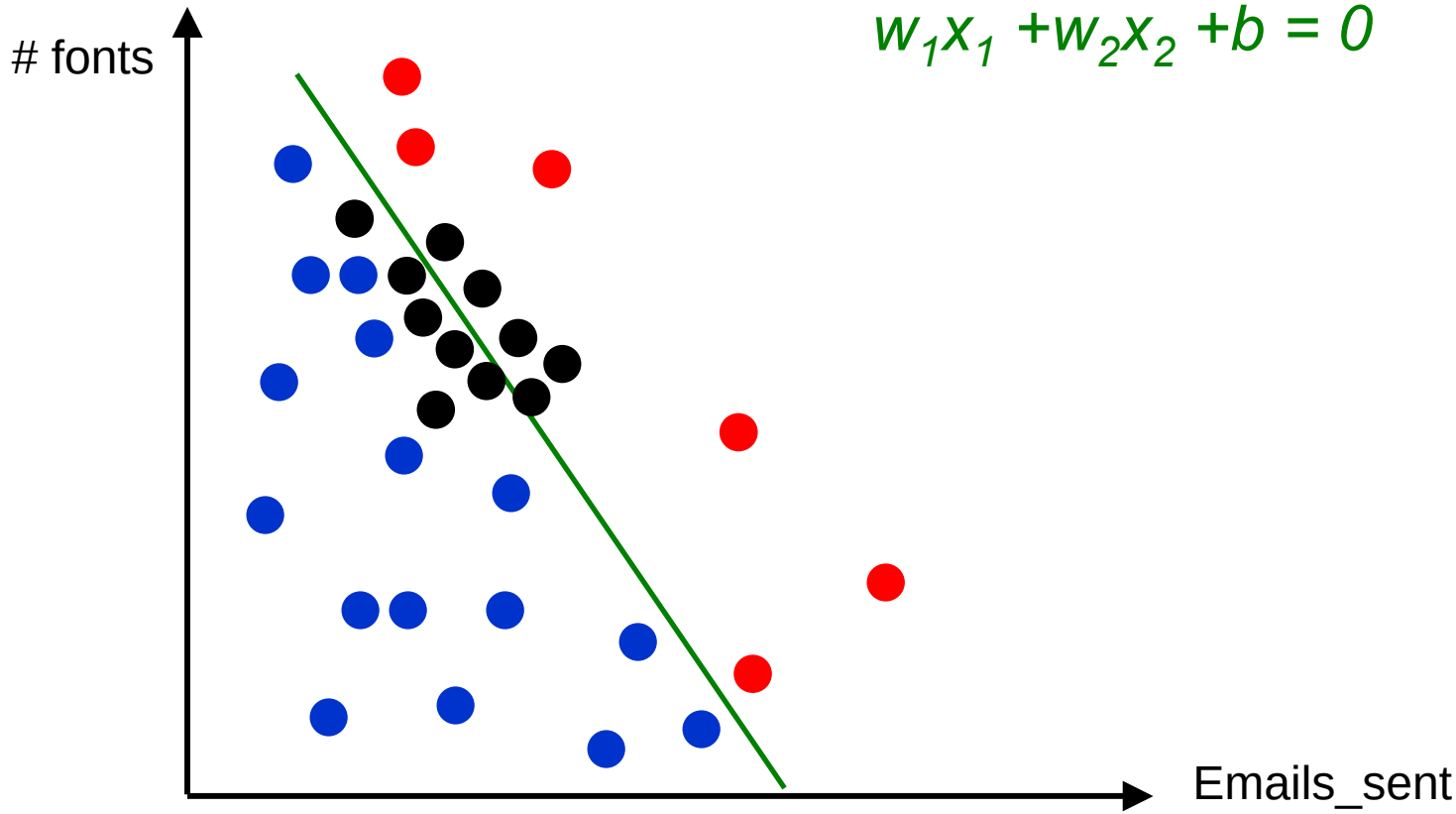
$w_1 x_1 + w_2 x_2 + b = 0$



Transductive Classification

Linear classifier:

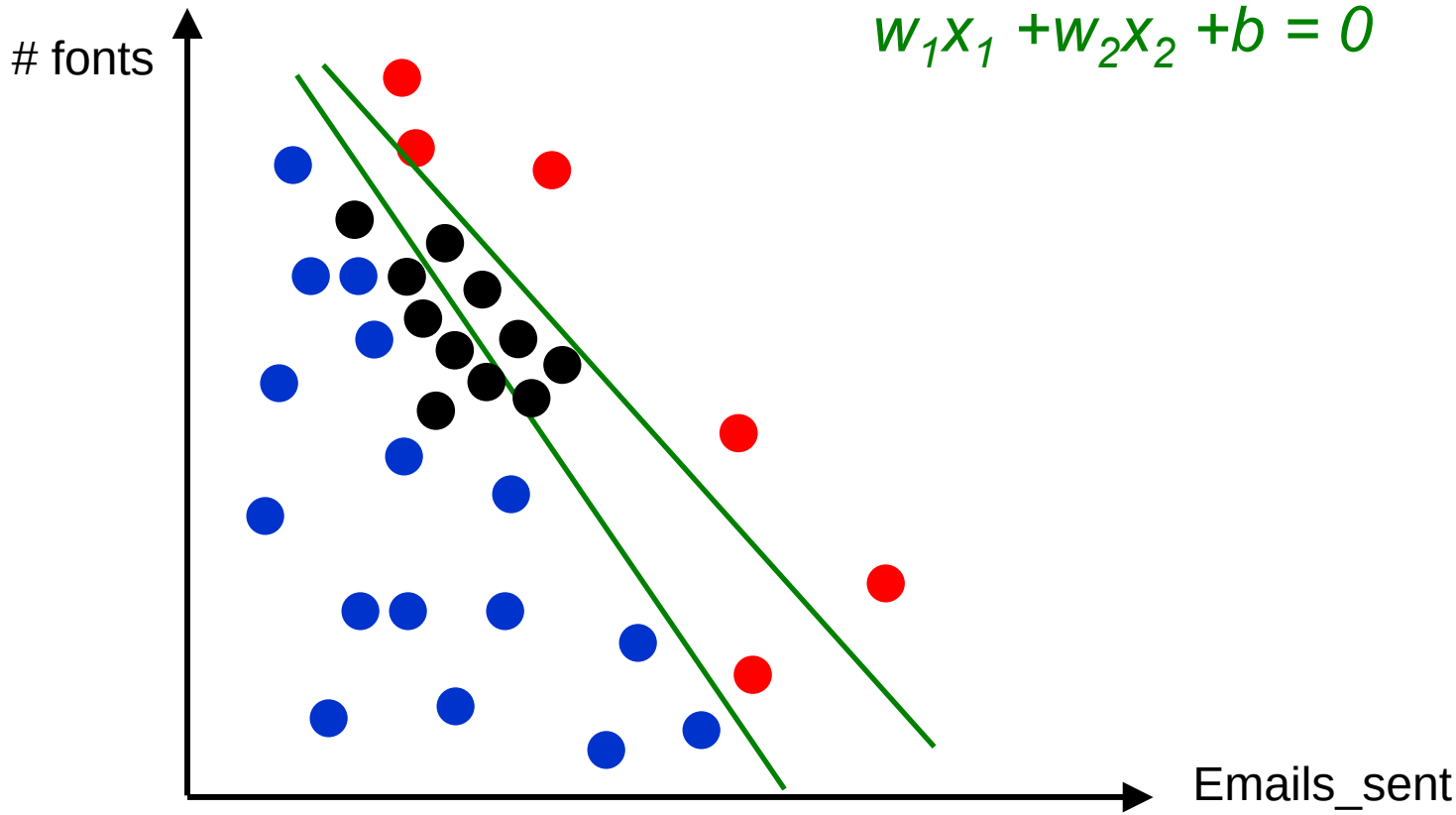
$w_1 x_1 + w_2 x_2 + b = 0$



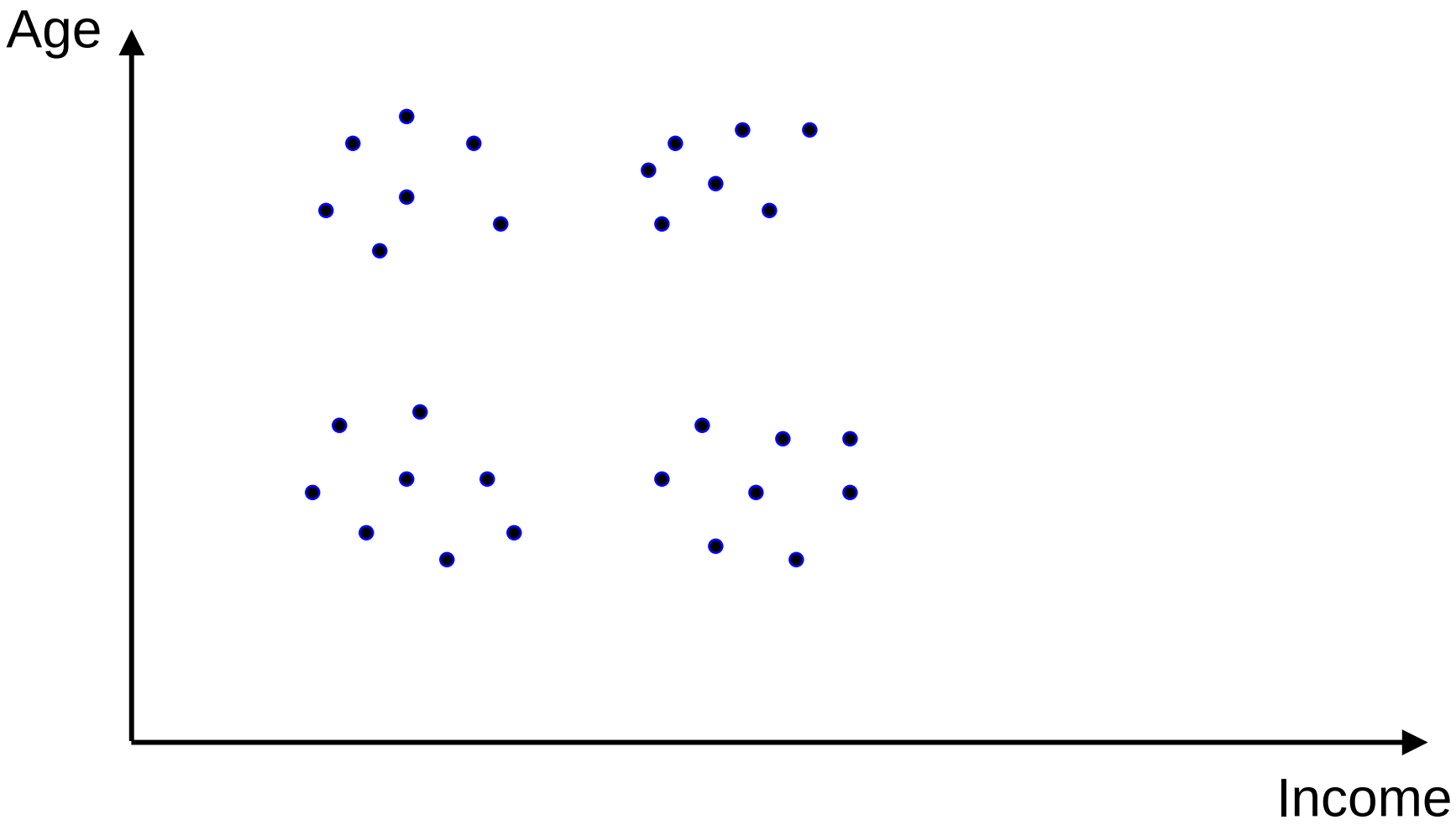
Transductive Classification

Linear classifier:

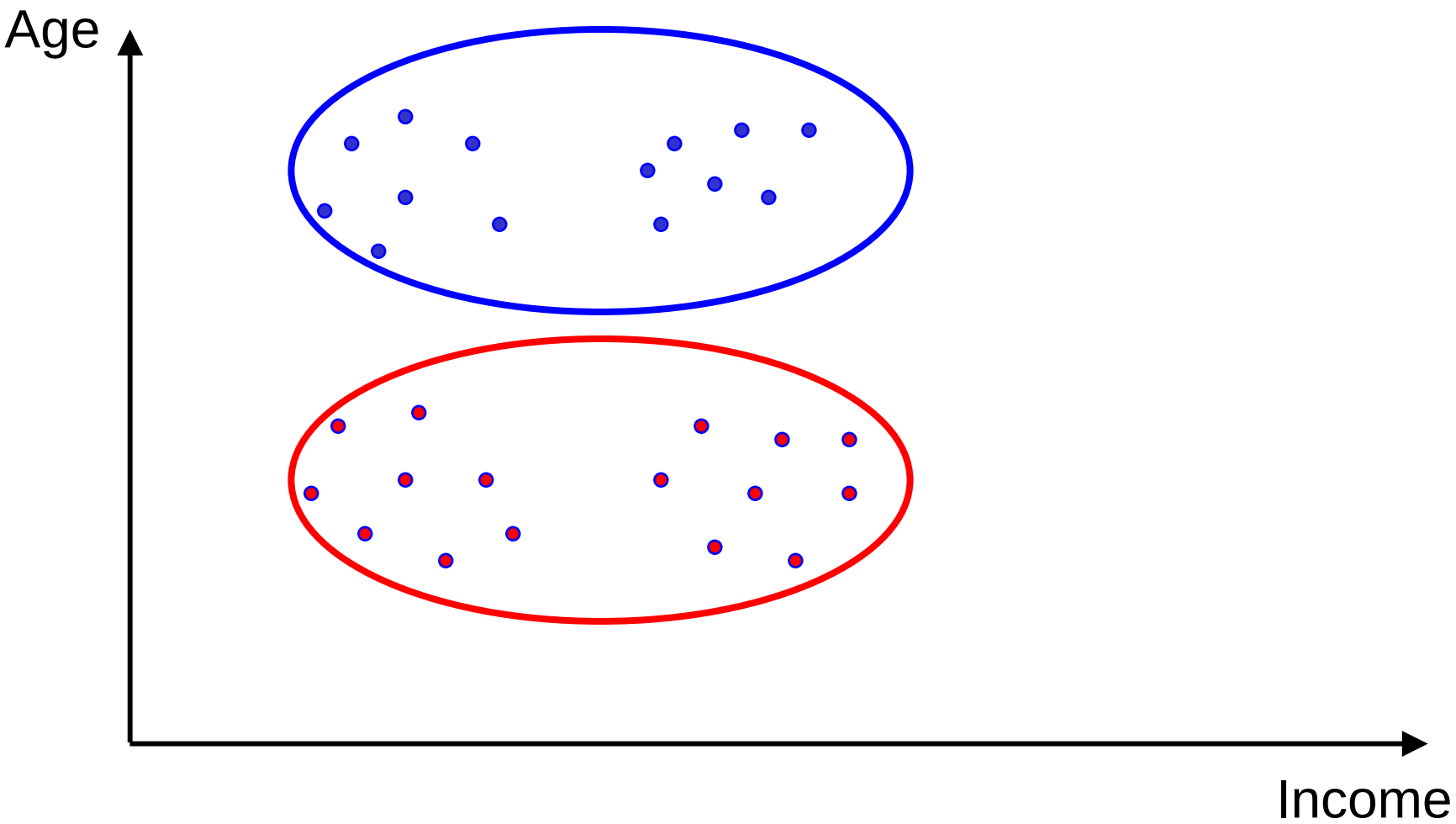
$w_1 x_1 + w_2 x_2 + b = 0$



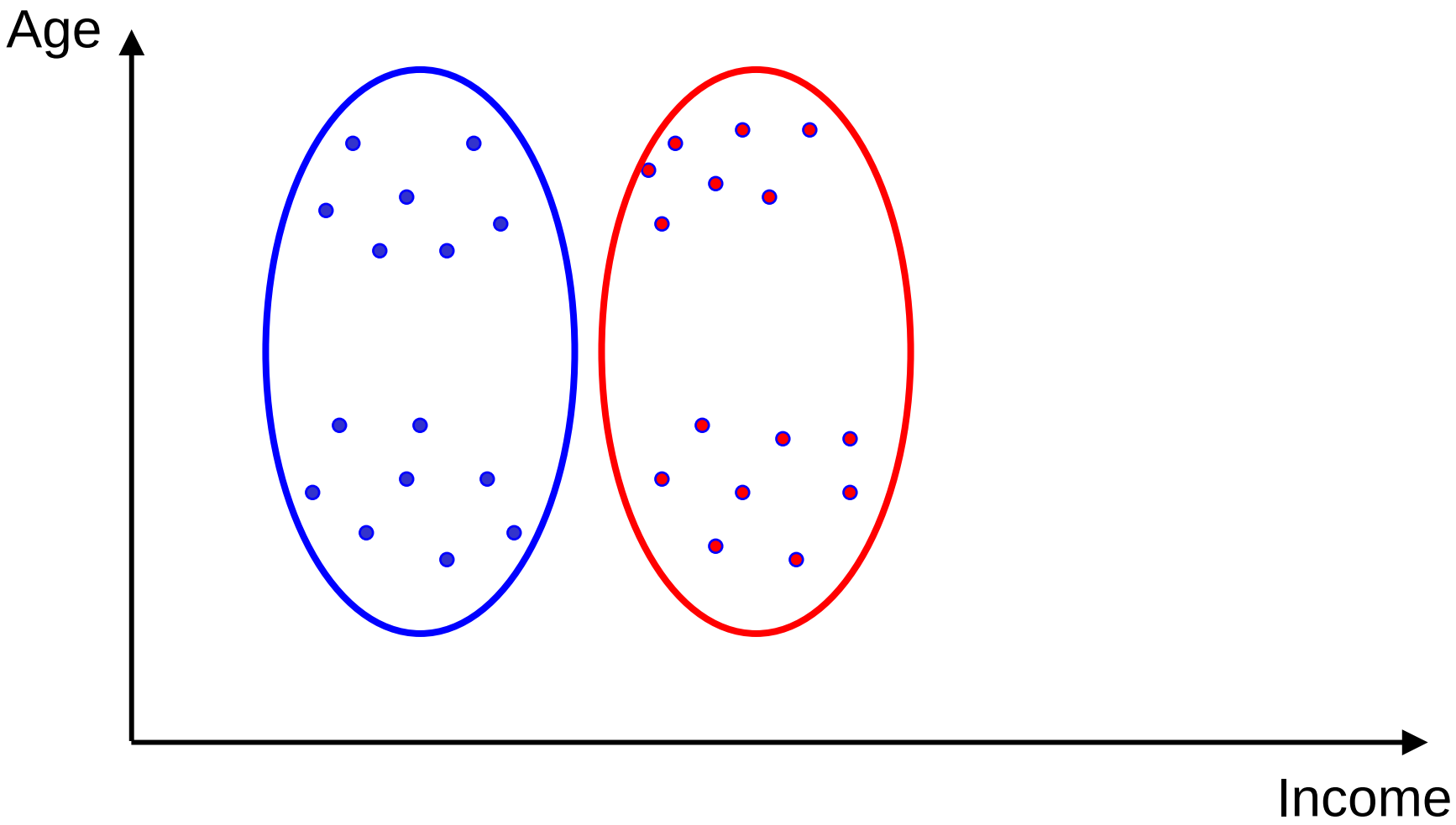
Semi-supervised Clustering



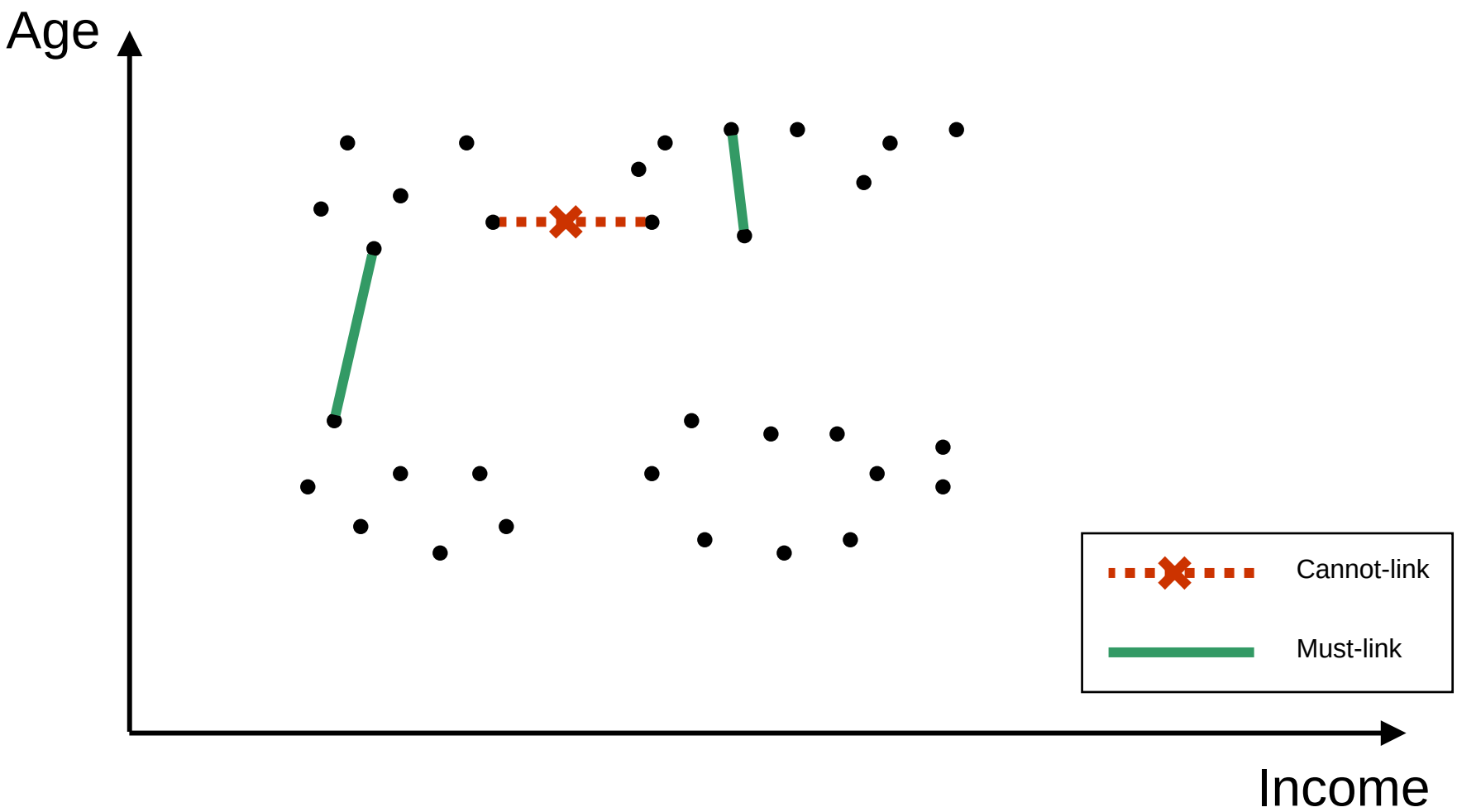
Semi-supervised Clustering



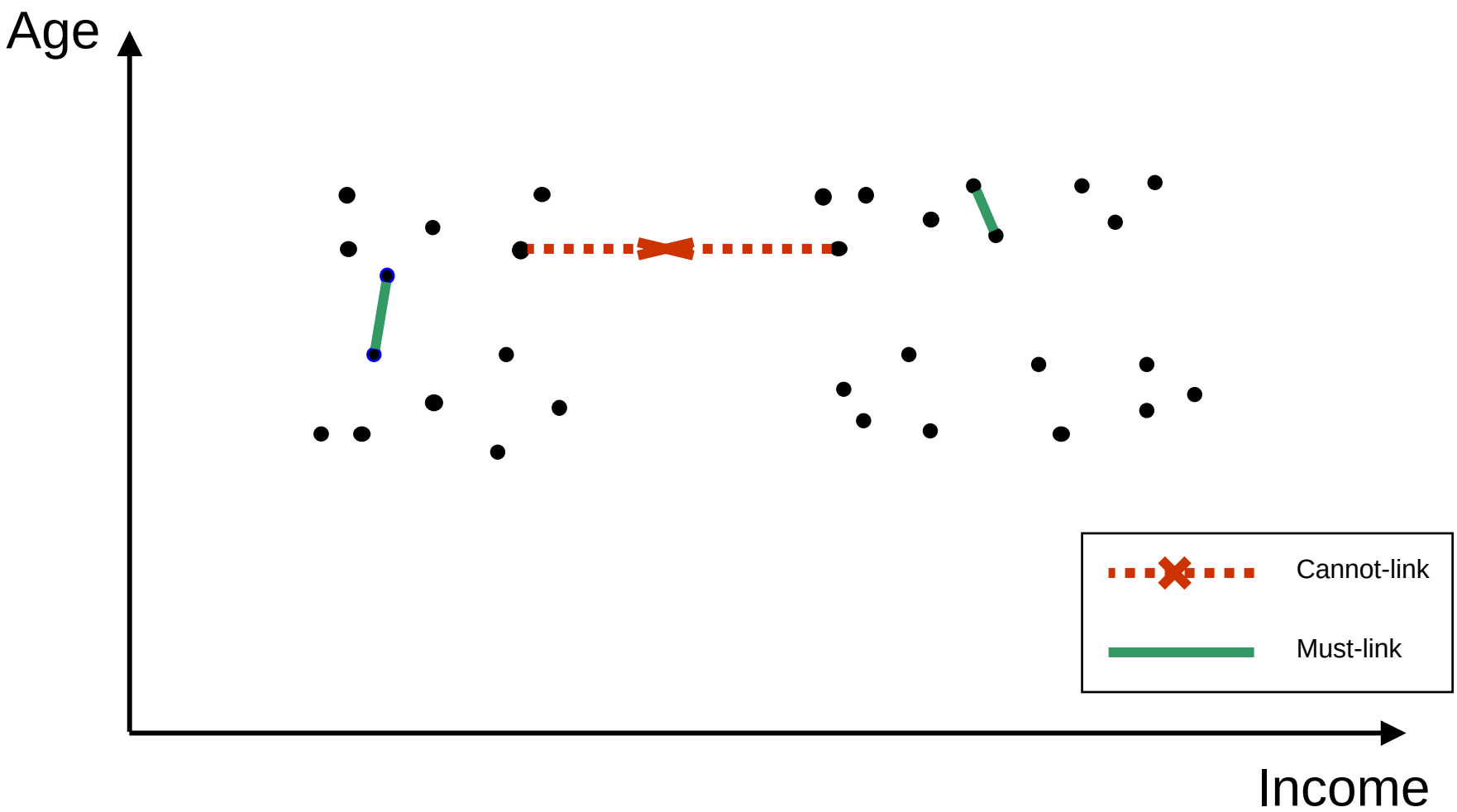
Semi-supervised Clustering



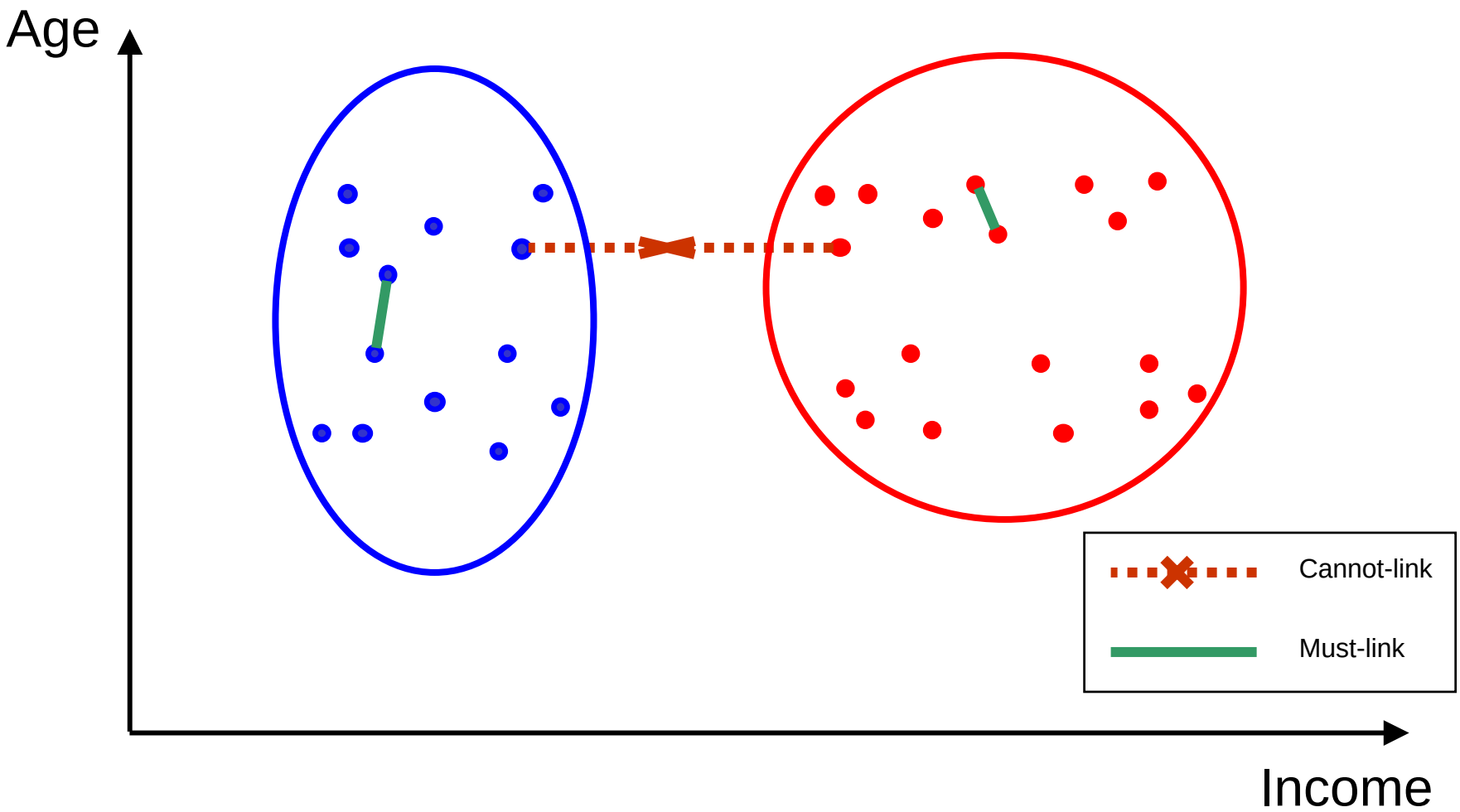
Semi-supervised Clustering



Semi-supervised Clustering



Semi-supervised Clustering



Outline

- Definition of Machine Learning
- Standard Problems
- Canonical Settings
- **Bayesian and Optimization Approaches to ML**

Bayesian Approach

- **Formulate knowledge about the situation probabilistically**
  - Define a model that expresses qualitative aspects of our knowledge (forms of distributions, independence assumptions, etc.) This model will have some unknown parameters
  - Specify a prior probability distribution for these unknown parameters (defines which values are more likely before seeing the data)
- **Gather data**
- **Compute the posterior probability distribution for the parameters, given the observed data**
- **Use this posterior distribution to**
  - Answer all kinds of questions, such as making predictions or inferring correlations

Bayesian Approach

- **Learning is finding values of parameters maximizing the posterior probability computed using the Bayes Rule**

$$parameters_{opt} = \arg\max_{parameters} P(parameters \mid data)$$

$$= \frac{P(parameters) * P(data \mid parameters)}{P(data)}$$

- **The denominator is just a normalizing constant which is often not necessary**
- **If the *P(parameters)* is uniform (all values are equally likely), then $parameters_{opt}$ is called a Maximum Likelihood Estimate (MLE)**
- **Otherwise, it is called Maximum Aposteriori Estimate (MAP)**

Optimization Approach

- **Formulate the knowledge of situation as assumptions about the form of the concept we want to learn**
  - Choose a class of functions (hypotheses) *H* so that the concept we want to learn can be expressed as a one of the functions from *H*
- **Gather data**
- **Use the data to find an optimal *f* from *H*, according to some notion of "optimal"**
  - For regression, find *f* minimizing the prediction error
  - For classification, find *f* maximizing accuracy
- **Often can prove that**

$$P(|\,testErrorRate - trainingErrorRate\,| > \varepsilon) < \partial$$

Summary

- **Machine Learning is concerned with improving the performance of an algorithm on a specific task with experience**
- **Many ML problems can be represented as one of, or a combination of several standard problems: classification, regression, and clustering**
- **Depending on the kind of information available, ML algorithms can be categorized into supervised, unsupervised, and semi-supervised**
- **Two most popular approaches to ML are Bayesian and Optimization approaches**

Note

- **This is an extremely brief overview of Machine Learning!**
- **There are many more problems, settings, and approaches in ML**
- **Often a relationship can be found among seemingly different approaches (e.g., Bayesian and Optimization approaches are often related)**

References

- **[Mitchell, 1997]** Mitchell, M.T., *Machine Learning*, McGraw Hill, 1997, ISBN 0070428077.

Part 2

Machine Learning against SPAM

Pavel Dmitriev, Mikhail Bilenko

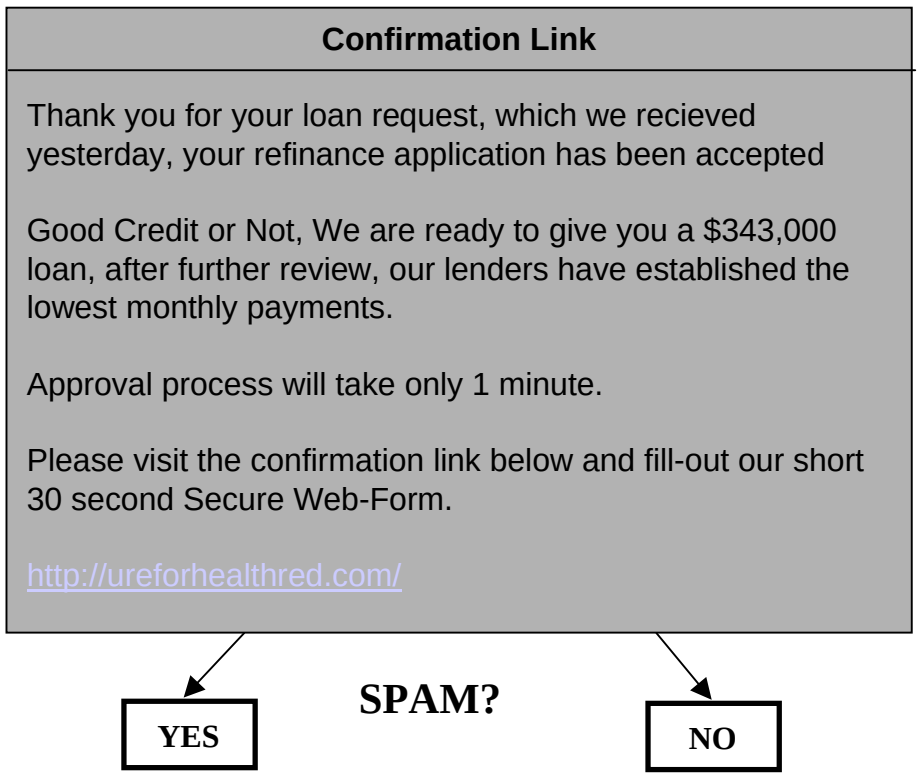
Some slides provided by Thorsten Joachims

Problem Definition

- **Given a piece of text dermine whether it is SPAM or not SPAM**
  - Email SPAM
  - Search Engine SPAM
  - Blog SPAM
  - Etc.

Classification Problem:

Assign pieces of text to predefined categories based on content



Outline

- **Bayesian Classification**
- **Naïve Bayes Classifier for Text**
- **Experimental Results for SPAM Filtering**

Generative vs. Discriminative Training

- **Training examples:** $(x_i, y_i) \sim P(X, Y)$, i=1..n

- **Discriminative Training**
  - Make assumptions about the set H of classifiers
  - Estimate error of classifiers in H from the training data
  - Select a classifier with the lowest error rate

- **Generative Training**
  - Make assumptions about the parametric form of $P(X, Y)$
  - Estimate the parameters of $P(X, Y)$ from the training data
  - Derive optimal classifier using Bayes' Rule

Bayes' Rule

- **If you know $P(Y = 1 \mid X)$ and $P(Y = -1 \mid X)$, the optimal classification is**

$$h(\bar{x}) = \begin{cases} 1, & \text{if } P(Y = 1 \mid X = \bar{x}) > P(Y = -1 \mid X = \bar{x}) \\ -1, & \text{otherwise} \end{cases}$$

- **Minimizes the prediction error**

Bayes' Theorem

- **It is possible to "switch" conditional probabilities according to the following rule**

$$P(A \mid B) = \frac{P(B \mid A) * P(A)}{P(B)}$$

- **Note that**

$$P(B) = \sum_{a \in A} P(B \mid A = a) * P(A = a)$$

Bayes' Rule/Theorem for Classification

- **Need to know conditional probability**

$$P(Y = 1 \mid X = \overline{x}) = 1 - P(Y = -1 \mid X = \overline{x})$$

to apply Bayes' Rule

- **Use Bayes' Theorem to get**

$$P(Y = 1 \mid X = \overline{x}) = \frac{P(X = \overline{x} \mid Y = 1) * P(Y = 1)}{P(X = \overline{x})}$$

- **Equivalence**

$$P(Y = 1 \mid X = \overline{x}) > P(Y = -1 \mid X = \overline{x})$$

$$\Leftrightarrow$$

$$P(X = \overline{x} \mid Y = 1) * P(Y = 1) > P(X = \overline{x} \mid Y = -1) * P(Y = -1)$$

Applying it to Text

- **Multinomial Model for Text**
  - Assume words are drawn randomly from class dependent lexicons (with replacement)
  - l_x = total number of words in document x
  - w_i = the i-th word in the document

$$P(X = \overline{x} \mid Y = 1) = \prod_{i=1}^{l_{\overline{x}}} P(W = w_i \mid Y = 1)$$

$$P(X = \overline{x} \mid Y = -1) = \prod_{i=1}^{l_{\overline{x}}} P(W = w_i \mid Y = -1)$$

Naïve Bayes Classifier for Text

- **Multinomial model for each class**

$$P(X = \overline{x} \mid Y) = \prod_{i=1}^{l_{\overline{x}}} P(W = w_i \mid Y)$$

- **Prior probabilities**

$$P(Y)$$

- **Classification rule**

$$h(\overline{x}) = \begin{cases} 1, \text{ if } P(Y=1) * \prod_{i=1}^{l_{\overline{x}}} P(W = w_i \mid Y=1) > P(Y=-1) * \prod_{i=1}^{l_{\overline{x}}} P(W = w_i \mid Y=-1) \\ -1, \text{ otherwise} \end{cases}$$

- $Y = 1$ **means SPAM,** $Y =$ **-1 means Not SPAM**

Estimating Parameters

- **Count frequences in the training data**
  - n = number of training examples
  - pos/neg = number of positive/negative training examples
  - $TF(w, y)$ = number of times word w occurs in class y
  - l_y = total number of word occurences in class y
- **Estimating** $P(Y)$

$$P(Y=1)=\frac{pos}{n} \qquad P(Y=-1)=\frac{neg}{n}$$

- **Estimating** $P(W \mid Y)$
  - Smoothing with Laplace estimate

$$P(W=w \mid Y=y)=\frac{TF(w,y)+1}{l_y+2}$$

Assumptions of Naïve Bayes

- **Words occur independently given a class according to one multinomial distribution per class**
- **Each document is in exactly one class**
- **Word probabilities do not depend on the document length**

Pros and Cons of Naïve Bayes

- **Pros:**
  - Explicit theoretical foundation
  - Relatively effective
  - Very simple
  - Very fast in learning and classification
  - Fast to update when new training examples become available

- **Cons:**
  - Multinomial model / independence assumption are clearly wrong for text
  - Typically performs worse than other methods in practice

Experimental Results for SPAM filtering

- **[Sahami et. al, 1998]**
  - 1789 e-mail messages, 1578 SPAM, 211 legitimate
  - Training set: 1538 messages
  - Test set: 251 messages

	Junk		Legitimate	
Feature Regime	Precision	Recall	Precision	Recall
Words only	97.1%	94.3%	87.7%	93.4%
Words + Phrases	97.6%	94.3%	87.8%	94.7%
Words + Phrases + Domain-Specific	100.0%	98.3%	96.2%	100.0%

Table 1: Classification results using various feature sets.

Experimental Results for SPAM filtering

- **[Michelakis et. al, 2004]**
  - 1099 e-mail messages, 618 legitimate, 481 SPAM
  - 10-fold cross-validation

	$\lambda = 1$			$\lambda = 9$		
	Pr	Re	*WAcc*	Pr	Re	*WAcc*
1–grams						
Naive Bayes	90.56	94.73	94.65	91.57	92.17	94.87
Flexible Bayes	95.55	89.89	95.15	98.88	74.63	97.76
LogitBoost	92.43	90.08	93.64	97.71	74.89	97.24
SVM	94.95	91.43	95.42	98.12	78.33	97.60
1/2/3–grams						
Flexible Bayes	92.98	91.89	93.89	97.43	81.36	96.91
SVM	94.73	91.70	95.05	98.70	76.40	97.67

Experimental Results for SPAM filtering

- **[Carreras et. al, 2001]**
  - 1099 e-mail messages, 618 legitimate, 481 SPAM
  - 10-fold cross-validation

	T	recall	precision	F_1	F_1^{max}
N. Bayes	-	83.98	95.11	89.19	-
D. Trees	-	89.81	88.71	89.25	-
Stumps	525	96.47	97.48	96.97	97.39
TreeBoost[1]	525	96.88	97.90	97.39	97.60
TreeBoost[2]	725	96.67	98.31	97.48	97.59
TreeBoost[3]	675	96.88	97.90	97.39	97.81
TreeBoost[4]	450	**97.09**	**98.73**	**97.90**	98.01
TreeBoost[5]	550	96.88	98.52	97.69	98.12

Summary

- **SPAM filtering can be viewed as a text classification problem**
- **Naïve Bayes is a simple and effective approach to address it**
  - Pros: simple, learning and updating is fast
  - Cons: the independence assumption is wrong
- **If you are willing to invest more into training, there are algorithms that outperform Naïve Bayes**
- **Often SPAM filtering is not just text classification**
  - Web SPAM
  - Click SPAM

References

- **[Sahami et. al., 1998]** Sahami, M., Dumais, S., Heckerman, D., Horvitz, E. *A Bayesian Approach to Filtering Junk E-mail*. AAAI-1998 Workshop on Learning for Text Categorization.
- **[Michelakis et. al., 2004]** Michelakis, E., Androutsopoulos, I., Paliouras, G., Sakkis, G., Stamatopoulos, P. *Filtron: A Learning-Based Anti-Spam Filter*. CEAS-2004.
- **[Carreras et. al., 2001]** Carreras, X., Marquez, L. *Boosting Trees for Anti-Spam E-mail Filtering*. RANLP-2001.

Part 3

Machine Learning for Information Extraction

Pavel Dmitriev, Mikhail Bilenko

Some slides provided by Ray Mooney and Andrew Moore

Outline

- **Definition of Information Extraction**
- **Naïve Approach to Information Extraction**
- **Hidden Markov Models**
- **Experimental Results for Information Extraction**

Natural Language Processing (NLP)

- **An entire field focused on tasks involving syntactic, semantic, and pragmatic *analysis* of natural language text**
  - Part-of-speech tagging
  - Discourse analysis
  - Text summarization
  - Opinion extraction
  - Machine translation
- **Using machine learning methods for automating these tasks is a very active area of research**

Information Extraction (IE)

- **Identify specific pieces of information (data) in a unstructured or semi-structured textual document**
  - Transform unstructured information in a corpus of documents or web pages into a structured database
- **Can be applied to different types of text**
  - Newspaper articles, web pages, scientific articles, newsgroup messages, classified ads, medical notes, …
- **Can employ output of other Natural Language Processing tasks for enriching the text representation ("NLP features")**

Information Extraction

- **Given a piece of text extract values for specific fields**
  - Job postings from newsgroups and web pages
  - Position details and requirements from a job posting
  - Product name, specs, and prices from web pages
  - Appartment rental ads from e-mails
  - Biological information from journal articles

- **A basis for many web start-up's, and of great interest to the intelligence community (CIA, NSA)**

Papa johns pizza san francisco - Googl...

Web Images Video News Maps Gmail more ▼

Google Papa johns pizza san francisco Search Advanced Search Preferences

Web Maps

Papa Johns Pizza
www.papajohns.com Order a **Papa John's pizza** online. High-quality **pizza** delivered to you

Local business results for **papa johns pizza** near **San Francisco, CA**

A **Papa Johns Pizza** - **papajohns**.com
2782 Pinole Valley Rd, Pinole - (510) 758-8000
Directions and more »

B **Papa John's Pizza** - **papajohns**.com
30593 Union City Blvd, Union City - (510) 487-7272
Directions, menu, and more »

C **Papa John's Pizza** - **papajohns**.com
36488 Fremont Blvd, Fremont - (510) 791-7272
Directions, menu, and more »

More results near **San Francisco, CA** »

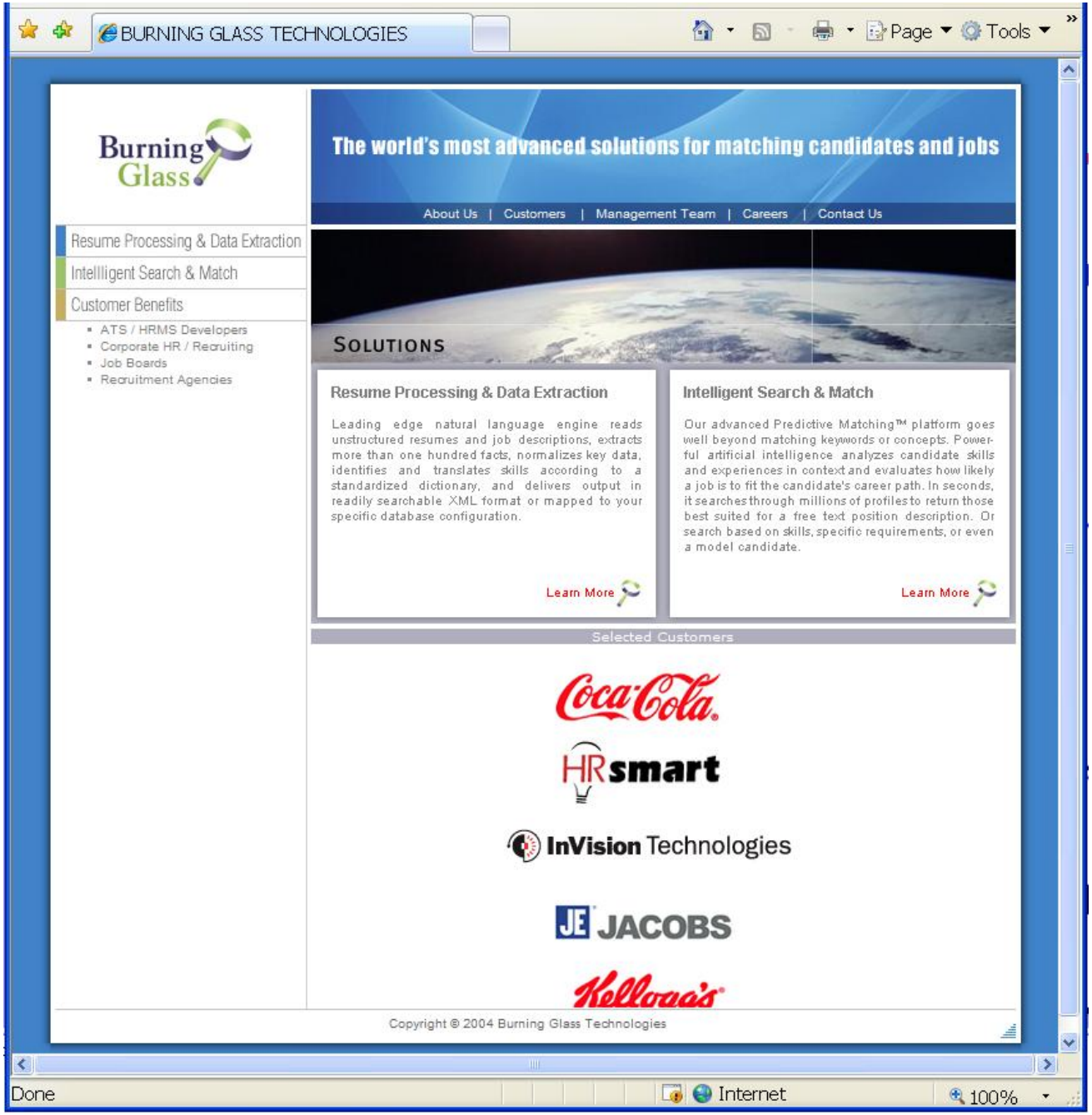
San Francisco Papa Johns Pizza
San Francisco, CA **Papa Johns Pizza**. **San Francisco**, CA Yellow Pages **... Papa Johns Pizza** in **San Francisco**, CA, Result 1 - 10 of 25 **...**
ziphip.com/**Papa+Johns+Pizza**.q.html - 50k - Cached - Similar pages

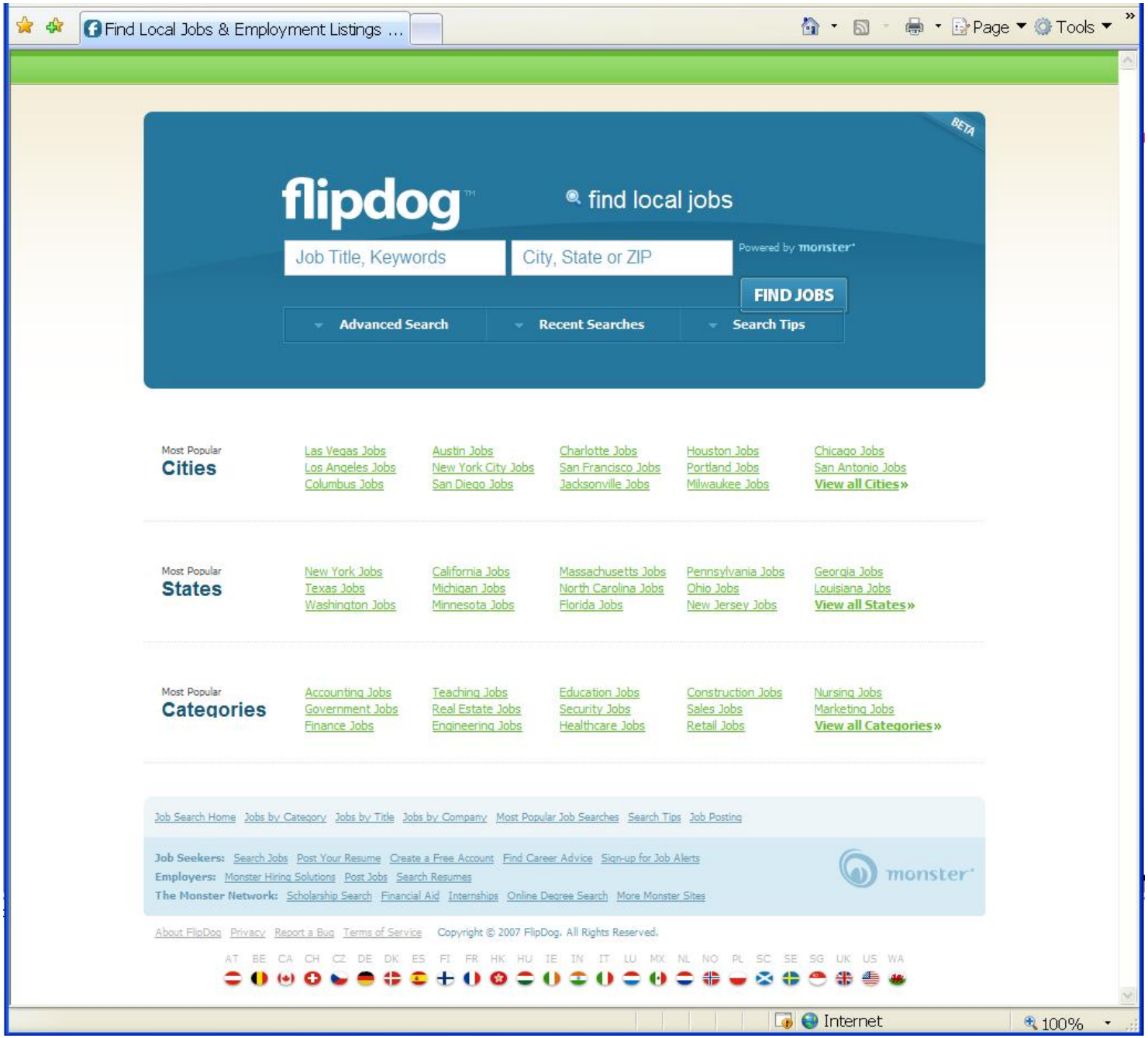
San Antonio **Papa Johns Pizza**
San Antonio, TX **Papa Johns Pizza**. **San** Antonio, TX Yellow Pages **... Papa Johns Pizza** in **San** Antonio, TX, Result 1 - 10 of 74 **...**
ziphip.com/**San**+Antonio--2C+TX/**Papa+Johns+Pizza**.zq.html - 53k - Cached - Similar pages
[More results from ziphip.com]

Papa John's - **Pizza** Franchise Opportunities | **Pizza** Marketplace
Pizza Franchise Opportunity - **San Francisco**/**San** Jose, California (CA) **...** Complete the form below to request information from **Papa John's** - **Pizza** Franchise **...**
www.**pizza**marketplace.com/storefronts.php?sf_id=56 - 34k - Cached - Similar pages

Papa John's Pizza Union City | Yelp
Papa John's Pizza Reviews - "Mmmm, mmm good!!! If you want something else other than the **...** Get weekly updates on what's Hot and New in **San Francisco**! **...**
www.yelp.com/biz/ihnFnzVq6w7ZW0_txN21uA - 28k - Cached - Similar pages

Papa John's Pizza Saint Paul | Yelp
Papa John's Pizza Reviews - "I have one question... how do they cook the **pizza** to get


BURNING GLASS TECHNOLOGIES
Page
Tools
Burning Glass
The world's most advanced solutions for matching candidates and jobs
About Us | Customers | Management Team | Careers | Contact Us
Resume Processing & Data Extraction
Intellligent Search & Match
Customer Benefits
ATS / HRMS Developers
Corporate HR / Recruiting
Job Boards
Recruitment Agencies
SOLUTIONS
Resume Processing & Data Extraction
Leading edge natural language engine reads unstructured resumes and job descriptions, extracts more than one hundred facts, normalizes key data, identifies and translates skills according to a standardized dictionary, and delivers output in readily searchable XML format or mapped to your specific database configuration.
Learn More
Intelligent Search & Match
Our advanced Predictive Matching™ platform goes well beyond matching keywords or concepts. Powerful artificial intelligence analyzes candidate skills and experiences in context and evaluates how likely a job is to fit the candidate's career path. In seconds, it searches through millions of profiles to return those best suited for a free text position description. Or search based on skills, specific requirements, or even a model candidate.
Learn More
Selected Customers
Coca-Cola
HRsmart
InVision Technologies
JE JACOBS
Kellogg's
Copyright © 2004 Burning Glass Technologies
Done
Internet
100%


Find Local Jobs & Employment Listings ...
Page
Tools
BETA
flipdog
find local jobs
Job Title, Keywords
City, State or ZIP
Powered by monster
FIND JOBS
Advanced Search
Recent Searches
Search Tips
Most Popular
Cities
Las Vegas Jobs
Los Angeles Jobs
Columbus Jobs
Austin Jobs
New York City Jobs
San Diego Jobs
Charlotte Jobs
San Francisco Jobs
Jacksonville Jobs
Houston Jobs
Portland Jobs
Milwaukee Jobs
Chicago Jobs
San Antonio Jobs
View all Cities»
Most Popular
States
New York Jobs
Texas Jobs
Washington Jobs
California Jobs
Michigan Jobs
Minnesota Jobs
Massachusetts Jobs
North Carolina Jobs
Florida Jobs
Pennsylvania Jobs
Ohio Jobs
New Jersey Jobs
Georgia Jobs
Louisiana Jobs
View all States»
Most Popular
Categories
Accounting Jobs
Government Jobs
Finance Jobs
Teaching Jobs
Real Estate Jobs
Engineering Jobs
Education Jobs
Security Jobs
Healthcare Jobs
Construction Jobs
Sales Jobs
Retail Jobs
Nursing Jobs
Marketing Jobs
View all Categories»
Job Search Home
Jobs by Category
Jobs by Title
Jobs by Company
Most Popular Job Searches
Search Tips
Job Posting
Job Seekers: Search Jobs Post Your Resume Create a Free Account Find Career Advice Sign-up for Job Alerts
Employers: Monster Hiring Solutions Post Jobs Search Resumes
The Monster Network: Scholarship Search Financial Aid Internships Online Degree Search More Monster Sites
monster
About FlipDog
Privacy
Report a Bug
Terms of Service
Copyright © 2007 FlipDog. All Rights Reserved.
AT BE CA CH CZ DE DK ES FI FR HK HU IE IN IT LU MX NL NO PL SC SE SG UK US WA
Internet
100%

Sample Job Posting

Subject: US-TN-SOFTWARE PROGRAMMER
Date: 17 Nov 1996 17:37:29 GMT
Organization: Reference.Com Posting Service
Message-ID: <56nigp$mrs@bilbo.reference.com>

SOFTWARE PROGRAMMER

Position available for Software Programmer experienced in generating software for PC-Based Voice Mail systems. Experienced in C Programming. Must be familiar with communicating with and controlling voice cards; preferable Dialogic, however, experience with others such as Rhetorix and Natural Microsystems is okay. Prefer 5 years or more experience with PC Based Voice Mail, but will consider as little as 2 years. Need to find a Senior level person who can come on board and pick up code with very little training. Present Operating System is DOS. May go to OS-2 or UNIX in future.

Please reply to:
Kim Anderson
AdNET
(901) 458-2888 fax
kimander@memphisonline.com

Sample Job Posting

Subject: **US**-**TN**-SOFTWARE PROGRAMMER
Date: **17 Nov 1996** 17:37:29 GMT
Organization: Reference.Com Posting Service
Message-ID: <**56nigp$mrs@bilbo.reference.com**>

SOFTWARE PROGRAMMER

Position available for Software Programmer experienced in generating software for PC-Based **Voice Mail** systems. Experienced in **C Programming**. Must be familiar with communicating with and controlling voice cards; preferable Dialogic, however, experience with others such as Rhetorix and Natural Microsystems is okay. Prefer **5** years or more experience with **PC** Based **Voice Mail**, but will consider as little as **2** years. Need to find a Senior level person who can come on board and pick up code with very little training. Present Operating System is **DOS**. May go to **OS-2** or **UNIX** in future.

Please reply to:
Kim Anderson
AdNET
(901) 458-2888 fax
kimander@memphisonline.com

Extracted Template

computer_science_job
id: 56nigp$mrs@bilbo.reference.com
title: SOFTWARE PROGRAMMER
salary:
company:
recruiter:
state: TN
city:
country: US
language: C
platform: PC \ DOS \ OS-2 \ UNIX
application:
area: Voice Mail
req_years_experience: 2
desired_years_experience: 5
req_degree:
desired_degree:
post_date: 17 Nov 1996

Outline

- Definition of Information Extraction
- **Naïve Approach to Information Extraction**
- Hidden Markov Models
- Experimental Results for Information Extraction

Text as Data

- **Representing documents: a continuum of richness**
  - **Vector-space**: text is a $|V|$-dimensional vector (V is vocabulary of all possible words), order is ignored ("bag-of-words")
  - **Sequence**: text is a string of contiguous tokens/characters
  - **Language-specific**: text is a sequence of contiguous tokens along with various syntactic, semantic, and pragmatic properties (e.g. part-of-speech features, semantic roles, discourse models)
- **Higher representation richness leads to higher computational complexity, more parameters to learn, etc., but may lead to higher accuracy**

Learning for IE

- **Given examples of labeled text, learn how to label tokens (or groups of tokens)**
- **Basic approach: token classification**
  - Treat each token as an isolated instance to be classified.
  - Features include token word, neighbors, capitalization, …
  - Use labeled data as a training set: fields to extract are positive examples, other tokens are negative examples

- **Example: biomedical text, protein name extraction**

O	*O*	*O*	*O*	*O*	*I*	*O*	*I*	*I*	*O*	*O*	…
to	map	the	interaction	of	PTHrP	with	importin	beta	using	a	…

IE via Token Classification

to	map	the	interaction	of	PTHrP	with	importin	beta	using	a	…
x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_8	x_9	x_{10}	…

- **Each token is represented by a feature vector**
- **Possible features:** {*token_value*, *is_dictionary_word*, *has_uppercase*, *ends_with_"-in"*, *is_noun*}
- **Task:** given training data, learn a classifier that labels every new t_i as either ***Inside*** or ***Outside***

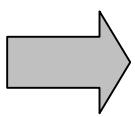
x_1={'to', T, F, F, F} y_1=*O*

…

x_6={'PTHrP', F, T, F, T} y_6=*I*

x_7={'with', T, F, F, F} y_7=*O*

x_8={'importin', F, F, T, T} y_8=*I*



x_i={''Cyclin", F, T, T, T}

y_i=?

Naïve Bayes for IE

- **Can use Naïve Bayes Classifier:**

$$p(y_i = I | x_i = \{x_{i1}, \ldots, x_{i5}\}) = \frac{p(y_i{=}I) \prod_{k=1}^{5} p(x_{ik}|y_i{=}I)}{\prod_{k=1}^{5} p(x_{ik})}$$

- **For individual features, probabilities can be obtained from training data sequences:**
  - $p(y_i{=}I) = 0.1$
  - $p(x_{i1}{=}\text{'PTHrP'} \mid y_i{=}I) = 0.9$, $p(x_{i1}{=}\text{'beta'} \mid y_i{=}I) = 0.7$, …
  - $p(x_{i2}{=}T \mid y_i{=}I) = 0.4$, $p(x_{i3}{=}T \mid y_i{=}I) = 1.0$, …

Shortcomings of Single-Token Classification

- **Natural language has very rich structure (syntax, semantics, topical structure, …)**
  - Many dependencies exist between words within the sentence
  - "Myopic" classification that considers one token a time is ignoring the dependencies

- **In many IE tasks, fields of interest are composed of *several adjacent tokens* ("Cornell University", "cyclin D1")**
  - Labels of adjacent tokens are *related,* labeling decisions should be made *collectively*

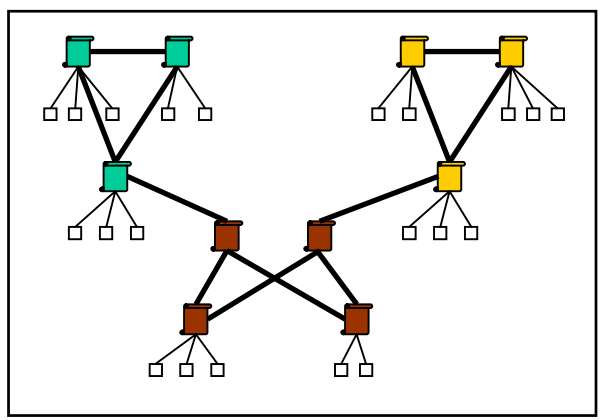
Outline

- Definition of Information Extraction
- Naïve Approach to Information Extraction
- **Hidden Markov Models**
- Experimental Results for Information Extraction

Relational Learning and Graphical Models

- **Want to learn a classifier that would account for relationships between the tokens**
- ***Graphical models* provide an intuitive and principled framework**
  - Instances are nodes, features are attributes of nodes
  - Edges encode dependencies between instance labels and features

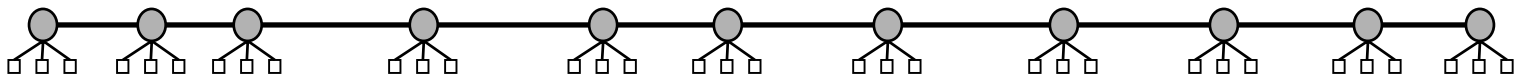
- **Example: web page classification**
  - Nodes = pages
  - Edges = hyperlinks
  - Attributes = words, etc.



Relational Learning for IE

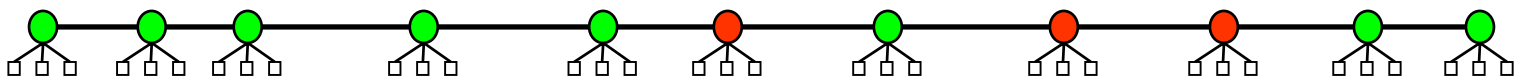
- **Strongest dependencies in text are between adjacent words**

to map the interaction of PTHrP with importin beta using a …



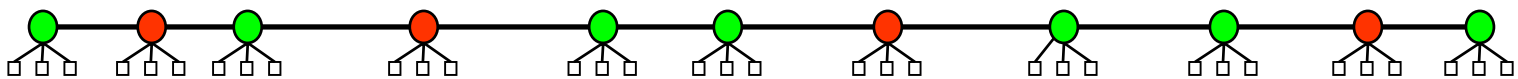
- **Labeling task: find "best" label configuration:**

y'



vs.

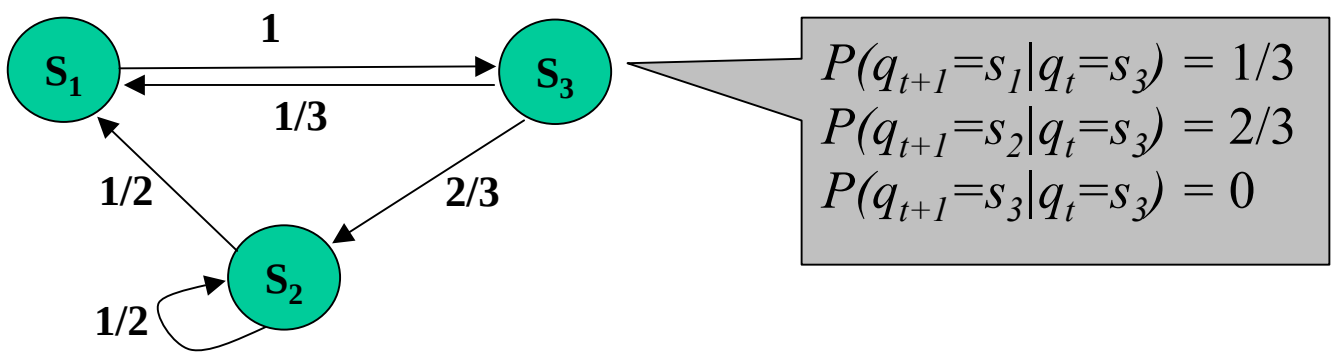
y''



- **What is "best" configurations?**
  - Need to define it, Hidden Markov Models is one option

Markov System

- **A Markov System has N states $s_1, \ldots, s_N$**
- **There are discrete timesteps $t=0$, $t=1, \ldots$**
- **On the tth timestep the system is in exactly one of the available states. Call it q_t**
- **Between each timestep, the next state is chosen randomly**
- **The current state determines the probability distribution for the next state**

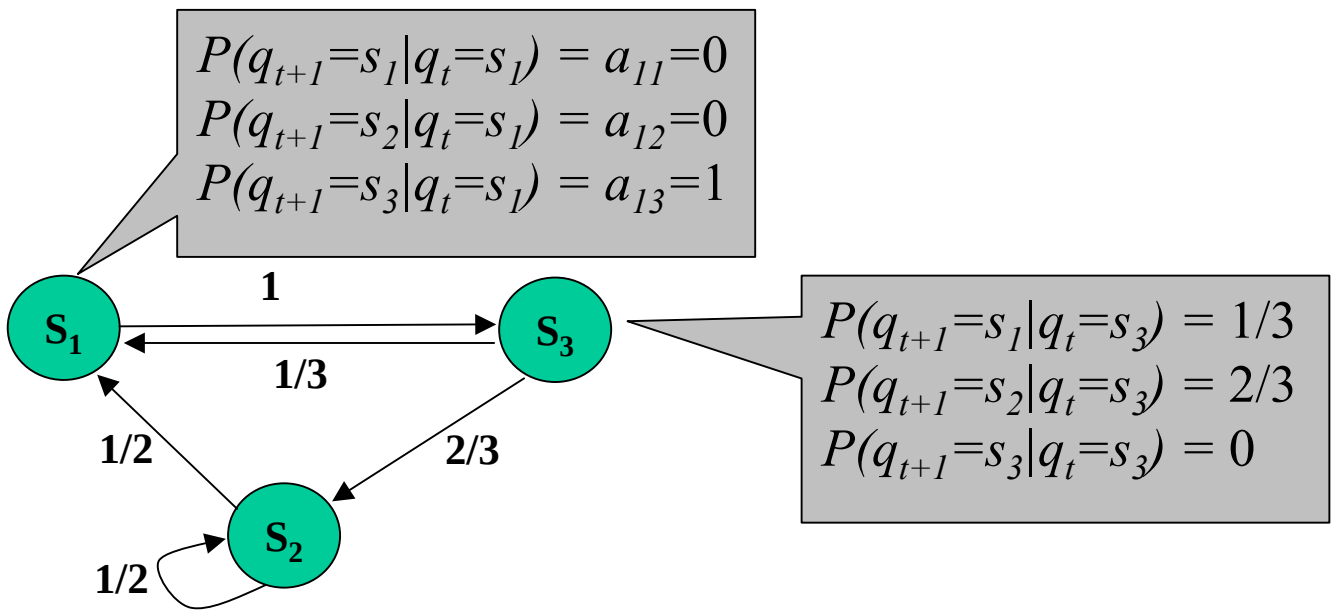


Markov Property

- **Markov property:** q_{t+1} **is conditionally independent of** $\{q_{t-1}, q_{t-2}, \ldots, q_0\}$ **given** q_t**. In other words**

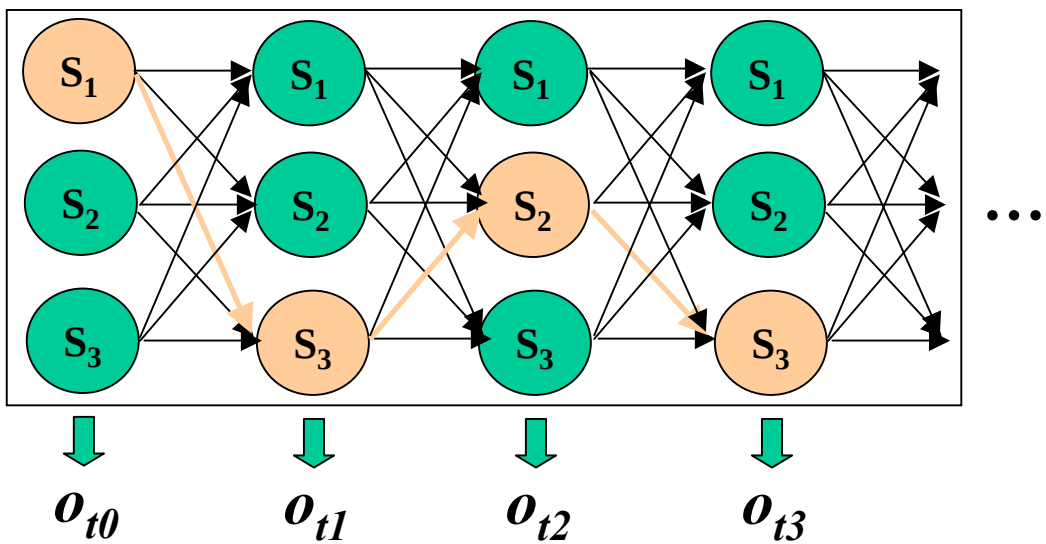
$$P(q_{t+1} = s_j \mid q_t = s_i) = P(q_{t+1} = s_j \mid q_t = s_i, \text{ any earlier history})$$

- **Notation:** $P(q_{t+1}=s_j|q_t=s_i) = a_{ij}$



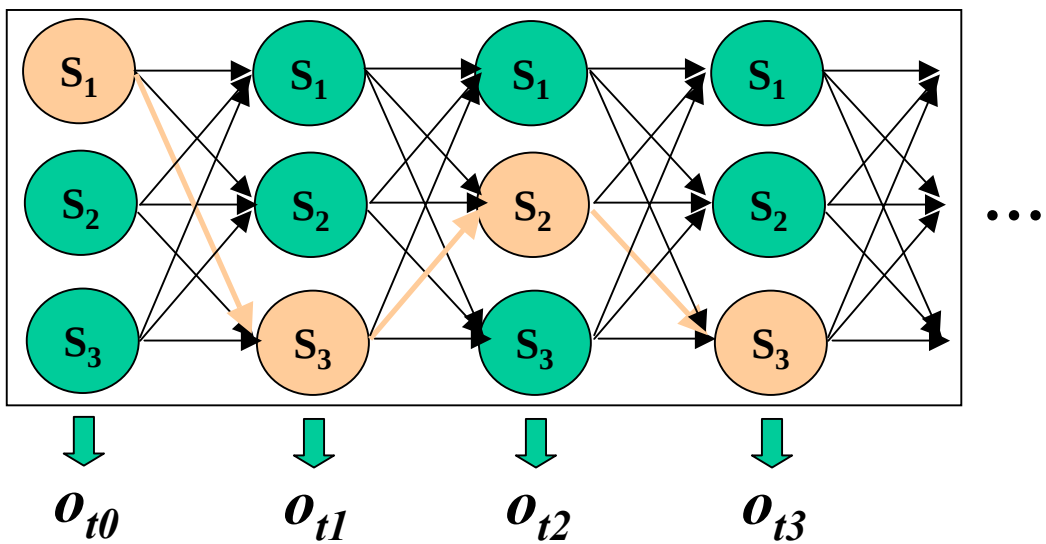
Hidden Markov Model (HMM)

- **A Markov System satisfying the Markov Property**
- **The states s_i are hidden**
- **At every state s_i one of the symbols $\{o_1, \ldots o_M\}$ is observed with probability $b_j(i)$, $j=1,\ldots,M$**



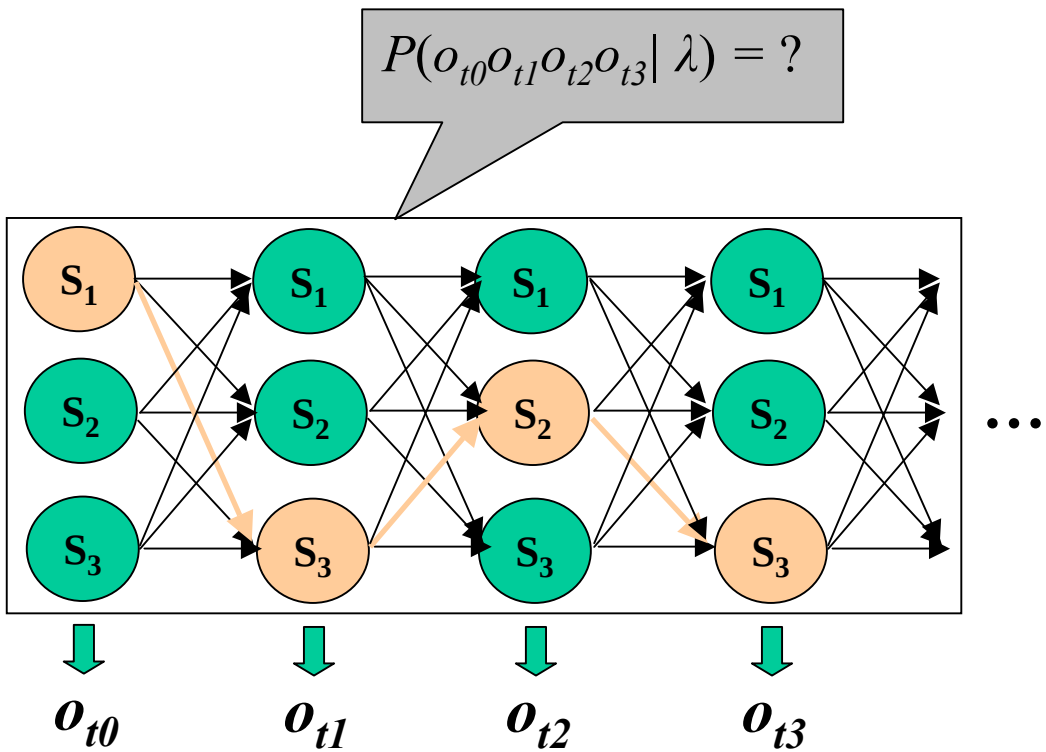
Formal Definition of HMM

- **An HMM λ is a 5-tuple consisting of**
  - N states
  - M possible observations
  - A = $\{a_{ij}\}$, matrix of state transition probabilities
  - B = $\{b_k(i)\}$, matrix of observation probabilities
  - $\pi = \{\pi_i\}$, the starting state probabilities



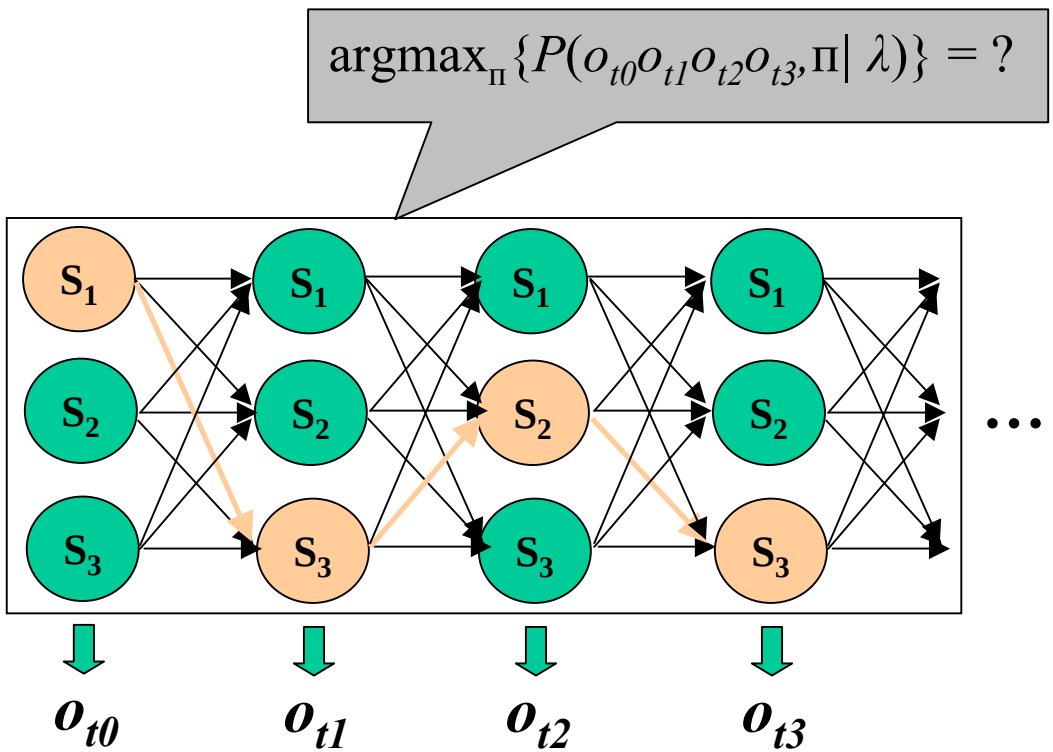
Central Problems in HMM Modelling

- **Problem 1: Evaluation**
  - Given an HMM and a sequence of observations, what is the probability of the HMM generating this sequence?



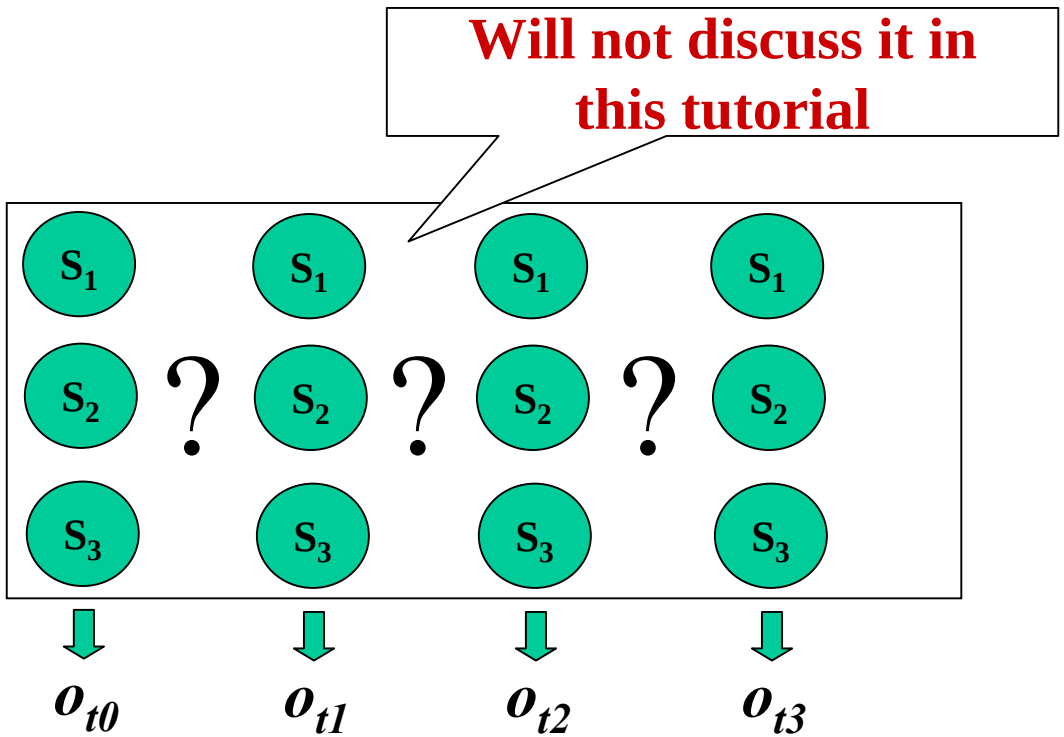
Central Problems in HMM Modelling

- **Problem 2: Decoding**
  - Given an HMM and a sequence of observations, what is the most probable path that could generate this sequence?



Central Problems in HMM Modelling

- **Problem 3: Learning**
  - Given a sequence of observations,what is the maximum likelihood HMM that could have produce that sequence?



Problem 1: Evaluation

- **Given an HMM λ and a sequence of observations O, what is the probability of the HMM generating this sequence?**
- **Naïve inefficient approach**

$$P(O \mid \lambda) = \sum_{\text{all paths } Q} P(O \mid Q, \lambda) * P(Q \mid \lambda)$$

- **Complexity:** $O(N^T * T)$

Problem 1: Evaluation

- **Given an HMM λ and a sequence of observations O, what is the probability of the HMM generating this sequence?**
- **Efficient Dynamic Programming approach**
  - Define $\alpha_t(i) = P(o_1, o_2, \ldots, o_t \ \& \ q_t = s_i \mid \lambda)$
  - $\alpha_t(i)$ is the probability that, in a random trial, we would have seen the first t observations, and we would have ended up at state s_i on step t
- **Can define $\alpha_t(i)$ recursively**

$$\alpha_1(i) = P(o_1 \ \& \ q_1 = s_i) = \pi_i * b_{o_1}(i)$$

$$\alpha_{t+1}(j) = P(o_1 o_2 \ldots o_{t+1} \ \& \ q_{t+1} = s_j) = \sum_{i=1}^{N} \alpha_t(i) * a_{ij} * b_{o_{t+1}}(j)$$

Problem 1: Evaluation

- **Given an HMM λ and a sequence of observations O, what is the probability of the HMM generating this sequence?**
- **Efficient Dynamic Programming approach**

$$\alpha_1(i) = P(o_1 \,\&\, q_1 = s_i) = \pi_i * b_{o_1}(i)$$

$$\alpha_{t+1}(j) = P(o_1 o_2 \ldots o_{t+1} \,\&\, q_{t+1} = s_j) = \sum_{i=1}^{N} \alpha_t(i) * a_{ij} * b_{o_{t+1}}(j)$$

- **Then the probability of observation sequence is**

$$P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

- **Complexity:** $O(N^2 * T)$

Problem 2: Decoding

- **Given an HMM λ and a sequence of observations O, what is the most probable path that could generate this sequence?**
- **Again, a Dynamic Programming approach, known as Viterbi Algorithm**
  - Want to compute $\arg\max_Q P(Q \mid O, \lambda)$
  - Define $\partial_t(i) = \max_{q_1, q_2, \ldots, q_{t-1}} P(q_1, q_2, \ldots, q_{t-1} \,\&\, q_t = s_i \,\&\, o_1 o_2 \ldots o_t)$
  - $\delta_t(i)$ = the probability of the path of length t-1 with the maximum chance of occuring, ending up in state s_i, and producing output $o_1 o_2 \ldots o_t$
  - Define $mpp_t(i)$ = that path

Problem 2: Decoding

- **Given an HMM λ and a sequence of observations O, what is the most probable path that could generate this sequence?**
- **Again a Dynamic Programming approach, known as Viterbi Algorithm**
  - Can define $\delta_t(i)$ and $mpp_t(i)$ recursively

$$\partial_1(i) = \pi_i * b_{o_1}(i)$$

$$mpp_t(i) = s_i$$

$$i^* = \arg\max_i \partial_t(i) * a_{ij} * b_{o_{t+1}}(j)$$

$$\partial_{t+1}(j) = \partial_t(i^*) * a_{ij} * b_{o_{t+1}}(j)$$

$$mpp_{t+1}(j) = mpp_t(i^*) s_{i^*}$$

Problem 2: Decoding

- **Given an HMM λ and a sequence of observations O, what is the most probable path that could generate this sequence?**
- **Again a Dynamic Programming approach, known as Viterbi Algorithm**
  - Then the most probable path is

$$q_T^* = mpp_T(\arg\max_{1 \le i \le N} \partial_T(i))$$

$$q_{t-1}^* = mpp_t(q_t^*)$$

  - And its corresponding probability is

$$P^* = \max_{1 \le i \le N} \partial_T(i)$$

  - Complexity: $O(N^2 * T)$

HMM for IE

- **Observations are words**
- **Hidden states are labels we want to find**
- **Finding the maximum likelyhood label configuration is solving the "decoding" problem**
- **Learning the transition and emission probabilities is solving the "learning" problem**
  - The Baum-Welch algorithm (an iterative Expectation-Maximization procedure)

Richer Models

- **HMMs only model dependencies between adjacent states (word labels)**
- **Model the joint distribution** $P(Q,O)$

- **Conditional Random Fields (CRFs) are models which allow for arbitrary dependencies between states as well as between features**
- **Model the conditional distribution** $P(Q|O)$
- **Have superior accuracy on IE and other similar tasks**
- **However, they are slower to train and do not scale well to large amounts of training data**

Summary

- **Information Extraction is an old problem which gained importance with the growing popularity of the Web**
- **Simple approaches such as Naïve Bayes do not work well due to strong dependencies between word labels**
- **HMMs, more sophisticated generative models which allow accounting for dependencies between adjacent word labels perform much better**
- **Discriminative models, such as CRFs, which directly optimize the desired property and allow for arbitrary dependencies perfrom even better**

References

- **[Rabiner, 1989]** Rabiner, L.R. *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition*, Proceedings of IEEE, Vol. 77, No. 2, pp. 257-286, 1989.

Part 4

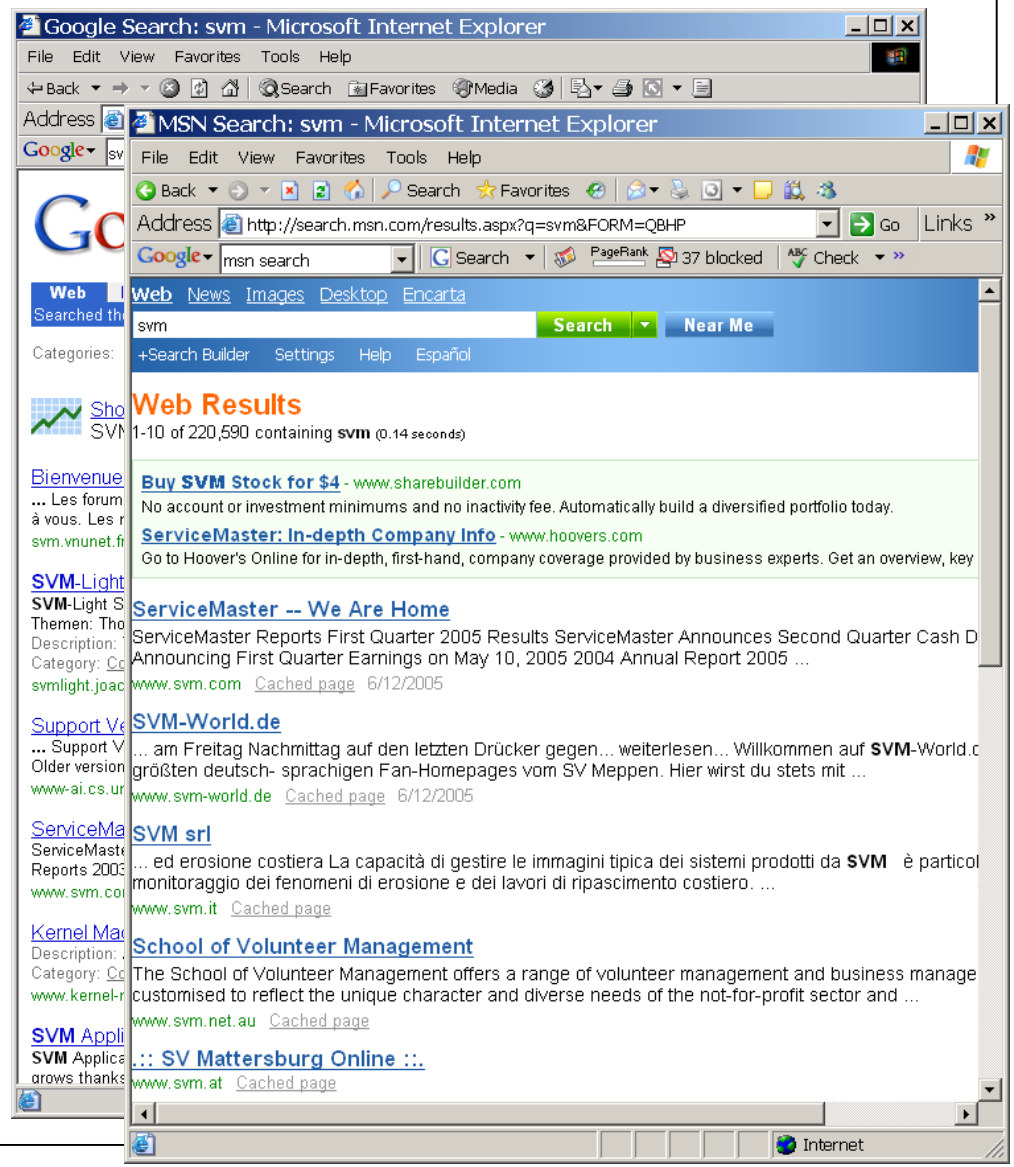
Machine Learning in Web Search

Pavel Dmitriev, Mikhail Bilenko

Some slides provided by Thorsten Joachims

Adaptive Search Engines

- **Current Search Engines**
  - One-size-fits-all
  - Hand-tuned retrieval function
- **Hypothesis**
  - Different users need different retrieval functions
  - Different collections need different retrieval functions
- **Machine Learning**
  - Learn improved retrieval functions
  - User Feedback as training data

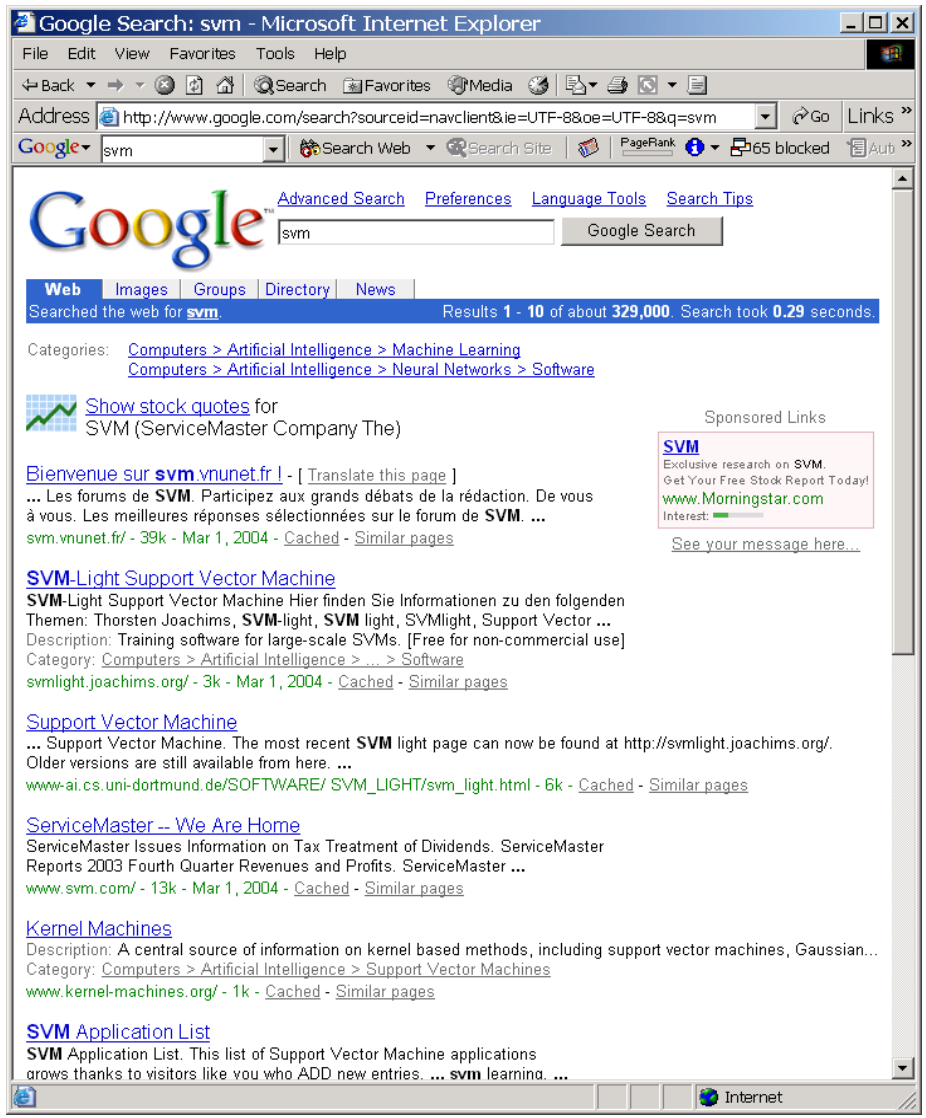


Outline

- **How can we get training data for learning improved retrieval functions?**
  - Explicit vs. implicit feedback
  - Absolute vs. relative feedback
  - User study with eye-tracking and relevance judgments
- **What learning algorithms can use this training data?**
  - Ranking Support Vector Machine
  - User study with meta-search engine

Sources of Feedback

- **Explicit Feedback**
  - Overhead for user
  - Only few users give feedback
    => not representative
- **Implicit Feedback**
  - Queries, clicks, time, mousing, scrolling, etc.
  - No Overhead
  - More difficult to interpret



Types of Feedback

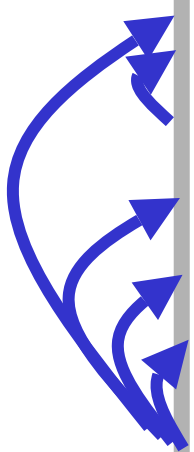
- **Absolute Feedback**
  - Feedback about relevance of document on absolute scale
  - Examples
    - Document d_i is relevant to query q
    - Document d_j is not relevant to query q
    - Document d_l has relevance 0.73 with respect to query q
- **Relative Feedback**
  - Feedback reflects preference between documents
  - Examples
    - Document d_i is more relevant to query q than document d_j
    - Document d_i is the least relevant to query q among $\{d_i, d_j, d_l, d_m\}$

Feedback from Clickthrough Data

Relative Feedback:
Clicks reflect preference between observed links.

Absolute Feedback:
The clicked links are relevant to the query.

$(3 < 2)$,
$(7 < 2)$,
$(7 < 4)$,
$(7 < 5)$,
$(7 < 6)$



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

Rel(1),
NotRel(2),
Rel(3),
NotRel(4),
NotRel(5),
NotRel(6),
Rel(7)

Is Implicit Feedback Reliable?

How do users view the results?

- How many abstracts do users evaluate before clicking?
- Do users scan abstracts from top to bottom?
- Do users view all abstracts above a click?
- Do users look below a clicked abstract?

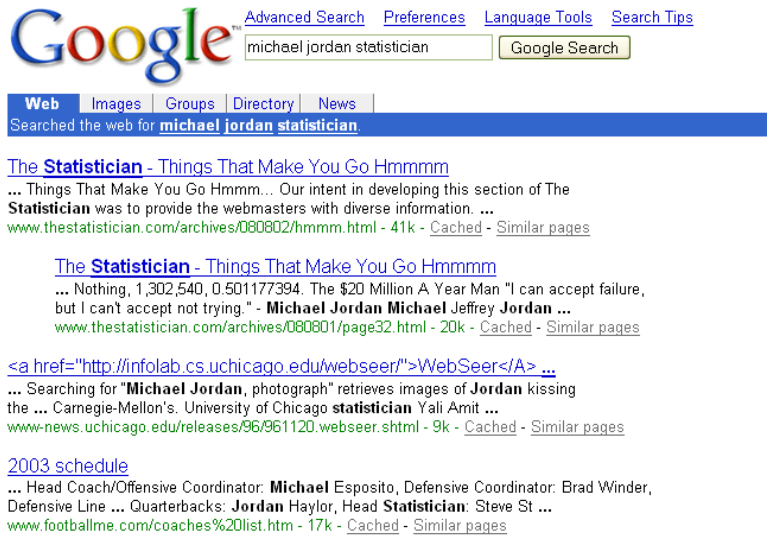
How do clicks relate to relevance?

- Absolute Feedback:
  Are clicked links relevant?
  Are not clicked links not relevant?
- Relative Feedback:
  Are clicked links more relevant than not clicked links?

1. Kernel Machines
   http://www.kernel-machines.org/
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 SVMs
   http://www.support-vector.net/
5. *Support Vector Machine and ...*
   http://svm.bell-labs.com/SVMrefs.html
6. Archives of SUPPORT-VECTOR...
   http://www.jisc.ac.uk/lists/SUPPORT...
7. Lucent Technologies: SVM demo applet
   http://svm.bell-labs.com/SVMsvt.html
8. Royal Holloway SVM
   http://svm.dcs.rhbnc.ac.uk
9. SVM World
   http://www.svmworld.com
10. Fraunhofer FIRST SVM page
    http://svm.first.gmd.de

User Study: Eye-Tracking and Relevance

- **Scenario**
  - WWW search
  - Google search engine
  - Subjects were not restricted
  - Answer 10 questions
- **Eye-Tracking**
  - Record the sequence of eye movements
  - Analyze how users scan the results page of Google
- **Relevance Judgements**
  - Ask relevance judges to explicitly judge the relevance of all pages encountered
  - Compare implicit feedback from clicks to explicit judgments



What is Eye-Tracking?

Eye tracking device

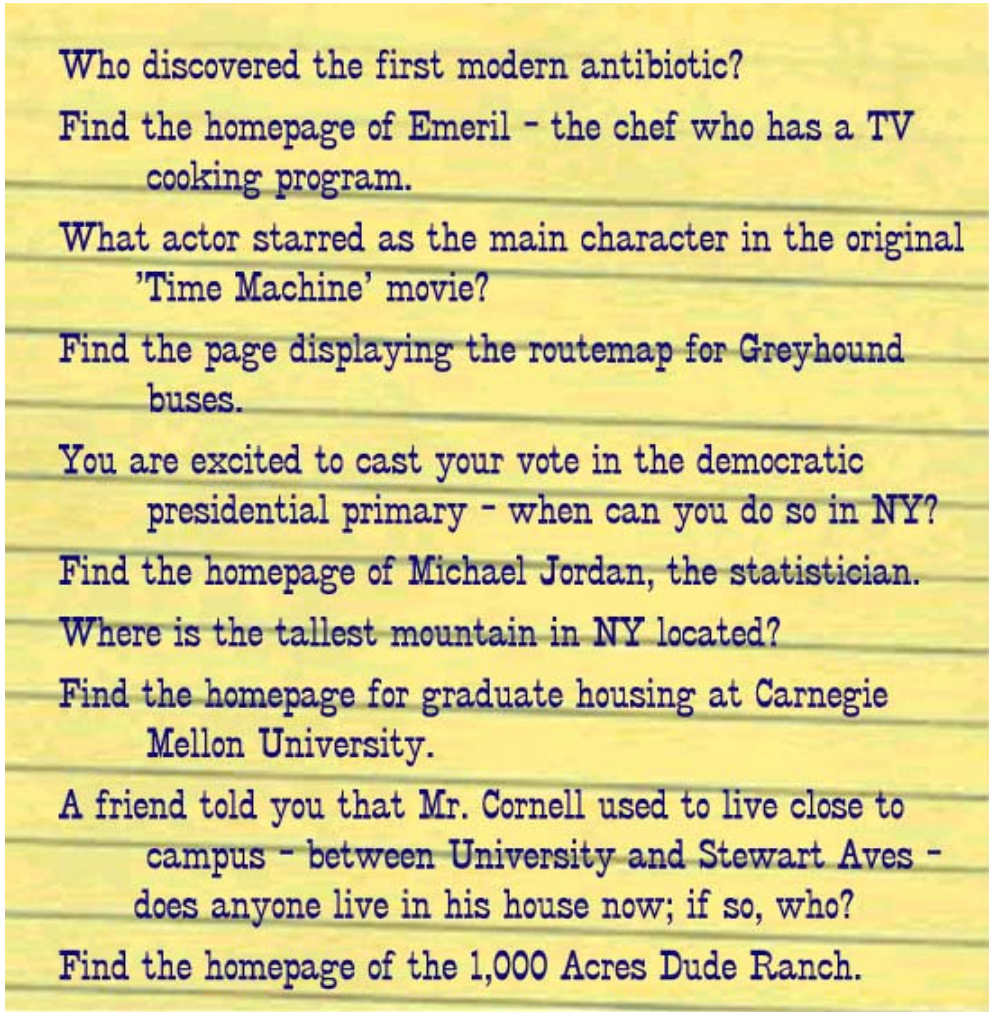
Device to detect and record where and what people look at

- **Fixations**: ~200-300ms; information is acquired
- **Saccades**: extremely rapid movements between fixations
- **Pupil dilation**: size of pupil indicates interest, arousal

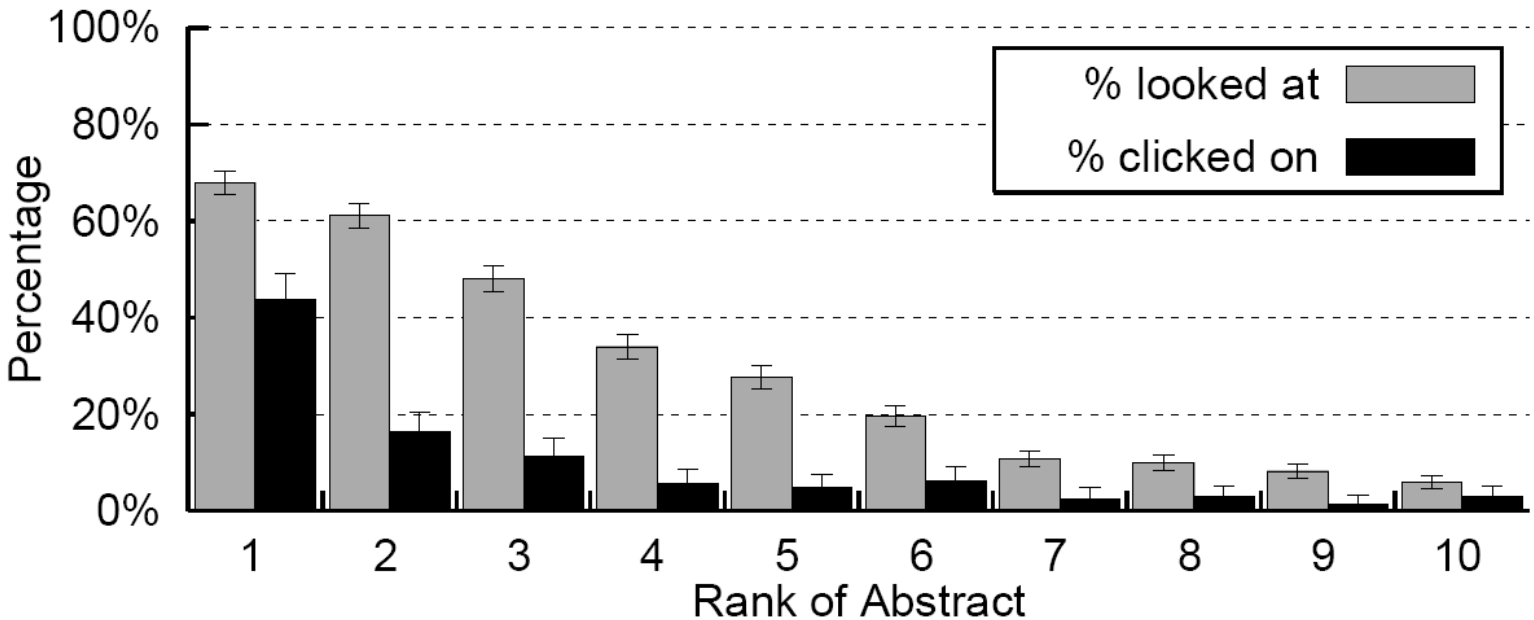
"Scanpath" output depicts pattern of movement throughout screen. Black markers represent fixations.

Experiment Setup

- **Study (Phase I)**
  - 36 subjects
  - Undergraduate students
  - Familiar with Google
- **10 Questions**
  - Balanced informational and navigational
- **Task**
  - Answer questions
  - Start with Google search, no restrictions
  - Users unaware of study goal

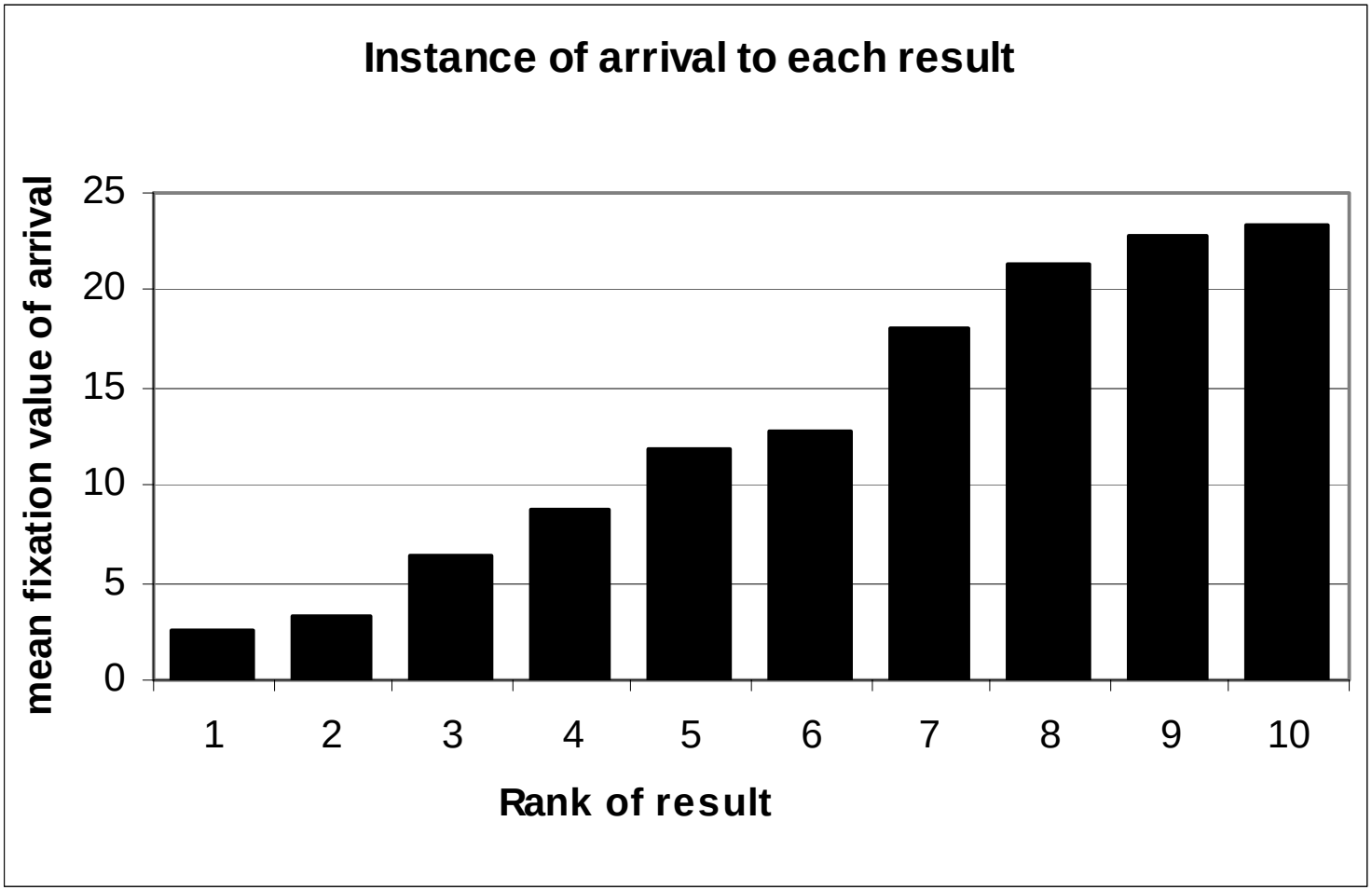


How often do Users View/Click on an Abstract?



=> Top ranked results are viewed/clicked substantially more often

In Which Order are the Results Viewed?



=> Users tend to read the results in order

Do Users Look Below the Clicked Link?

Viewed Rank	Clicked Rank 1	2	3	4	5	6
1	90.6%	76.2%	73.9%	60.0%	54.5%	45.5%
2	56.8%	90.5%	82.6%	53.3%	63.6%	54.5%
3	30.2%	47.6%	95.7%	80.0%	81.8%	45.5%
4	17.3%	19.0%	47.8%	93.3%	63.6%	45.5%
5	8.6%	14.3%	21.7%	53.3%	100.0%	72.7%
6	4.3%	4.8%	8.7%	33.3%	18.2%	81.8%

=> Users typically do not look at links below before they click (except maybe the next link)

Conclusion: Viewing Behavior

- Users most frequently view two abstracts
- Users typically view results in order from top to bottom
- Users view links one and two more thoroughly and often
- Users click most frequently on link one
- Users typically do not look at links below before they click (except maybe the next link)

=> **Design strategies for interpreting clickthrough data that respect these properties!**

How do Clicks Relate to Relevance?

- **Experiment (Phase II)**
  - Additional 16 subjects
  - Manually judged relevance
    - Abstract
    - Page
- **Manipulated Rankings**
  - **Normal**: Google's ordering
  - **Swapped**: Top Two Swapped
  - **Reversed**: Ranking reversed
- **Experiment Setup**
  - Same as Phase I
  - Manipulations not detectable

1. Kernel Machines
   http://www.kernel-machines.org/
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 SVMs
   http://www.support-vector.net/
5. *Support Vector Machine and ...*
   http://svm.bell-labs.com/SVMrefs.html
6. Archives of SUPPORT-VECTOR...
   http://www.jisc.ac.uk/lists/SUPPORT...
7. Lucent Technologies: SVM demo applet
   http://svm.bell-labs.com/SVMsvt.html
8. Royal Holloway SVM
   http://svm.dcs.rhbnc.ac.uk
9. SVM World
   http://www.svmworld.com
10. Fraunhofer FIRST SVM page
    http://svm.first.gmd.de

Presentation Bias

Hypothesis: ~~Order of presentation influences where users look, but not where they click!~~

"normal"	l_1^-, l_2^-	l_1^+, l_2^-	l_1^-, l_2^+	l_1^+, l_2^+	total
$rel(l_1) > rel(l_2)$	15	19	1	1	36
$rel(l_1) < rel(l_2)$	11	5	2	2	20
$rel(l_1) = rel(l_2)$	19	9	1	0	29
total	45	33	4	3	85

"swapped"	l_1^-, l_2^-	l_1^+, l_2^-	l_1^-, l_2^+	l_1^+, l_2^+	total
$rel(l_1) > rel(l_2)$	11	15	1	1	28
$rel(l_1) < rel(l_2)$	17	10	7	2	36
$rel(l_1) = rel(l_2)$	36	11	3	0	50
total	64	36	11	3	114

Quality-of-Context Bias

Hypothesis: ~~**Clicking depends only on the link itself, but not on other links.**~~

	Rank of clicked link as sorted by relevance judges
Normal + Swapped	**2.67**
Reversed	**3.27**

=> Users click on less relevant links, if they are embedded between irrelevant links.

Are Clicks Absolute Relevance Judgments?

- **Clicks depend not only on relevance of a link, but also**
  - On the position in which the link was presented
  - The quality of the other links

=> Interpreting Clicks as absolute feedback is extremely difficult!

Strategies for Generating Relative Feedback

Strategies

- "Click > Skip Above"
  - **(3>2), (5>2), (5>4)**
- "Last Click > Skip Above"
  - **(5>2), (5>4)**
- "Click > Earlier Click"
  - **(3>1), (5>1), (5>3)**
- "Click > Skip Previous"
  - **(3>2), (5>4)**
- "Click > Skip Next"
  - **(1>2), (3>4), (5>6)**

1. Kernel Machines
   http://www.kernel-machines.org/
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 SVMs
   http://www.support-vector.net/
5. *Support Vector Machine and ...*
   http://svm.bell-labs.com/SVMrefs.html
6. Archives of SUPPORT-VECTOR...
   http://www.jisc.ac.uk/lists/SUPPORT...
7. Lucent Technologies: SVM demo applet
   http://svm.bell-labs.com/SVMsvt.html
8. Royal Holloway SVM
   http://svm.dcs.rhbnc.ac.uk
9. SVM World
   http://www.svmworld.com
10. Fraunhofer FIRST SVM page
    http://svm.first.gmd.de

Comparison with Explicit Feedback

Explicit Feedback Data Strategy	Abstracts Phase I "normal"
Inter-Judge Agreement	89.5
Click > Skip Above	80.8 ± 3.6
Last Click > Skip Above	83.1 ± 3.8
Click > Earlier Click	67.2 ± 12.3
Click > Skip Previous	82.3 ± 7.3
Click > No Click Next	84.1 ± 4.9

=> All but "Click > Earlier Click" appear accurate

Conclusions: Implicit Feedback

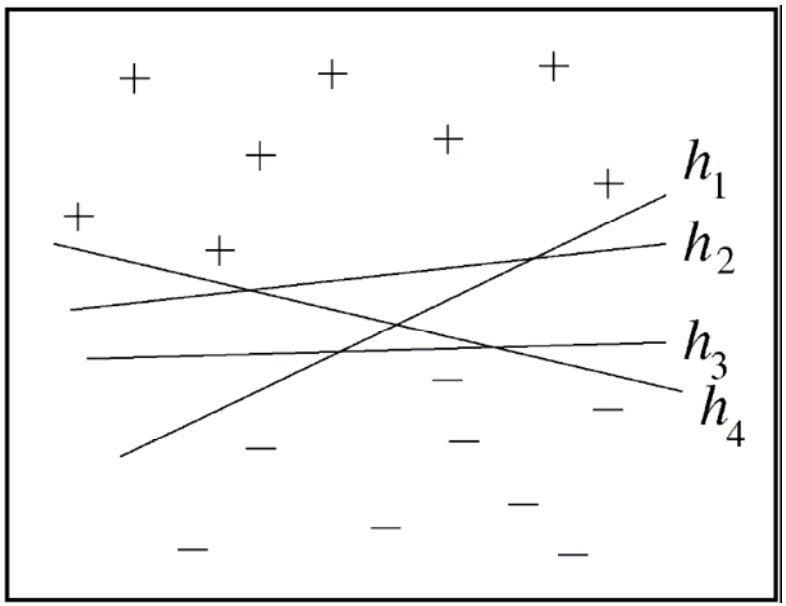
- **Interpreting clicks as absolute feedback is difficult**
  - Presentation Bias
  - Quality-of-Context Bias
- **Relative preferences derived from clicks are accurate**
  - “Click > Skip Above”
  - “Last Click > Skip Above”
  - “Click > Skip Previous”

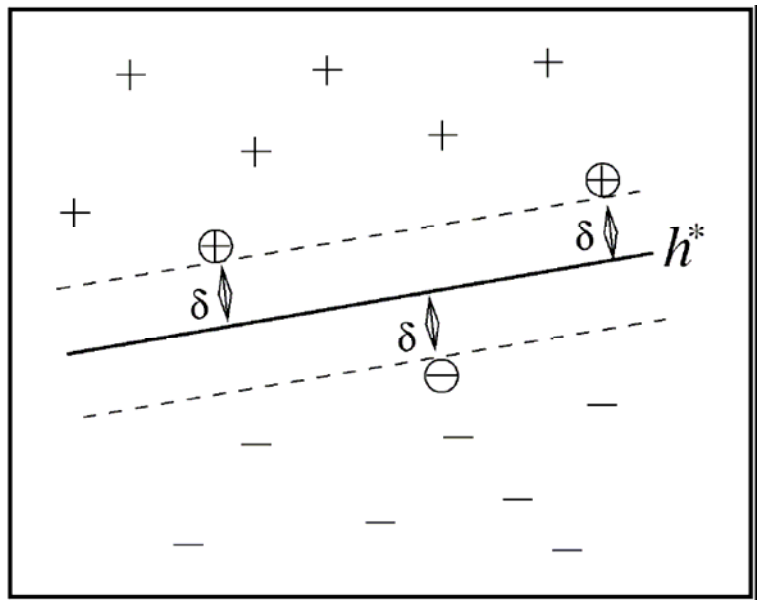
Outline

- **How can we get training data for learning improved retrieval functions?**
  - Explicit vs. implicit feedback
  - Absolute vs. relative feedback
  - User study with eye-tracking and relevance judgments
- **What learning algorithms can use this training data?**
  - Ranking Support Vector Machine
  - User study with meta-search engine

Optimal Hyperplanes
Linear Hard-Margin Support Vector Machine

Assumption: Training examples are linearly separable.

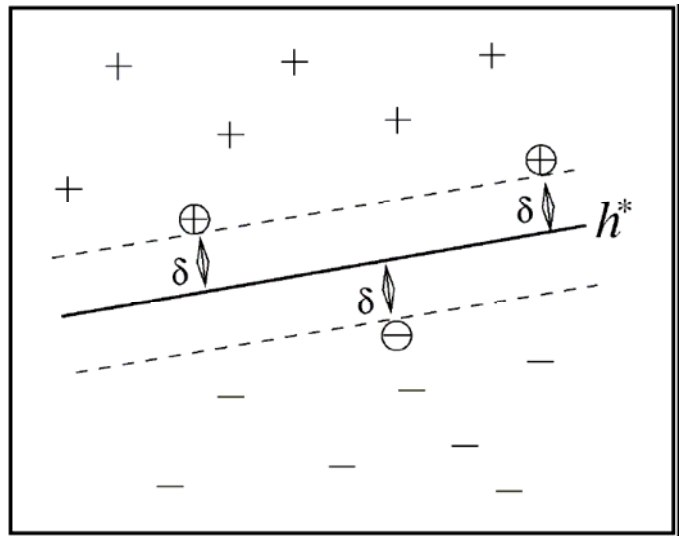




The Optimization Problem

Requirement 1: Zero Training Error

$$\begin{cases} y_1(x_1 w + b) > 0 \\ \dots \\ y_n(x_n w + b) > 0 \end{cases}$$



Requirement 2: Maximum Margin

$$\max_{w,b} \delta, \text{ where } \delta = \min_i \left| \frac{1}{\|w\|}(x_i w + b) \right|$$

The Optimization Problem

Requirements 1 and 2 together:

$$\begin{cases} \max_{w,b} \delta \\ \forall i: \; y_i\left(\frac{1}{\|w\|}(x_i w + b)\right) \geq \delta \end{cases}$$

Choosing $\|w\| = 1/\delta$, get:

$$\begin{cases} \min_{w,b} \frac{1}{2} ww \\ \forall i: \; y_i(x_i w + b) \geq 1 \end{cases}$$

Hard-Margin Separation

Goal: Find hyperplane with the largest distance to the closest training examples.

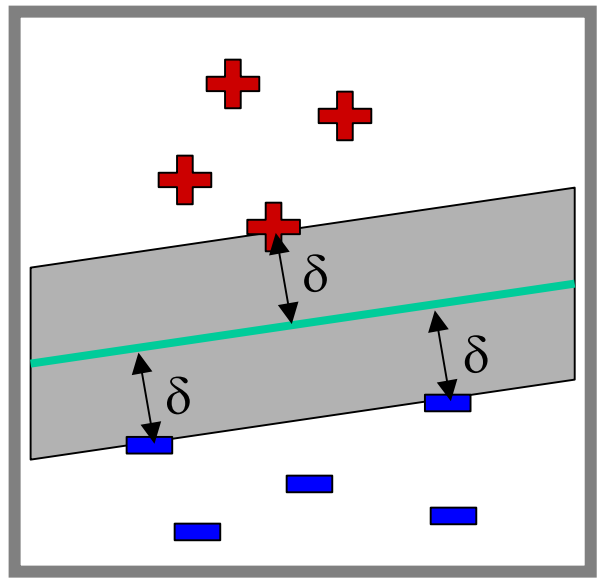
Optimization Problem (Primal):

$$\min_{\vec{w},b} \quad \frac{1}{2}\vec{w}\cdot\vec{w}$$

$$s.t. \quad y_1(\vec{w}\cdot\vec{x}_1 + b) \geq 1$$

$$\ldots$$

$$y_n(\vec{w}\cdot\vec{x}_n + b) \geq 1$$



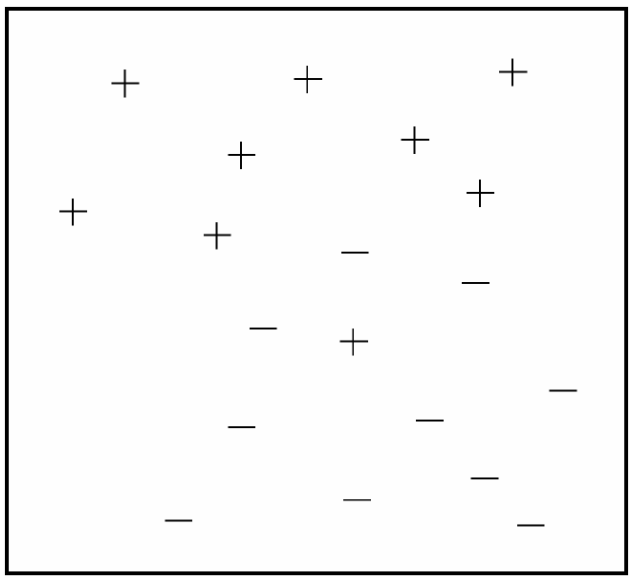
Support Vectors: Examples with minimal distance (i.e. margin).

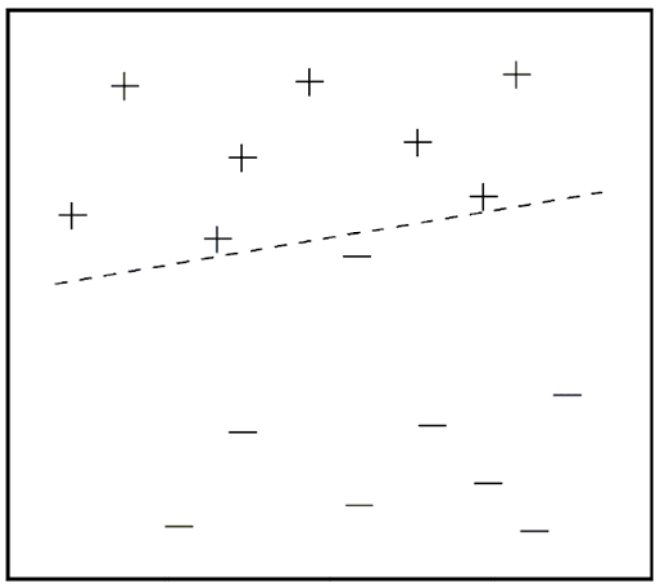
For a new example, classify it according to the $sign(wx_i+b)$

Non-Separable Training Data

Limitations of hard-margin formulation

- For some training data, there is no separating hyperplane.
- Complete separation (i.e. zero training error) can lead to suboptimal prediction error.





Soft-Margin Separation

Idea: Maximize margin and minimize training error.

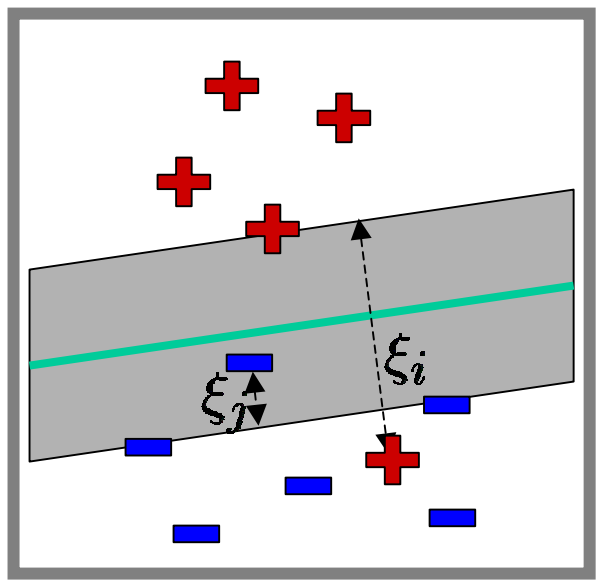
Hard-Margin OP (Primal):

$$\min_{\vec{w},b} \quad \frac{1}{2}\vec{w}\cdot\vec{w}$$
$$s.t. \quad y_1(\vec{w}\cdot\vec{x}_1+b)\geq 1$$
$$\ldots$$
$$y_n(\vec{w}\cdot\vec{x}_n+b)\geq 1$$

Soft-Margin OP (Primal):

$$\min_{\vec{w},\vec{\xi},b} \frac{1}{2}\vec{w}\cdot\vec{w}+C\sum_{i=1}^{n}\xi_i$$
$$s.t.\ y_1(\vec{w}\cdot\vec{x}_1+b)\geq 1-\xi_1 \wedge \xi_1\geq 0$$
$$\ldots$$
$$y_n(\vec{w}\cdot\vec{x}_n+b)\geq 1-\xi_n \wedge \xi_n\geq 0$$

- Slack variable ξ_i measures by how much (x_i,y_i) fails to achieve margin δ
- $\Sigma\xi_i$ is upper bound on number of training errors
- C is a parameter that controls trade-off between margin and training error.



Controlling Soft-Margin Separation

- $\Sigma\xi_i$ is upper bound on number of training errors
- C is a parameter that controls trade-off between margin and training error.

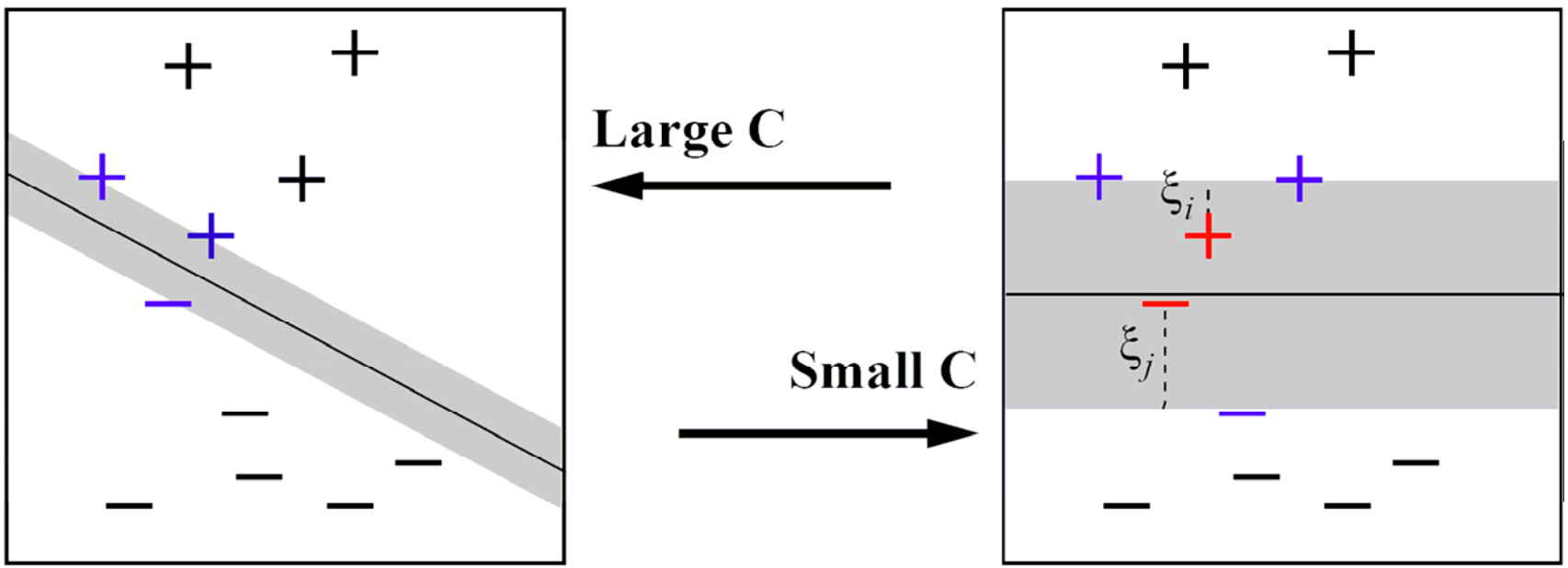
Soft-Margin OP (Primal):

$$\min_{\vec{w},\vec{\xi},b} \frac{1}{2}\vec{w}\cdot\vec{w} + C\sum_{i=1}^{n}\xi_i$$

$$s.t.\ y_1(\vec{w}\cdot\vec{x}_1 + b) \geq 1-\xi_1 \wedge \xi_1 \geq 0$$

$$\ldots$$

$$y_n(\vec{w}\cdot\vec{x}_n + b) \geq 1-\xi_n \wedge \xi_n \geq 0$$



Dual SVM Optimization Problem

- **Primal Optimization Problem**

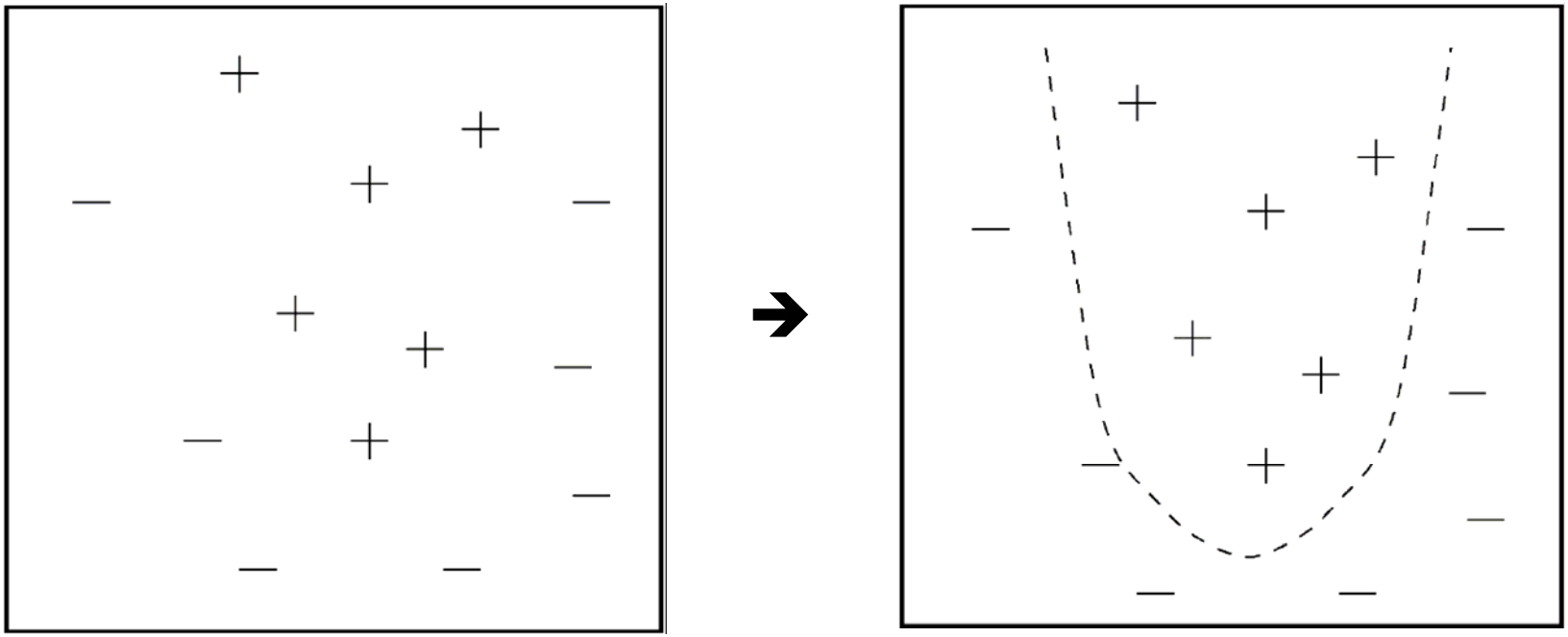
$$\begin{aligned} &\text{minimize:} && P(\vec{w}, b, \vec{\xi}) = \frac{1}{2}\vec{w}\cdot\vec{w} + C\sum_{i=1}^{n}\xi_i \\ &\text{subject to:} && \forall_{i=1}^{n} : y_i[\vec{w}\cdot\vec{x}_i + b] \geq 1 - \xi_i \\ & && \forall_{i=1}^{n} : \xi_i > 0 \end{aligned}$$

- **Dual Optimization Problem**

$$\begin{aligned} &\text{maximize:} && D(\vec{\alpha}) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j (\vec{x}_i \cdot \vec{x}_j) \\ &\text{subject to:} && \sum_{i=1}^{n} y_i \alpha_i = 0 \\ & && \forall_{i=1}^{n} : 0 \leq \alpha_i \leq C \end{aligned}$$

- **Theorem:** If w^* is the solution of the Primal and α^* is the solution of the Dual, then $\vec{w}^* = \sum_{i=1}^{n} \alpha_i^* y_i \vec{x}_i$

Non-Linear Problems



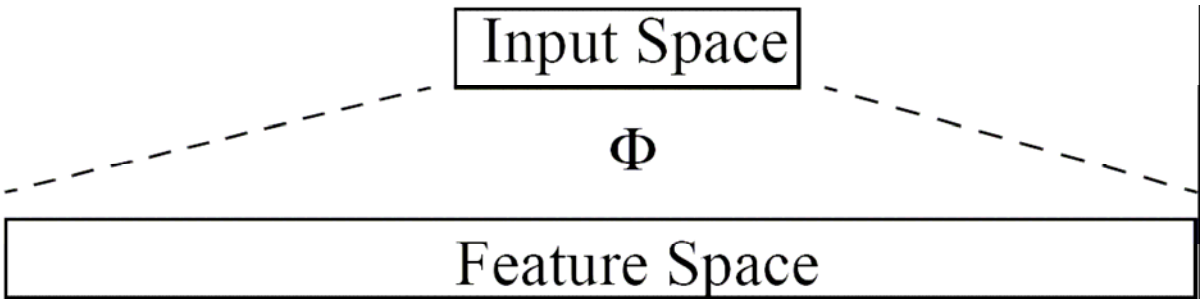
Problem:

- some tasks have non-linear structure
- no hyperplane is sufficiently accurate

How can SVMs learn non-linear classification rules?

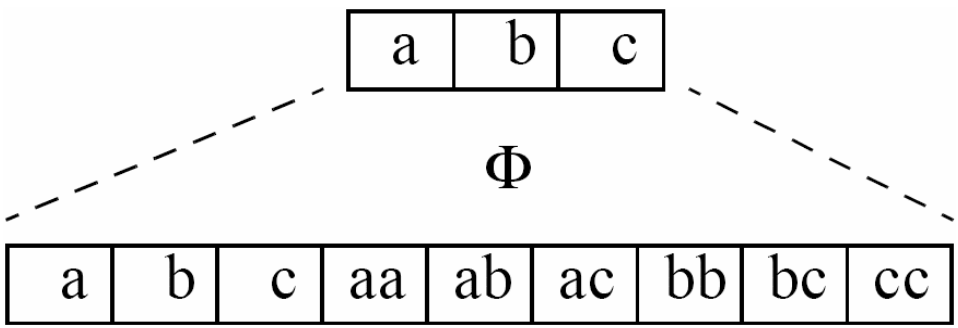
Extending the Hypothesis Space

Idea: add more features



➔ Learn linear rule in feature space.

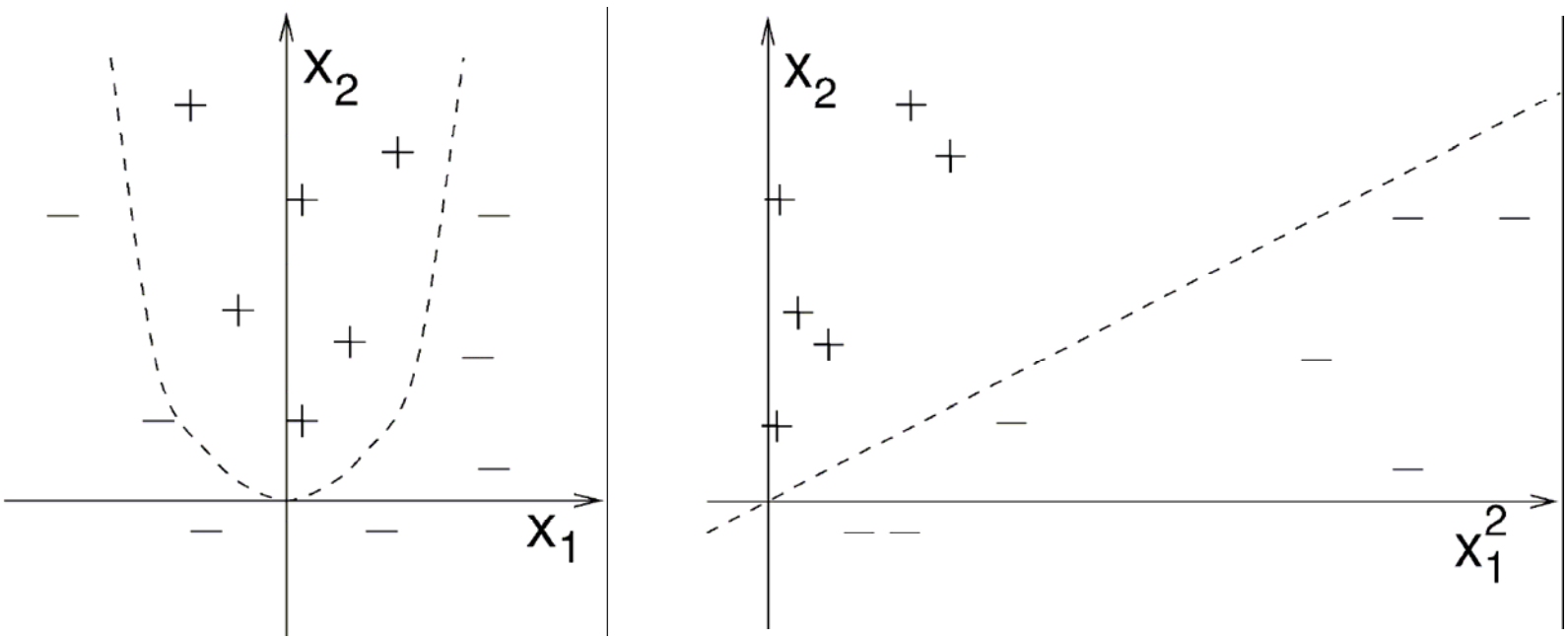
Example:



➔ The separating hyperplane in feature space is degree two polynomial in input space.

Example

- **Input Space:** $\vec{x} = (x_1, x_2)$**(2 attributes)**
- **Feature Space:** $\Phi(\vec{x}) = (x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2, 1)$
  (6 attributes)



SVM with Kernel

Training:

maximize: $$D(\vec{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j K(\vec{x}_i, \vec{x}_j)$$

subject to: $$\sum_{i=1}^{n} y_i \alpha_i = 0$$

$$\forall_{i=1}^{n} : 0 \leq \alpha_i \leq C$$

Classification:

$$h(\vec{x}) = sign\left(\left[\sum_{i=1}^{n} \alpha_i y_i \Phi(\vec{x}_i)\right] \cdot \Phi(\vec{x}) + b\right)$$

$$= sign\left(\sum_{i=1}^{n} \alpha_i y_i K(\vec{x}_i, \vec{x}) + b\right)$$

New hypotheses spaces through new Kernels:

- **Linear:** $K(\vec{a}, \vec{b}) = \vec{a} \cdot \vec{b}$
- **Polynomial:** $K(\vec{a}, \vec{b}) = [\vec{a} \cdot \vec{b} + 1]^d$
- **Radial Basis Function:** $K(\vec{a}, \vec{b}) = exp(-\gamma[\vec{a} - \vec{b}]^2)$
- **Sigmoid:** $K(\vec{a}, \vec{b}) = tanh(\gamma[\vec{a} \cdot \vec{b}] + c)$

Back to the Problem of Learning from Clickthrough Data

Training Data: preferences in the form $(q, d_i) > (q, d_j)$

Idea: Learn a ranking function, so that number of violated pair-wise training preferences is minimized.

Form of Ranking Function:

$$\begin{aligned} rsv(q,d_i) &= w_1 * (\#\text{of query words in title of } d_i) \\ &\quad + w_2 * (\#\text{of query words in anchortext}) \\ &\quad + \ldots \\ &\quad + w_n * (\text{page-rank of } d_i) \\ &= w * \Phi(q,d_i) \end{aligned}$$

Training: Select w so that

if user prefers d_i to d_j for query q, then

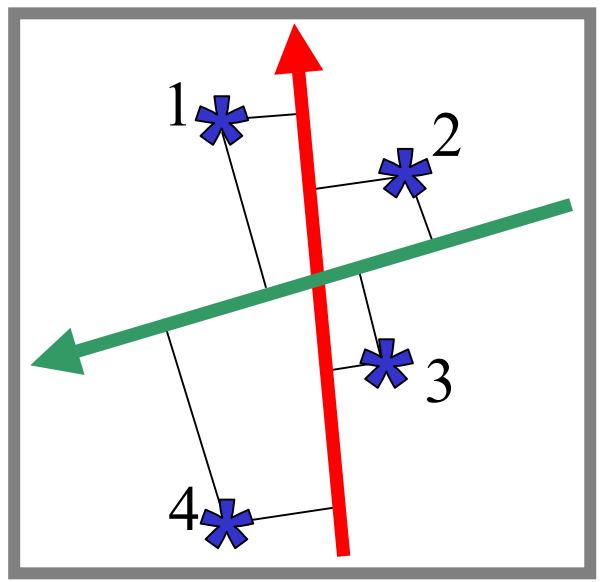
$rsv(q, d_i) > rsv(q, d_j)$

Ranking Support Vector Machine

- **Find ranking function with low error and large margin**

$$
\begin{aligned}
min \quad & \frac{1}{2}\vec{w}\cdot\vec{w} + C\sum_{i,j,k}\xi_{kij} \\
s.t. \quad & \vec{w}\cdot\Phi(q_1,d_i) \geq \vec{w}\cdot\Phi(q_1,d_j) + 1 - \xi_{1ij} \\
& \ldots \\
& \vec{w}\cdot\Phi(q_n,d_i) \geq \vec{w}\cdot\Phi(q_n,d_j) + 1 - \xi_{nij}
\end{aligned}
$$

- **Properties**
  - Convex quadratic program
  - Can learn non-linear functions using Kernels



Experiment

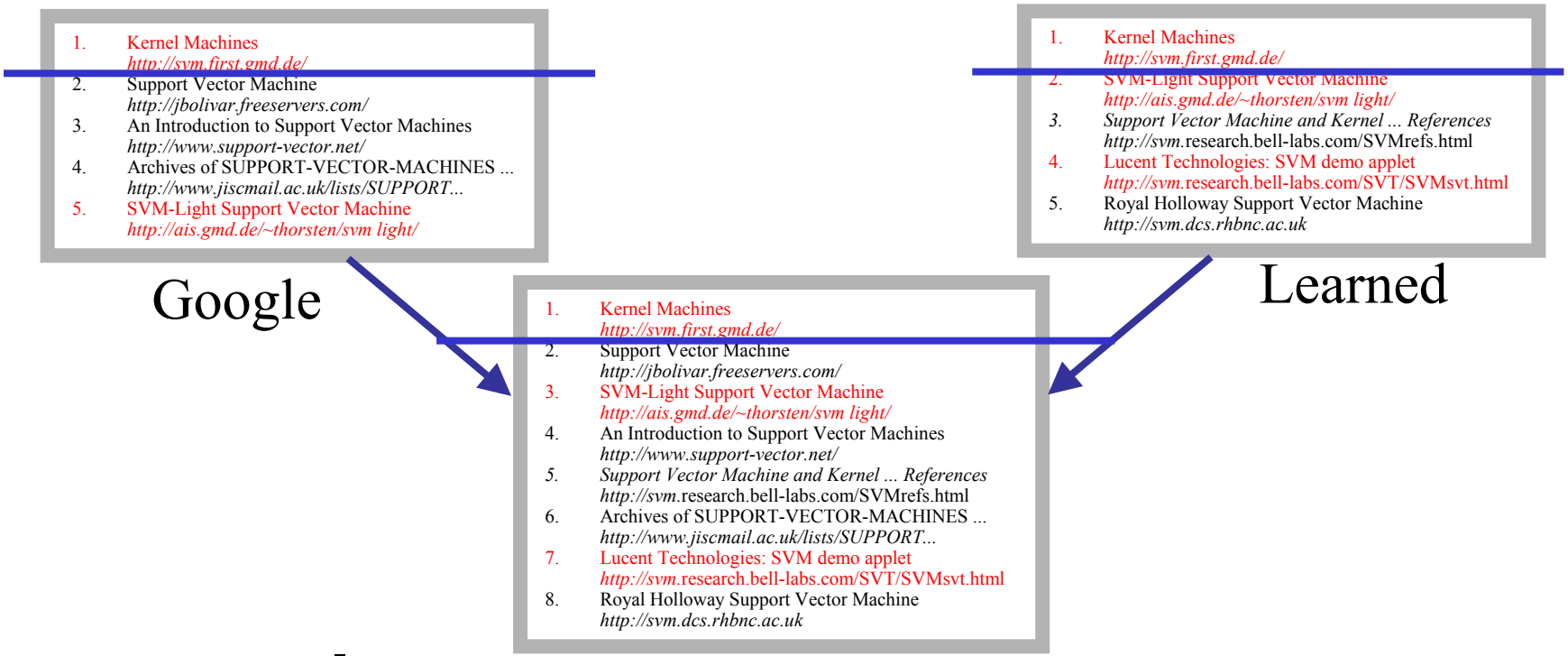
Meta-Search Engine "Striver"

- Implemented meta-search engine on top of Google, MSNSearch, Altavista, Hotbot, Excite
- Retrieve top 100 results from each search engine
- Re-rank results with learned ranking functions

Experiment Setup

- User study on group of ~20 German machine learning researchers and students
  => homogeneous group of users
- Asked users to use the system like any other search engine
- Train ranking SVM on 3 weeks of clickthrough data
- Test on 2 following weeks

Which Ranking Function is Better?



- **Approach**
  - Combine the rankings in a "fair and unbiased" way: at each position in the combined ranking #links from Learned equals # links from Google plus/minus 1
  - See which links users prefer

Results

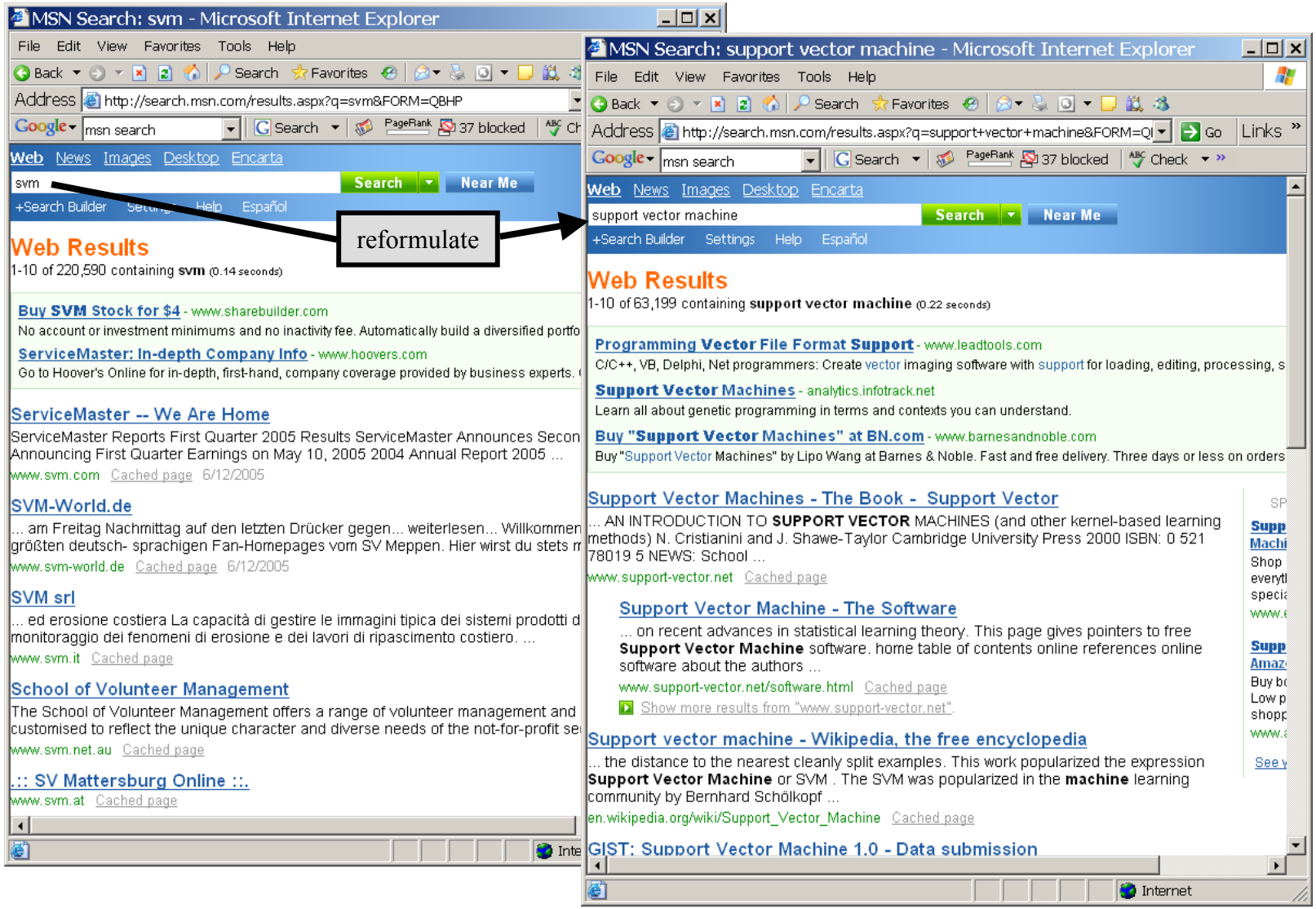
Ranking A	Ranking B	A better	B better	Tie	Total
Learned	Google	29	13	27	69
Learned	MSNSearch	18	4	7	29
Learned	Toprank	21	9	11	41

Result:

- Learned > Google
- Learned > MSNSearch
- Learned > Toprank

Toprank: rank top 1 results from all 5 search engines first, then top 2, etc.

Feedback across Query Chains [KDD 2005]



Summary

- **Clickthrough data can provide accurate feedback**
  - Clickthrough provides relative instead of absolute judgments
- **Ranking SVM can learn effectively from relative preferences**
  - Improved retrieval through personalization in meta search
- **Interesting directions for future work**
  - Exploiting query chains
  - Robustness to “click-spam”
  - Learning theory for interactive learning with preference

References

- **[Joachims, 2005]** Joachims, T., Granka, L., Pang, B., Hembrooke, H., Gay, G. *Accurately Interpreting Clickthrough Data as Implicit Feedback*. SIGIR-2005.
- **[Joachims, 2002]** Joachims, T. *Optimizing Search Engines Using Clickthrough Data*. KDD-2002.
- **[Radlinski, 2005]** Radlinski, F., Joachims, T. *Query Chains: Learning to Rank from Implicit Feedback*. KDD-2005.

- **More Information, Papers, and Software**
  - `http://www.joachims.org`

Part 5

Machine Learning in the News

Pavel Dmitriev, Mikhail Bilenko

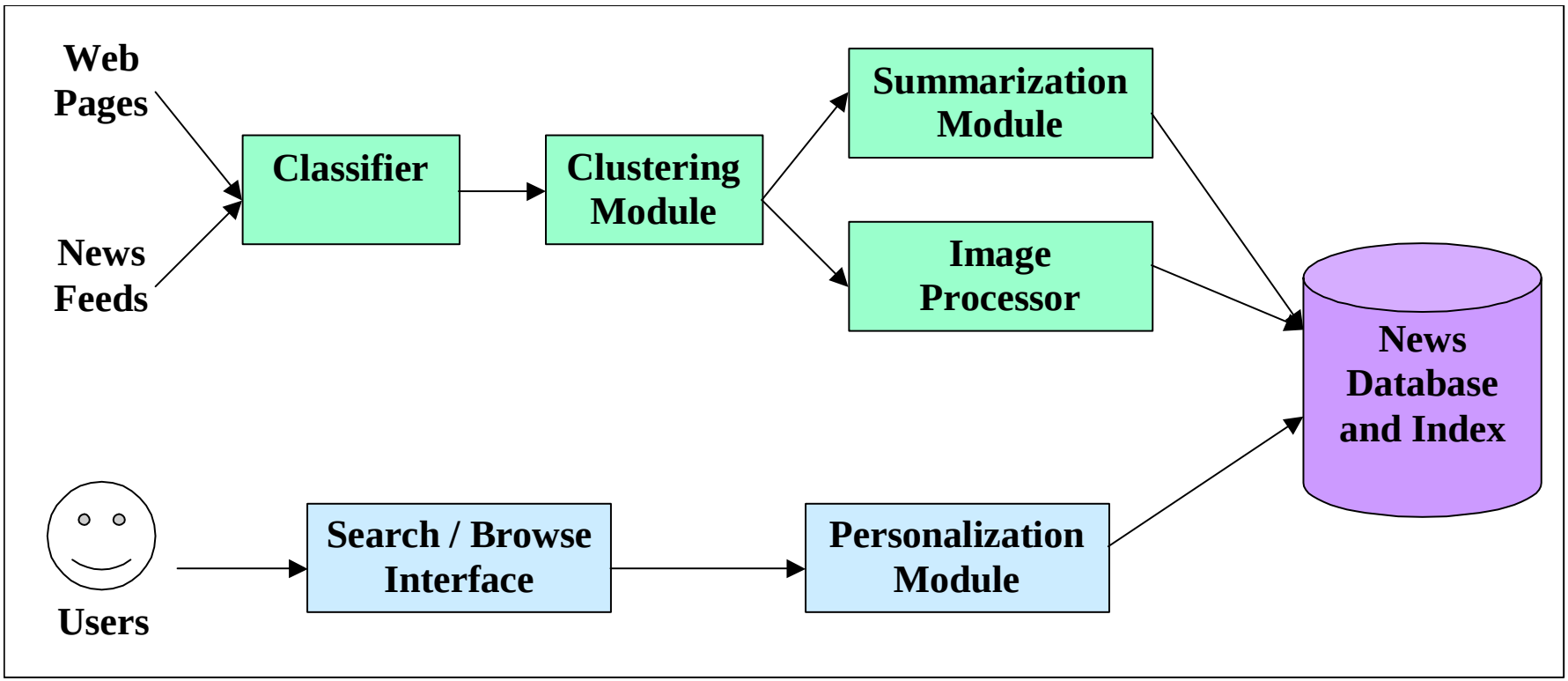
Some slides provided by Thorsten Joachims, Rich Caruana, and Ray Mooney

Outline

- **Architecture of an online news system**
- **Clustering news articles**
- **Personalization / Recommendation**

Examples of Online News Systems

Architecture of an Online News System



- **Based on [Gulli, 2005]**

Classifier

- **Given an article need to classify it into one of the categories**
  - E.g. *Sports*, *Business*, *Science*,…
- **Some articles are already assigned a category**
  - Training examples to continuously update the classifier
- **Can use any classifier**
  - Naïve Bayes, SVM, Decision Trees, etc.
  - Train one classifier for each category

Outline

- Architecture of an online news system
- **Clustering news articles**
- Personalization / Recommendation

Clustering Module

- **Given the articles in a particular category, need to determine which ones are on the same topic**
  - Only show a representative of the set to the user
- **Clustering**
  - Given a dataset and a similarity/distance function
  - Find a partitioning of data such that similar/close points are grouped together
- **Tasks**
  - Define a similarity/distance function
  - Choose a clustering algorithm

Types of Clustering

- **Types of Clustering**
  - Partitioning: *K-means, K-medoids, EM clustering*
  - Hierarchical: *Divisive, Agglomerative*

- **Partitioning Clustering**
  - Hard: *each object is in only one cluster*
  - Soft: *each object has a probability of being in each cluster*

- **Distance/Similarity Space**
  - Vector space: distance between any two points is given
  - Pairwise distance: only distances between some pairs of points are given

Hierarchical Agglomerative Clustering

- **Start with all instances in a separate cluster, then repeatedly merge the two most similar clusters until there is only one cluster with all points**

- **The history of merging forms a binary tree or hierarchy**

- **Merging criteria**
  - Single link: *similarity of two most similar members*
  - Complete link: *similarity of two least similar members*
  - Group average: *average similarity between members*

- **Typical complexity is** $O(n^2)$

Partitioning as Optimization Problem

- **Clustering Criterion**
  - Function that assignes (usually real-valued) value to a clustering

- **Find clustering that maximizes the criterion**
  - Global optimization (often intractable)
  - Greedy Search
  - Approximation algorithms

Centroid-based Clustering

- **Assumes points are in vector space**

- **Clusters are represented via *centroids* – mean points in a cluster. For a cluster c**

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

- **Given k, find a k-partitioning such that the sum of the distances of points to their centroids is minimum**

$$\min_c \sum_c \sum_{\bar{x}\in c} dist(\bar{x}, \bar{\mu}(c))$$

- **NP-hard**

K-means Algorithm

- **Input:**
  - k – number of clusters
  - *dist* – distance function

- **Algorithm**
  - Select k random instances $\{s_1,\ldots,s_k\}$ as initial centroids
  - For each object x_i
    - Assign x_i to the cluster c_j such that $dist(x_i, s_j)$ is minimal
  - For each cluster c_j
    - Update s_j

Properties of K-means Algorithm

- **Always converges to a local optima depending on initial centers**
  - Variety of heuristics for selecting good initial ceners
- **Convergence is fast**
- **Complexity**
  - Computing a distance between two points is O(*m*)
  - Reassigning clusters for n points is O(*knm*)
  - Recomputing centroids is O(*nm*)
  - Assume the two steps are each done for *i* iterations: O(*iknm*)
  - Linear in all relevant factors, more efficient than HAC

Choosing K

- **Typically heuristic / application specific**

- **K-means**
  - The goal is to minimize the objective function
  - Always minimized for $k = n$
  - Start with small k, and gradually increase, stop when the reduction in objective function is small
- **HAC**
  - Use the K-means approach
  - Use similarity threshold for the merging criteria (can be determined empirically or learned from the training data)

Clustering News Articles

- **News articles are represented as N-dimensional vectors of features**
  - Frequencies of occurrence of words in the title and body, phrases, named entities, etc.
  - Features are assigned different weights

- **Similarity measure: weighted cosine similarity**

$$sim(\overline{x}_1, \overline{x}_2) = \frac{\overline{x}_1 \bullet \overline{x}_2}{|\overline{x}_1| * |\overline{x}_2|}$$

- **Already have an existing clustering**
  - Approximately know the value for *k*

Outline

- Architecture of an online news system
- Clustering news articles
- **Personalization / Recommendation**

Personalization

- **Want to recommend the user articles he/she will be most interested in**
- **Two approaches**
  - Collaborative filtering approach
  - Content based approach
- **Machine Learning can allow learning a *user model* or *profile* of a particular user based on interaction history**
- **This model or profile can be used to recommend new articles to the user or to filter out unwanted articles**

Collaborative Filtering

- **Maintain a database of users' reading history**
- **For a given user, find other similar users whose reading histories strongly correlate with the current user**
- **Recommend articles read by these similar users, but not read by the current user**

- **Note:** The framework can easily incorporate ratings. Most existing commercial recommender systems use this approach.

Collaborative Filtering Details

- **Similarity weighting:**
  - Cosine similarity: $sim(\overline{r}_1, \overline{r}_2) = \frac{\overline{r}_1 \bullet \overline{r}_2}{|\overline{r}_1| * |\overline{r}_2|}$
  - Pearson correlation coefficient: $sim(\overline{r}_1, \overline{r}_2) = \frac{\mathrm{cov}(\overline{r}_1, \overline{r}_2)}{\sigma_{\overline{r}_1} * \sigma_{\overline{r}_2}}$
- **Typically, include significance weights based on the number of co-read articles *m*:**

$$adj\_sim(\overline{r}_1, \overline{r}_2) = w(m) * sim(\overline{r}_1, \overline{r}_2)$$

- **"Interestingness" prediction:**

$$pred(a,i) = r_a^{avg} + \frac{\sum_{u=1}^{n} sim(\overline{r}_a, \overline{r}_u) * (r_{u,i} - r_u^{avg})}{\sum_{u=1}^{n} sim(\overline{r}_a, \overline{r}_u)}$$

Problems with Collaborative Filtering

- **Cold Start:** There needs to be enough other users already in the system to find a match.
- **Sparsity:** If there are many articles to be recommended, even if there are many users, the user/read matrix is sparse, and it is hard to find users that have read the same articles.
- **First Rater:** Cannot recommend an article that has not been previously read
  - New articles
  - Esoteric articles
- **Popularity Bias:** Cannot recommend articles to someone with unique tastes
  - Tends to recommend popular articles

Content-Based Approach

- **Recommendations are based on the *content* of items rather than on other users' opinions**
- **Build a profile of the user preferences using content of the articles the user has previously read**
- **Recommend articles most similar to the user profile**
  - Can use user feedback to learn an optimal similarity function
- **Note:** works for news articles, but typically is not applicable to non-textual items (movies, photo-cameras, etc.)

Content-Based Approach

- **Pros:**
  - No need for data on other users (no cold-start, sparsity, or first-rater problems)
  - Able to recommend to users with unique tastes
  - Able to recommend new and unpopular items
  - Can provide explanations of recommended items by listing content-features that caused an item to be recommended
- **Cons:**
  - Users' tastes must be represented as a learnable function of these content features
  - Unable to exploit quality judgments of other users
  - Requires content that can be encoded as meaningful features

Challenges in applying CF to News Articles

- **Scale**
  - Millions of users
  - Millions of articles
- **Frequent updates**
  - News stories change every few minutes
  - The most recent stories are the most important
- **Noisy data**
  - Interpreting clicks as absolute relevance judegements is dangerous

- **See [Das et. al, 2007] for an example of how these issues can be addressed in a real system**

Summary

- **Online news system is an example of a real system where ML algorithms are used on several stages**
  - Classification
  - Clustering
  - Recommending
  - Ranking (potentially)
- **Two types of clustering algorithms are hierarchical (HAC) and partitioning (K-means); either can be used for clustering new articles**
- **Two approaches to recommending are collaborative filtering and content-based; either can be used for recommending news articles**
- **Main challenges in applying ML algorithms in this setting are scalability, frequent updates, and noisy feedback data**

References

- **[Gulli, 2005]** Gulli, A. *The Anatomy of a News Search Engine*. WWW-2005.
- **[Das et. al, 2007]** Das, A., Datar, M., Garg, A., Rajaram, S. *Google News Personalization: Scalable Online Collaborative Filtering*. WWW-2007.

Part 6

Machine Learning for Finding Compound Documents

Pavel Dmitriev, Mikhail Bilenko

Compound Documents

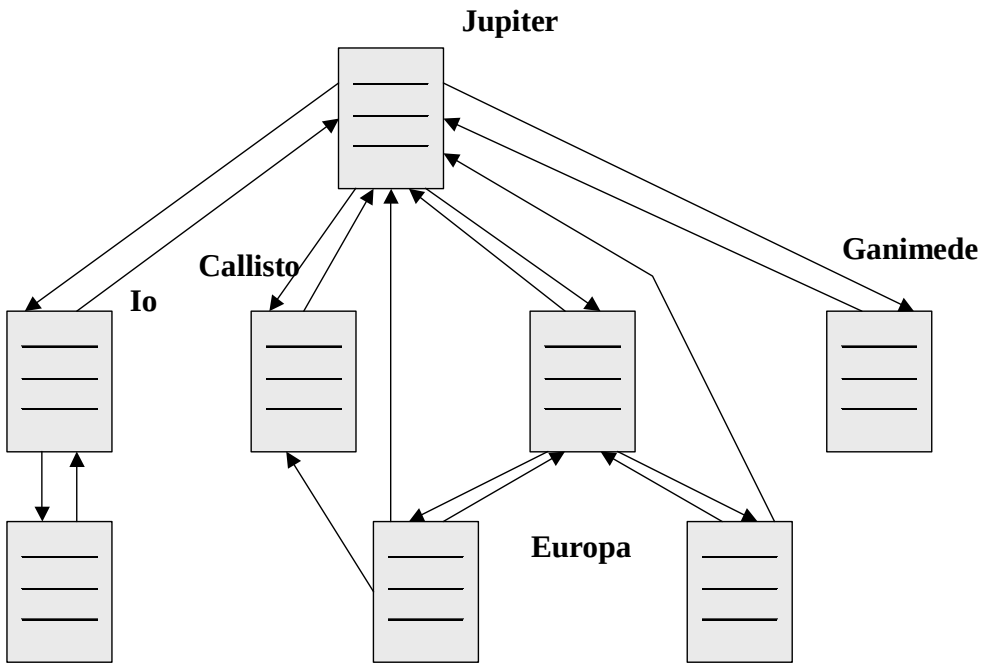
- **A *Compound Document* (*cDoc*) is a group of web pages that in aggregate correspond to a coherent informaiton entity**
  - A news article on the web consisting of several physical HTML pages
  - An entry in an online encyclopedia
  - A set of web pages describing product's specifications, price, reviews, etc.

- **Want to design an algorithm for automatically identifying cDocs**

Finding cDocs as a weighted graph clustering problem

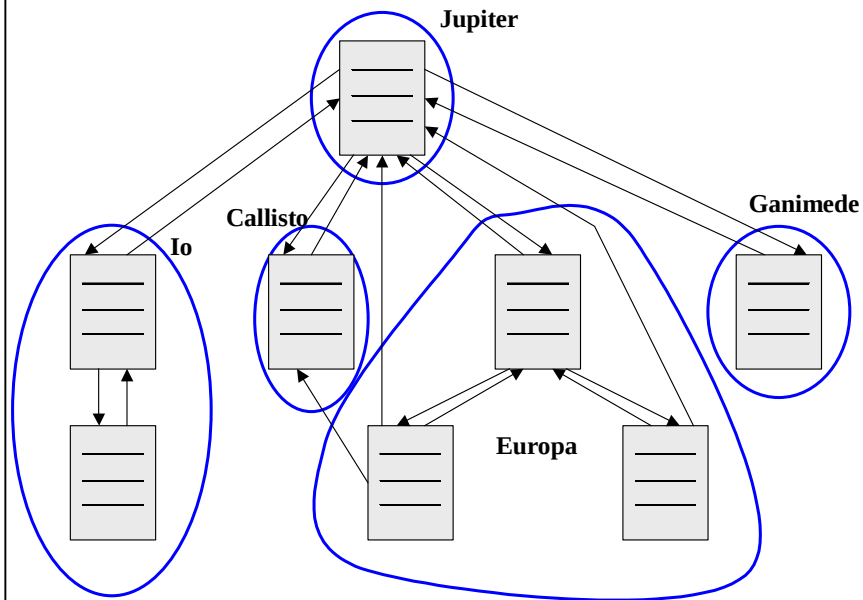
- **Represent a web site as a directed graph**
  - Nodes are web pages
  - Edges are hyperlinks
- **Assign weights to the edges**
  - The weight on a hyperlink between two web pages is higher if they are more likely to be in the same cDoc
- **The set of cDocs is a clustering of this graph**
  - Every cluster of more than 1 node is a cDoc

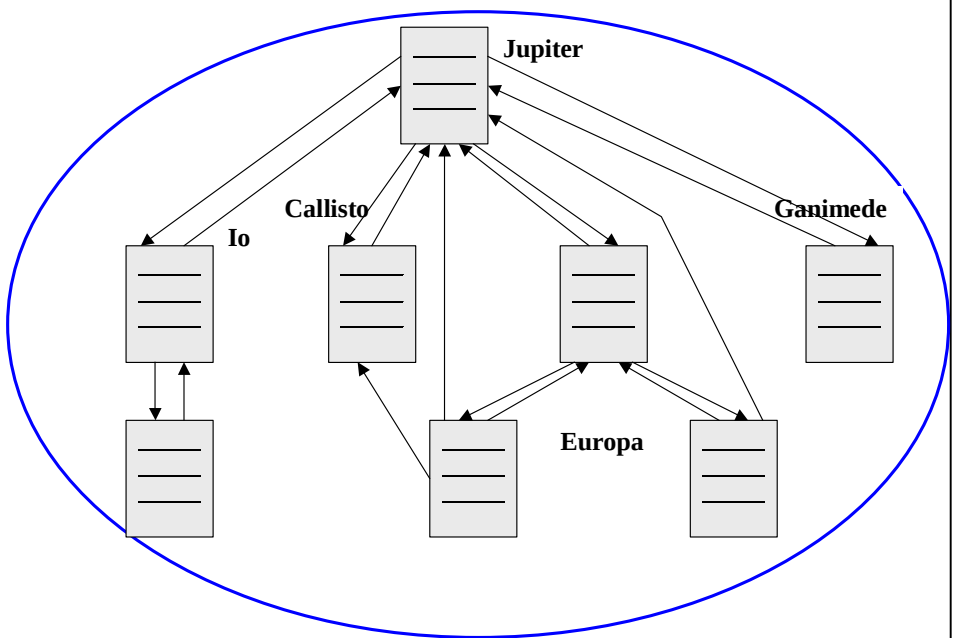
Observation



What are the cDocs?

Observation





Definition of a cDocs is subjective

(depends on the application)

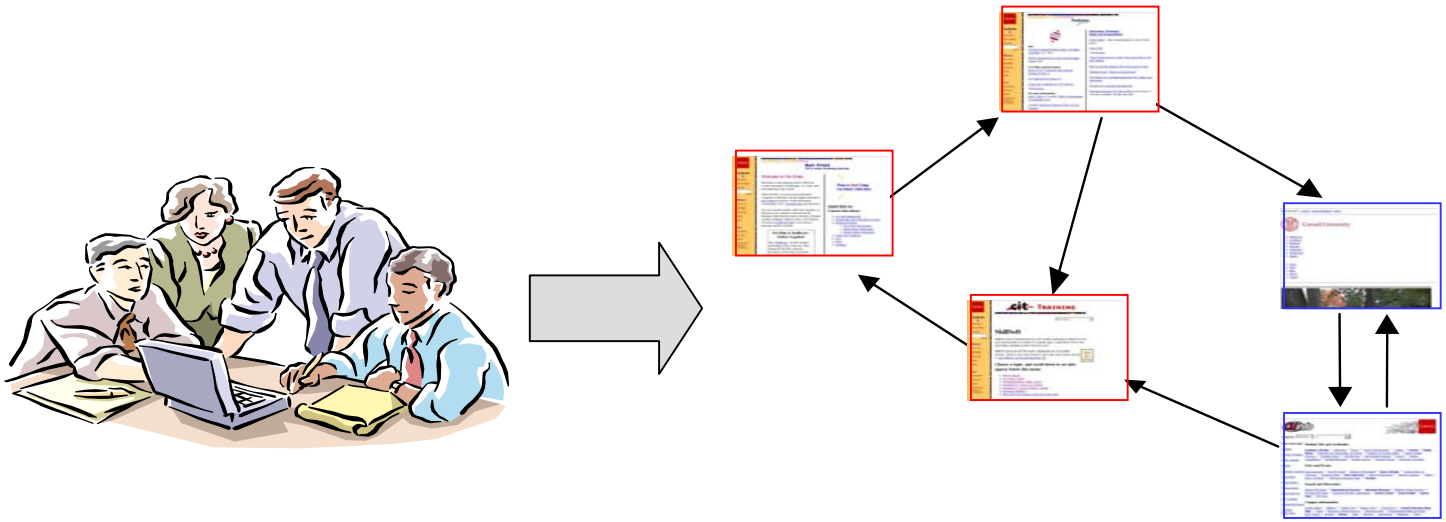
Machine Learning Framework

- **Goal: given a web site and a few examples of cDocs on this web site, identify other cDocs on the same web site**

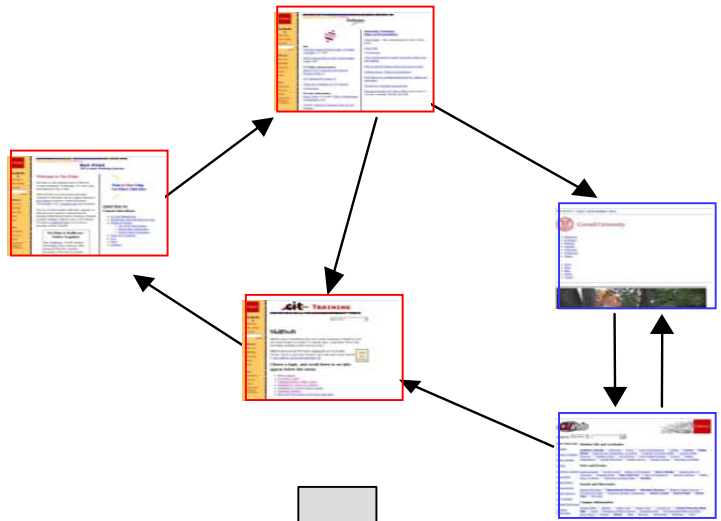
- **Step 1**
  - Use user examples to learn a “description” of a cDoc

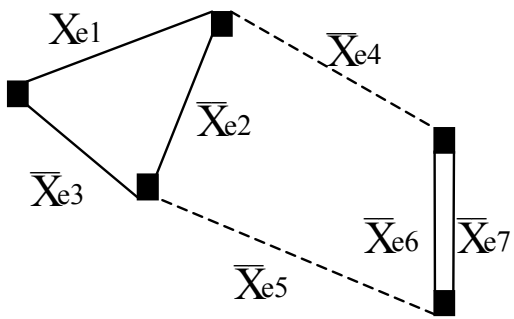
- **Step 2**
  - Automatically identify new cDocs beased on the learned “description”

Step 1: Learning

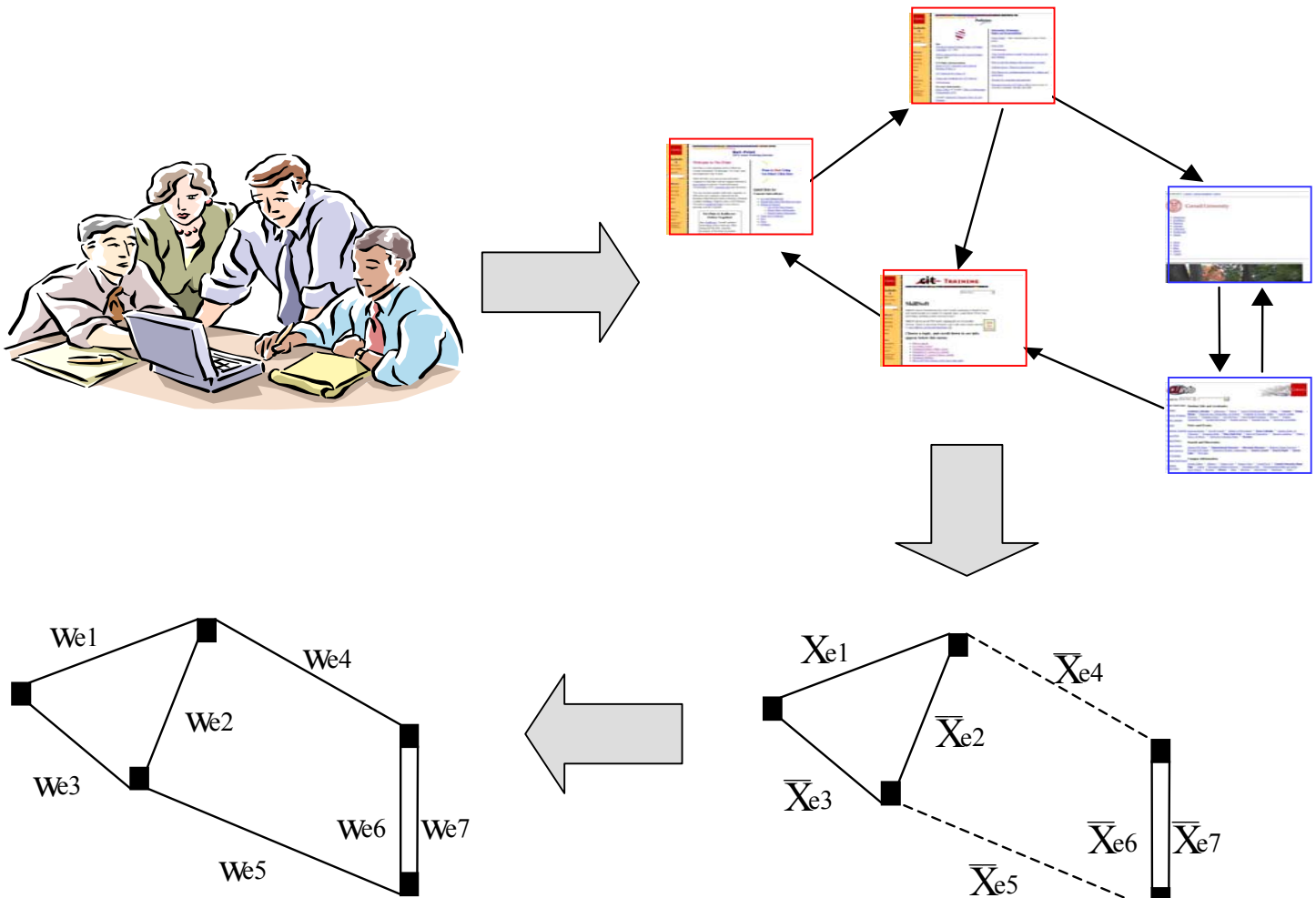


Step 1: Learning

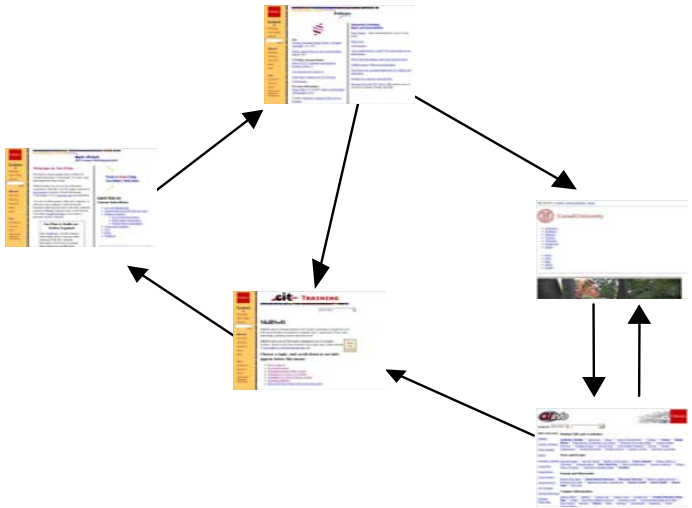




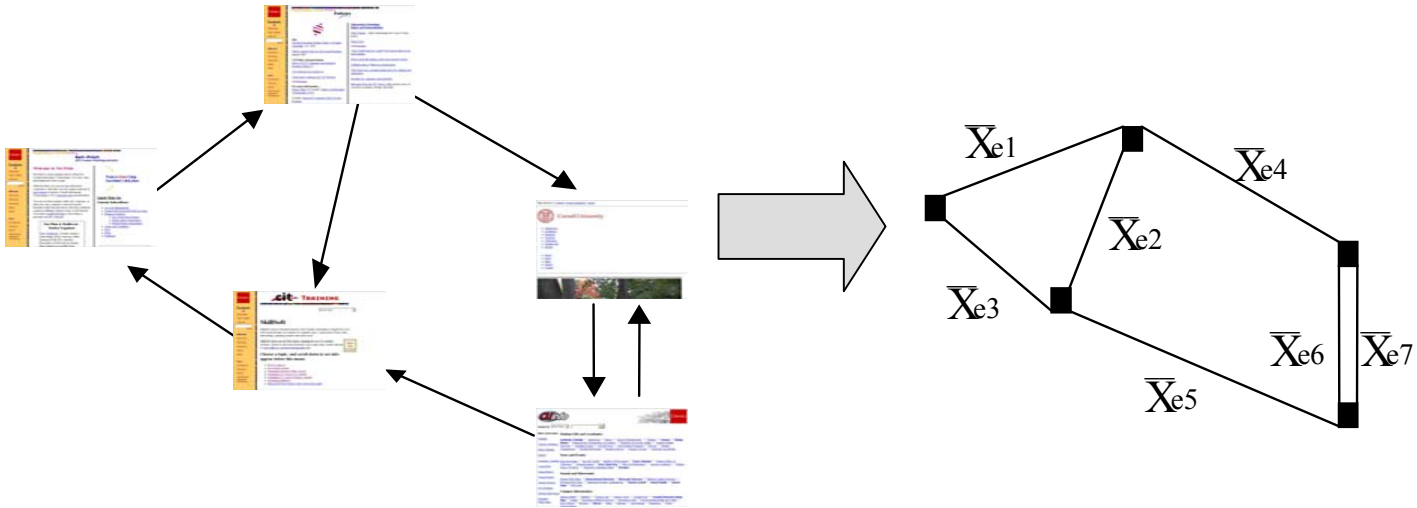
Step 1: Learning



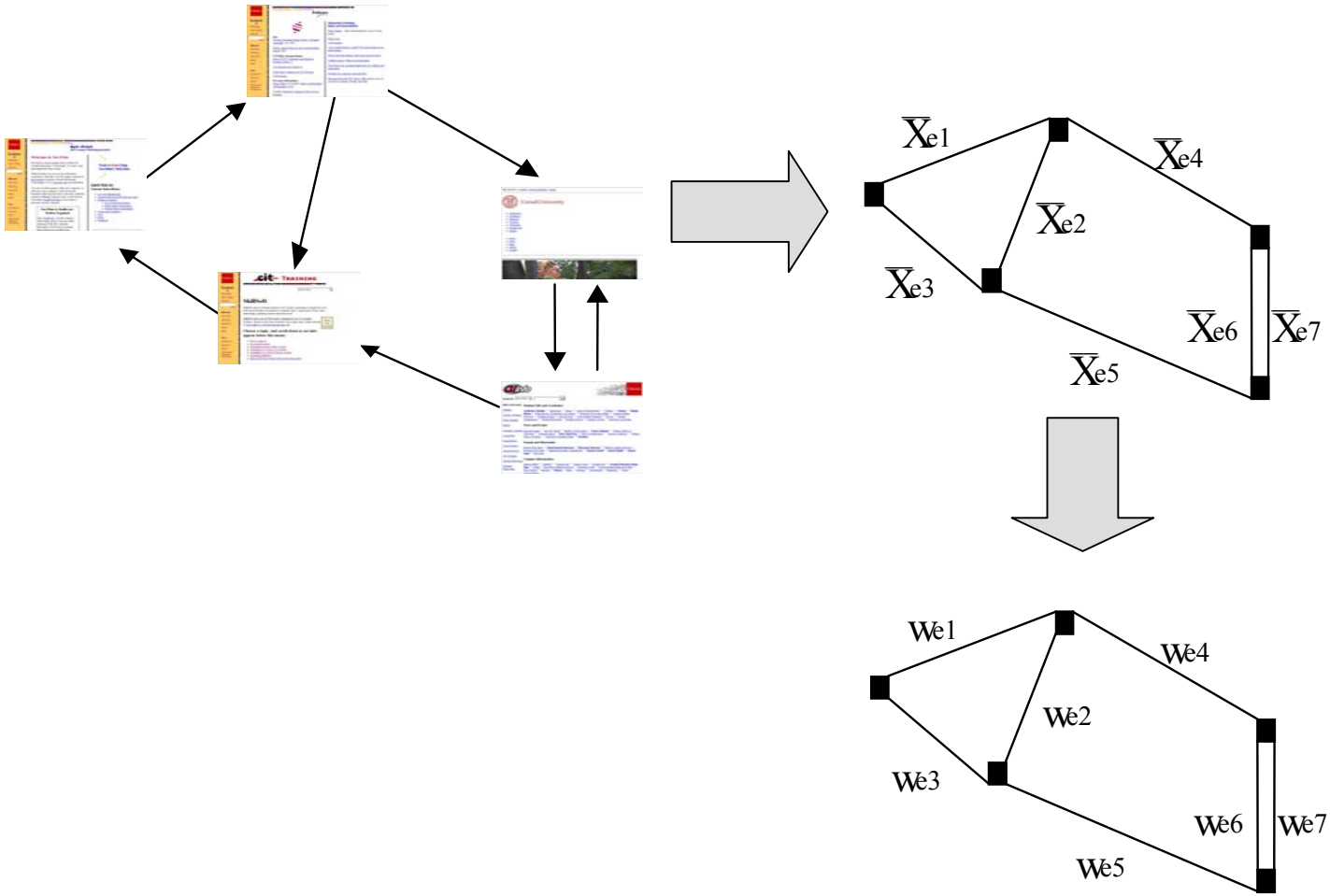
Step 2: Inference



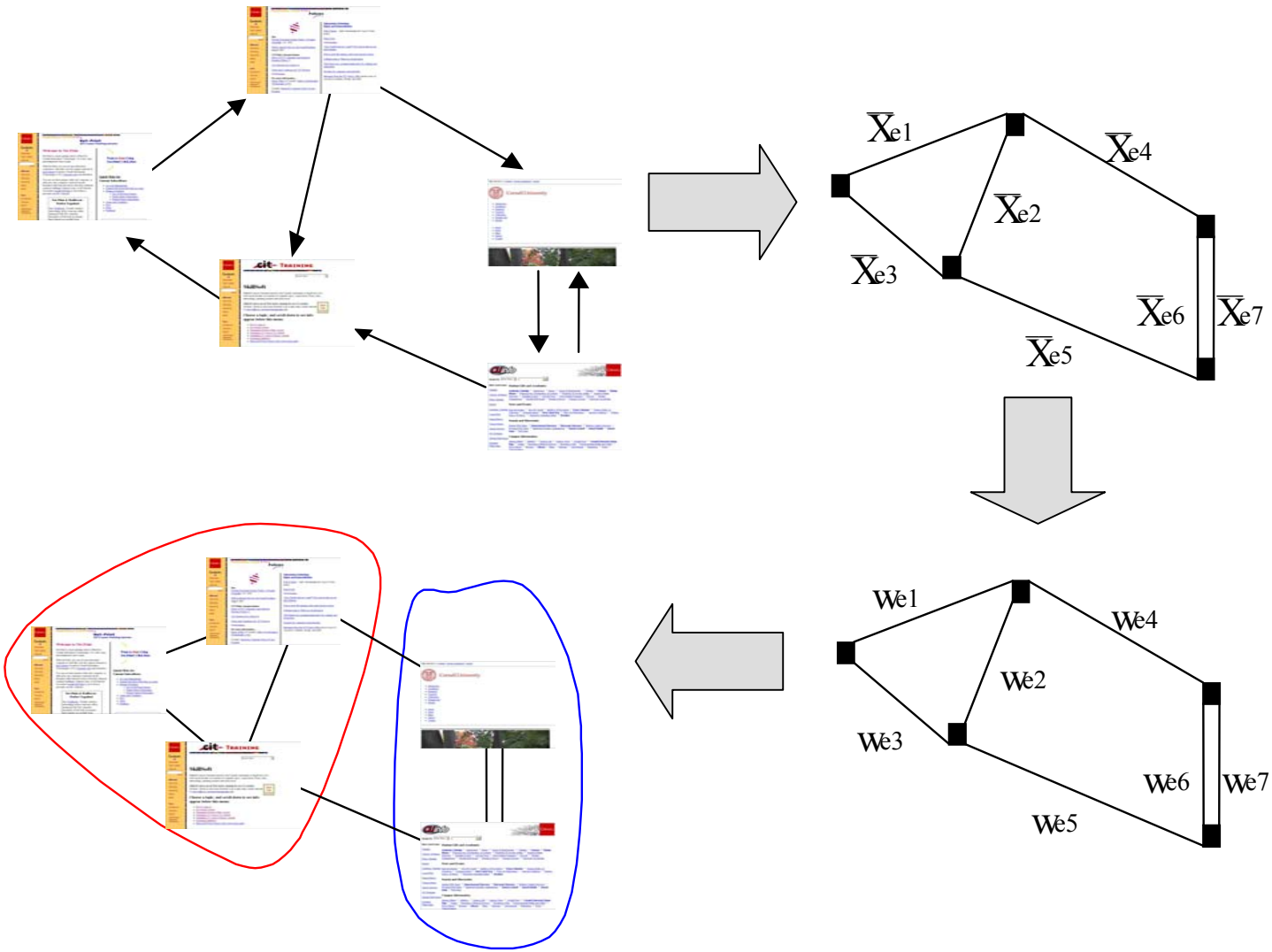
Step 2: Inference



Step 2: Inference



Step 2: Inference



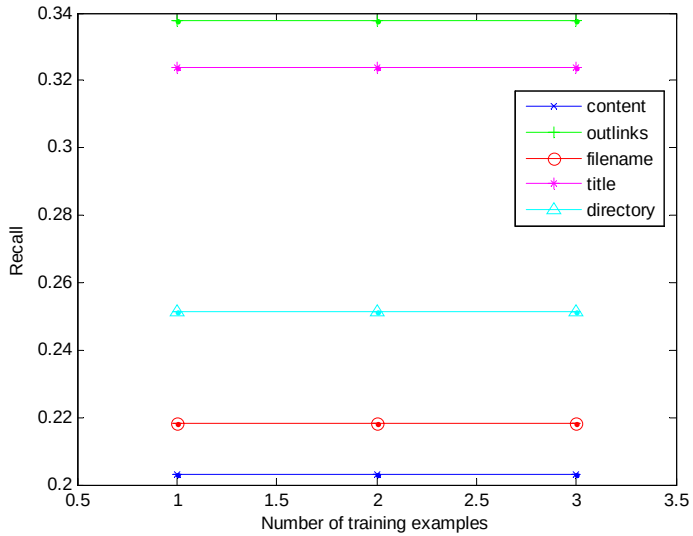
Combination of supervised and unsupervised learning

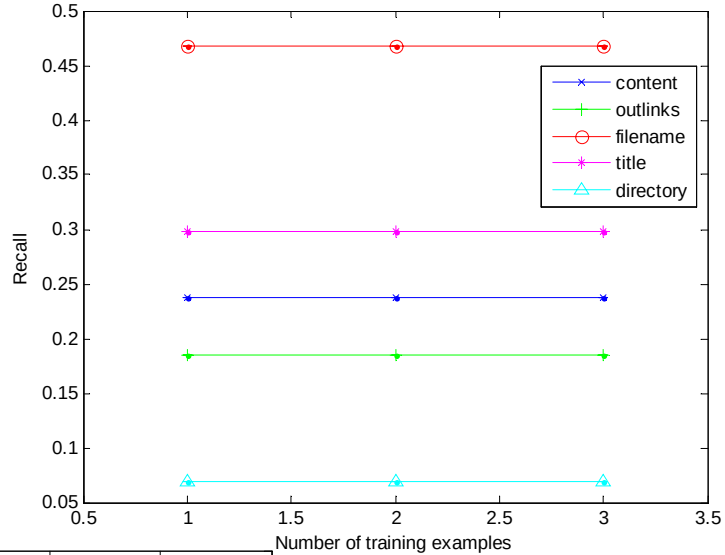
- **Do unsupervised learning (clustering) in a supervised way (learn the similarity function)**
- **Learning algorithm**
  - Any density estimator (experiments used logistic regression)
- **Clustering algorithm**
  - A variant of HAC
  - Number of clusters is determined automatically using a threshold learned during the training stage

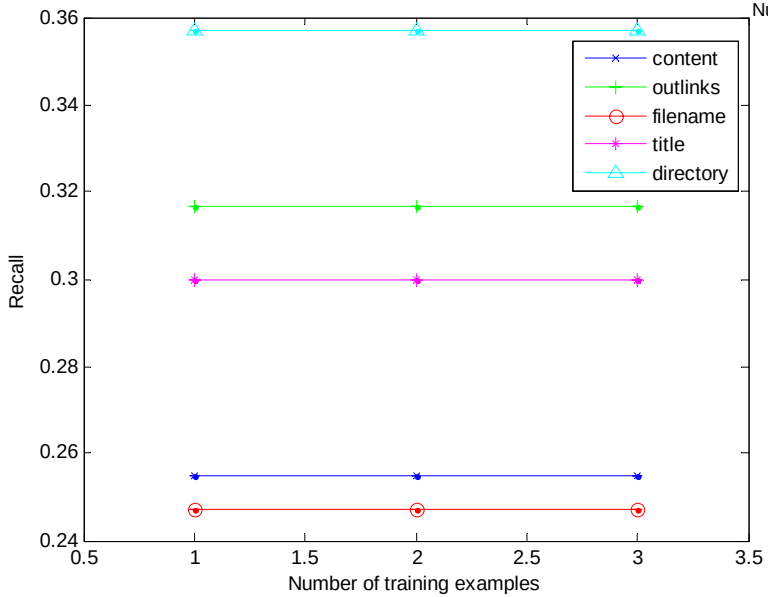
Experimental Results

- **Dataset: 60 real websites**
  - 20 educational, 20 news, 20 commercial
  - 169 pages, 1400 hyperlinks, 19 cDocs on average
- **Training Data**
  - 1, 2, 3 training examples picked at random
- **Evaluation criterion**
  - Recall (percentage of cDocs identified exactly right)
- **Compare to approaches based on a single feature, directory sructure, and pattern matching**

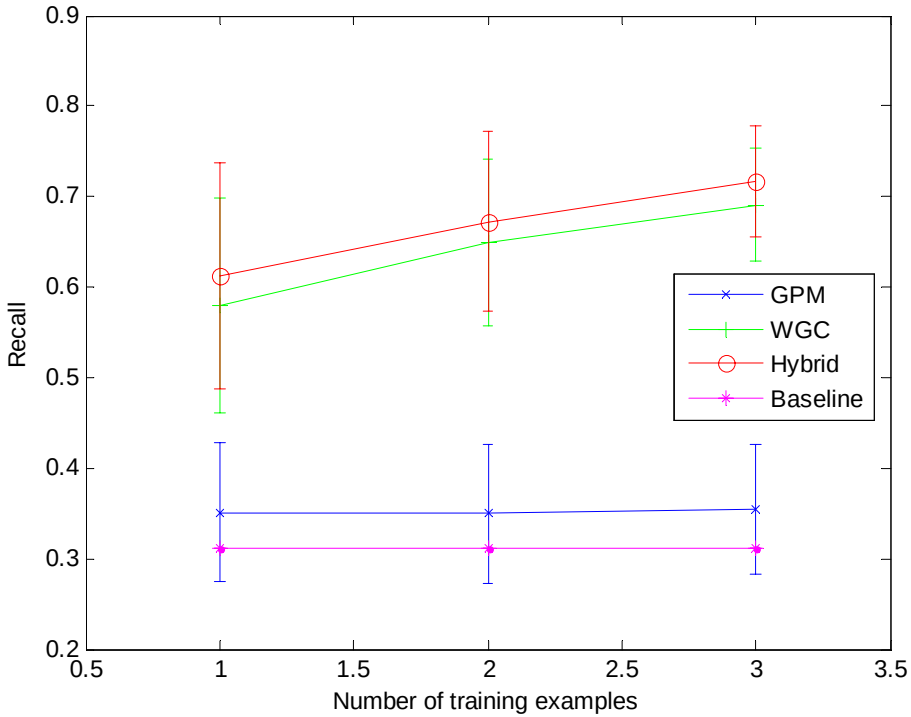
Experimental Results: Single Feature







Experimental Results: WGC



Summary

- **Used unsupervised learning (clustering) in a supervised way to solve the problem of finding boundaries of cDocs**
  - Learned the similarity function

- **Other approaches to learning how to cluster**
  - Use training examples as constraints for the clustering algorithm
  - Train a classifier which for every pair of points will predict whether they should be in the same cluster or in different clusters. Somehow resolve the conflicts.
  - Directly optimize the clustering objective function

References

- **[Dmitriev et al, 2005]** Dmitriev, P., Lagoze, C., Suchkov, B. *"As We May Perceive: Inferring Logical Documents from Hypertext"*. Hypertext-2005.

Conclusion

Pavel Dmitriev, Mikhail Bilenko

Summary

- **Introduction to ML**
  - ML is conserned with developing algorithms that can improve their performance with experience
  - There are standard (very often encountered in practice) ML problems: classification, regression, and clustering
  - By the type of information used, ML algorithms can be classified in unsupervised, supervised, and semi-supervised
  - Two most common approaches to ML are Bayesian (or generative) and Optimization (or discriminative) approaches

Summary

- **ML against SPAM**
  - E-mail SPAM detection can be represented as a classification problem
  - Naïve Bayes is a simple generative method that can be used to solve it
  - Advantages of NB are that it is simple, fast to train and use, and easy to update
  - The main disadvantage is that the feature independence given class labels assumption it makes typically does not hold

Summary

- **ML for Information Extraction**
  - IE is the task of extracting values for specific fields from text
  - Can be viewed as a classification problem. In order to solve it effectively, need to account for dependencies among words
  - Hidden Markov Model is a generative model which allows specifying dependencies among adjacent words
  - Expectation-Maximization and Dynamic Programming can be used to perform learning and inference, still an order of magnitude slower than Naïve Bayes
  - There are more expressive, slower, and often more effective models that can also be used (such as CRFs)

Summary

- **ML in Web Search**
  - Can use clickthrough data to learn a ranking function for a search engine
  - Clickthrough data is noisy and difficult to interpret
  - A user study showed that interpreting clickthrough data as absolute feedback is difficult, but interpreting it as relative feedback seems reliable
  - SVM is a discriminative linear classifier; it can still be used when the training data is not linearly separable; it can use kernels to learn non-linear functions
  - SVM can be used to learn a ranking function from relative feedback obtained from clickthrough data

Summary

- **ML in the News**
  - Example of a real world system where ML algorithms are used on several stages: classification, clustering, recommending, ranking
  - Two types of clustering algorithms are hierarchical (HAC) and partitioning (K-means); either can be used for clustering new articles
  - Two approaches to recommending are collaborative filtering and content-based; either can be used for recommending news articles
  - Main challenges in applying the above ML algorithms in the setting of an online news system are scalability, frequent updates, and noisy feedback data

Summary

- **ML for finding Compound Documents**
  - Given a web site, want to identify sets of pages corresponding to semantically coherent entites (cDocs)
  - The definition of a cDoc depends on the user/application, so want to use ML
  - Can formulate finding cDocs as a weighted graph clustering problem
  - Can use ML to learn the distance function (edge weights), resulting in a semi-supervised clustering algorithm

Note

- **The goal of this tutorial was really to "introduce", rather than to "explain"…**

- **"Introduce" you to**
  - The general area of ML
  - Some Web-related problems that can be addressed using ML methods
  - How to adapt classical ML algorithms to solve these problems
  - What to look out for when doing that

- **Many (very interesting) details were left out**

More Information

- **Books**
  - Mitchell, M.T., *Machine Learning*, McGraw Hill, 1997, ISBN 0070428077
  - Hastie, T., Tibshirani, R., *The Elements of Statistical Learning*, …
- **Conferences**
  - ICML
  - NIPS
  - KDD
- **Web**
  - In <your_favourite_search_engine> type

"<ml_algorithm_name> tutorial"

RuSSIR

Russian Summer School in Information Retrieval 2007

Thank you!

Pavel Dmitriev, Mikhail Bilenko

Microsoft® Research

Y!™