IR 11:
Text Classification;
The Naïve Bayes algorithm

Relevance feedback

- In relevance feedback, the user marks a number of documents as relevant/nonrelevant.
- We then try to use this information to return better search results.
- Suppose we just tried to learn a filter for nonrelevant documents
- This is an instance of a text classification problem:
  - Two “classes”: **relevant**, **nonrelevant**
  - For each document, decide whether it is relevant or nonrelevant
- The notion of classification is very general and has many applications within and beyond information retrieval.

Standing queries

- The path from information retrieval to text classification:
  - You have an information need, say:
    - Unrest in the Niger delta region
  - You want to rerun an appropriate query periodically to find new news items on this topic
  - You will be sent new documents that are found
    - I.e., it's classification not ranking
- Such queries are called ***standing queries***
  - Long used by "information professionals"
  - A modern mass instantiation is **Google Alerts**

Spam filtering: Another text classification task

From: "" <takworlld@hotmail.com>
Subject: real estate is the only way... gem oalvgkay

Anyone can buy real estate with no money down

Stop paying rent TODAY !

There is no need to spend hundreds or even thousands for similar courses

I am 22 years old and I have already purchased 6 properties using the methods outlined in this truly INCREDIBLE ebook.

Change your life NOW !

===

Click Below to order:
http://www.wholesaledaily.com/sales/nmd.htm

Text classification: Naïve Bayes Text Classification

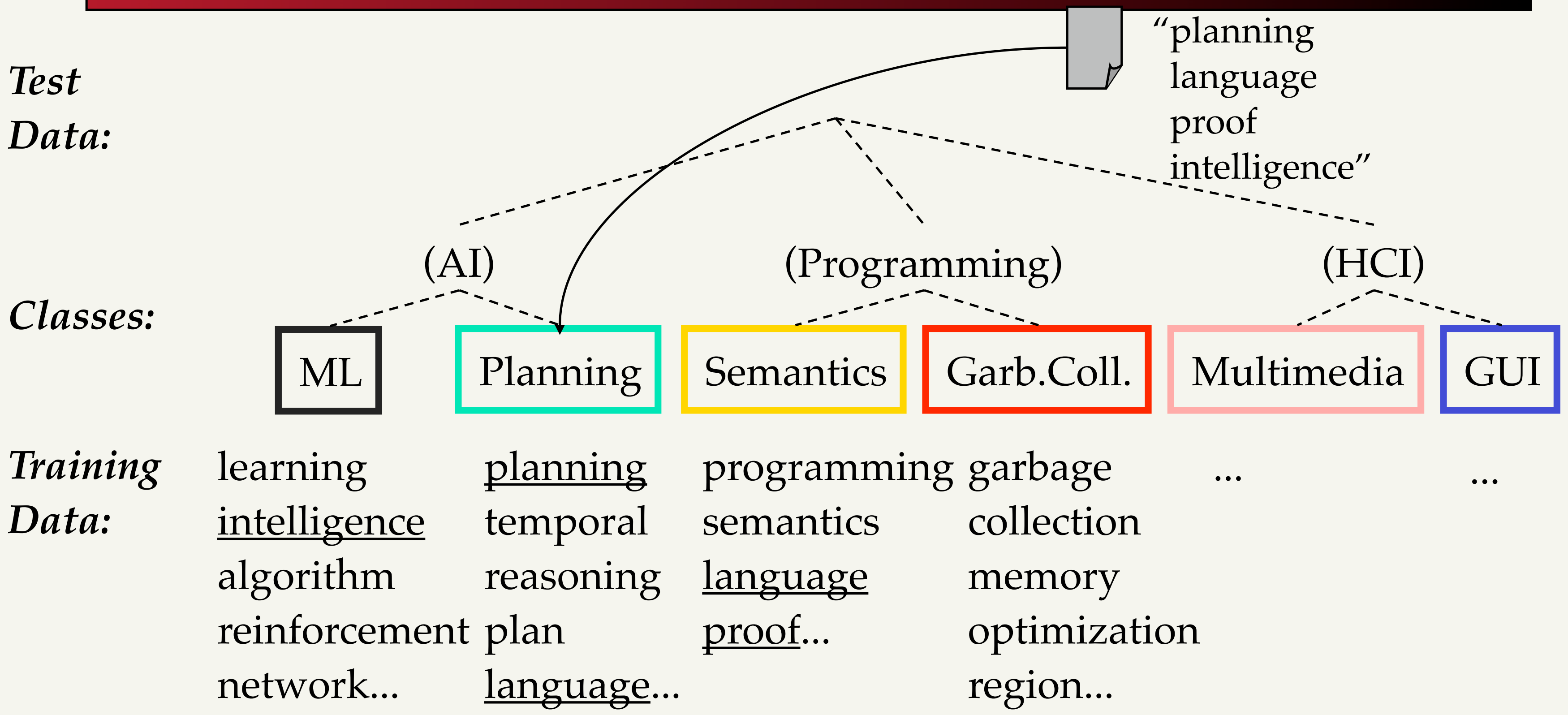
- Introduction to Text Classification
  - Also widely known as “text categorization”. Same thing.
- Probabilistic Language Models
- Naïve Bayes text classification

Categorization/Classification

- Given:
  - A description of an instance, $x \in X$, where X is the *instance language* or *instance space*.
    - Issue: how to represent text documents.
  - A fixed set of classes:

    $C = \{c_1, c_2, \ldots, c_J\}$
- Determine:
  - The category of x: $c(x) \in C$, where $c(x)$ is a *classification function* whose domain is X and whose range is C.
    - We want to know how to build classification functions ("classifiers").

Document Classification


Test Data:
"planning language proof intelligence"
(AI)
(Programming)
(HCI)
Classes:
ML
Planning
Semantics
Garb.Coll.
Multimedia
GUI
Training Data:
learning intelligence algorithm reinforcement network...
planning temporal reasoning plan language...
programming semantics language proof...
garbage collection memory optimization region...
...
...

(Note: in real life there is often a hierarchy, not present in the above problem statement; and also, you get papers on ML approaches to Garb. Coll.)

More Text Classification Examples:

Many search engine functionalities use classification

Assign labels to each document or web-page:

- Labels are most often topics such as Yahoo-categories
  e.g., "finance," "sports," "news>world>asia>business"
- Labels may be genres
  e.g., "editorials" "movie-reviews" "news“
- Labels may be opinion on a person/product
  e.g., “like”, “hate”, “neutral”
- Labels may be domain-specific
  e.g., "interesting-to-me" : "not-interesting-to-me”
  e.g., “contains adult language” : “doesn’t”
  e.g., language identification: English, French, Chinese, …
  e.g., search vertical: about Linux versus not
  e.g., “link spam” : “not link spam”

Classification Methods (1)

- Manual classification
  - Used by Yahoo! (originally; now present but downplayed), Looksmart, about.com, ODP, PubMed
  - Very accurate when job is done by experts
  - Consistent when the problem size and team is small
  - Difficult and expensive to scale
    - Means we need automatic classification methods for big problems

Classification Methods (2)

- Automatic document classification
  - Hand-coded rule-based systems
    - One technique used by CS dept's spam filter, Reuters, CIA, etc.
    - Companies (Verity) provide "IDE" for writing such rules
    - E.g., assign category if document contains a given boolean combination of words
    - Standing queries: Commercial systems have complex query languages (everything in IR query languages + accumulators)
    - Accuracy is often very high if a rule has been carefully refined over time by a subject expert
    - Building and maintaining these rules is expensive

Classification Methods (3)

- Supervised learning of a document-label assignment function
  - Many systems partly rely on machine learning (Autonomy, MSN, Verity, Enkata, Yahoo!, …)
    - k-Nearest Neighbors (simple, powerful)
    - Naive Bayes (simple, common method)
    - Support-vector machines (new, more powerful)
    - … plus many other methods
    - No free lunch: requires hand-classified training data
    - But data can be built up (and refined) by amateurs
- Note that many commercial systems use a mixture of methods

Probabilistic relevance feedback

Recall this idea:

- Rather than reweighting in a vector space…
- If user has told us some relevant and some irrelevant documents, then we can proceed to build a probabilistic classifier, such as the Naive Bayes model we will look at today:
  - $P(t_k|R) = |\mathbf{D}_{rk}| / |\mathbf{D}_r|$
  - $P(t_k|NR) = |\mathbf{D}_{nrk}| / |\mathbf{D}_{nr}|$
    - t_k is a term; $\mathbf{D}_r$ is the set of known relevant documents; $\mathbf{D}_{rk}$ is the subset that contain t_k; $\mathbf{D}_{nr}$ is the set of known irrelevant documents; $\mathbf{D}_{nrk}$ is the subset that contain t_k.

9.1.2

Recall a few probability basics

- For events *a* and *b:*
- Bayes' Rule

$$p(a,b) = p(a \cap b) = p(a \mid b)\,p(b) = p(b \mid a)\,p(a)$$

$$p(\bar{a} \mid b)\,p(b) = p(b \mid \bar{a})\,p(\bar{a})$$

$$p(a \mid b) = \frac{p(b \mid a)\,p(a)}{p(b)} = \frac{p(b \mid a)\,p(a)}{\sum_{x=a,\bar{a}} p(b \mid x)\,p(x)}$$

Prior

Posterior

- Odds: $$O(a) = \frac{p(a)}{p(\bar{a})} = \frac{p(a)}{1 - p(a)}$$

Bayesian Methods

- Our focus this lecture
- Learning and classification methods based on probability theory.
- Bayes theorem plays a critical role in probabilistic learning and classification.
- Build a *generative model* that approximates how data is produced
- Uses *prior* probability of each category given no information about an item.
- Categorization produces a *posterior* probability distribution over the possible categories given a description of an item.

Bayes' Rule

$$P(C,D) = P(C \mid D)P(D) = P(D \mid C)P(C)$$

$$\boxed{P(C \mid D) = \frac{P(D \mid C)P(C)}{P(D)}}$$

Naive Bayes Classifiers

Task: Classify a new instance *D* based on a tuple of attribute values $D = \langle x_1, x_2, \ldots, x_n \rangle$ into one of the classes $c_j \in C$

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

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

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

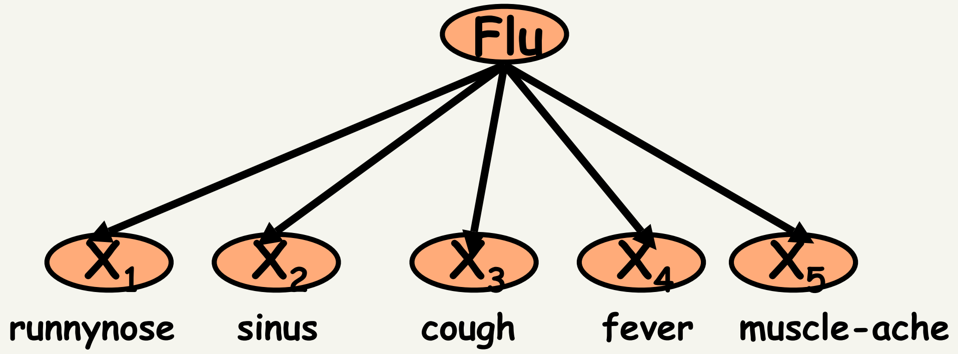
Naïve Bayes Classifier: Naïve Bayes Assumption

- $P(c_j)$
  - Can be estimated from the frequency of classes in the training examples.
- $P(x_1,x_2,\ldots,x_n|c_j)$
  - $O(|X|^n \bullet |C|)$ parameters
  - Could only be estimated if a very, very large number of training examples was available.

Naïve Bayes Conditional Independence Assumption:

- Assume that the probability of observing the conjunction of attributes is equal to the product of the individual probabilities $P(x_i|c_j)$.

The Naïve Bayes Classifier



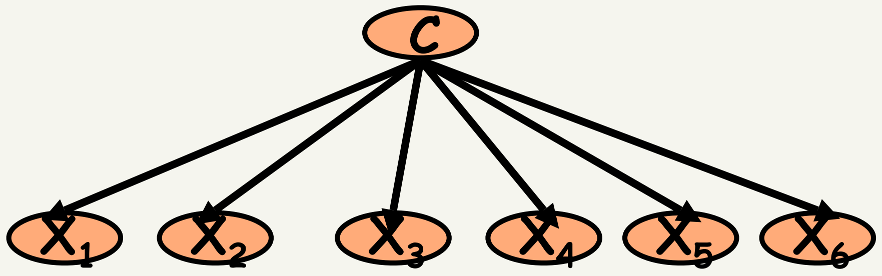
- **Conditional Independence Assumption:** features detect term presence and are independent of each other given the class:

$$P(X_1,\ldots,X_5 \mid C) = P(X_1 \mid C) \bullet P(X_2 \mid C) \bullet \cdots \bullet P(X_5 \mid C)$$

- This model is appropriate for binary variables
  - Multivariate Bernoulli model

13.3

Learning the Model



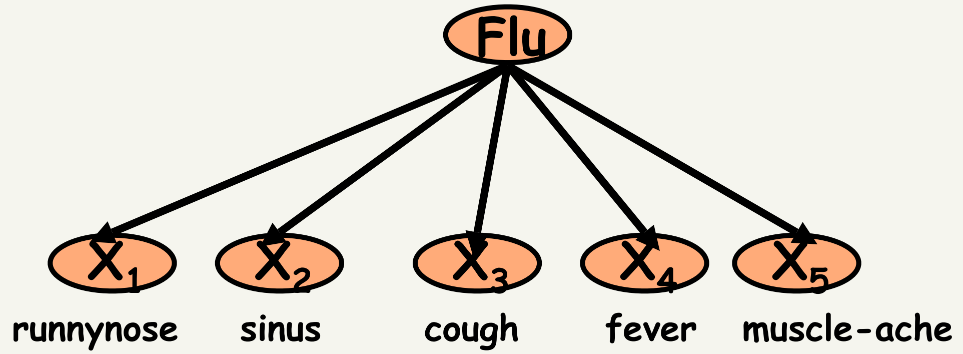
- First attempt: maximum likelihood estimates
  - simply use the frequencies in the data

$$\hat{P}(c_j) = \frac{N(C = c_j)}{N}$$

$$\hat{P}(x_i \mid c_j) = \frac{N(X_i = x_i, C = c_j)}{N(C = c_j)}$$

13.3

Problem with Max Likelihood



$$P(X_1,\ldots,X_5 \mid C) = P(X_1 \mid C) \bullet P(X_2 \mid C) \bullet \cdots \bullet P(X_5 \mid C)$$

- What if we have seen no training cases where patient had no flu and muscle aches?

$$\hat{P}(X_5 = t \mid C = nf) = \frac{N(X_5 = t, C = nf)}{N(C = nf)} = 0$$

- Zero probabilities cannot be conditioned away, no matter the other evidence!

$$\ell = \arg\max_c \hat{P}(c) \prod_i \hat{P}(x_i \mid c)$$

Smoothing to Avoid Overfitting

$$\hat{P}(x_i \mid c_j) = \frac{N(X_i = x_i, C = c_j) + 1}{N(C = c_j) + k}$$

# of values of X_i

- Somewhat more subtle version

overall fraction in data where $X_i = x_{i,k}$

$$\hat{P}(x_{i,k} \mid c_j) = \frac{N(X_i = x_{i,k}, C = c_j) + m p_{i,k}}{N(C = c_j) + m}$$

extent of "smoothing"

Stochastic Language Models

- Models *probability* of generating strings (each word in turn) in the language (commonly all strings over ∑). E.g., unigram model

Model M

0.2	the
0.1	a
0.01	man
0.01	woman
0.03	said
0.02	likes
…	

the	man	likes	the	woman
0.2	0.01	0.02	0.2	0.01

multiply

$P(s \mid M) = 0.00000008$

13.2.1

Stochastic Language Models

- Model *probability* of generating any string

Model M1	
0.2	the
0.01	class
0.0001	sayst
0.0001	pleaseth
0.0001	yon
0.0005	maiden
0.01	woman

Model M2	
0.2	the
0.0001	class
0.03	sayst
0.02	pleaseth
0.1	yon
0.01	maiden
0.0001	woman

the	class	pleaseth	yon	maiden
0.2	0.01	0.0001	0.0001	0.0005
0.2	0.0001	0.02	0.1	0.01

$P(s|M2) > P(s|M1)$

Unigram and higher-order models

P (🔴 🟡 🔴 🔵)

- **= P (🔴) P (🟡 | 🔴) P (🔴 | 🔴 🟡) P (🔵 | 🔴 🟡 🔴)**
- Unigram Language Models

  P (🔴) P (🟡) P (🔴) P (🔵)

Easy.
Effective!

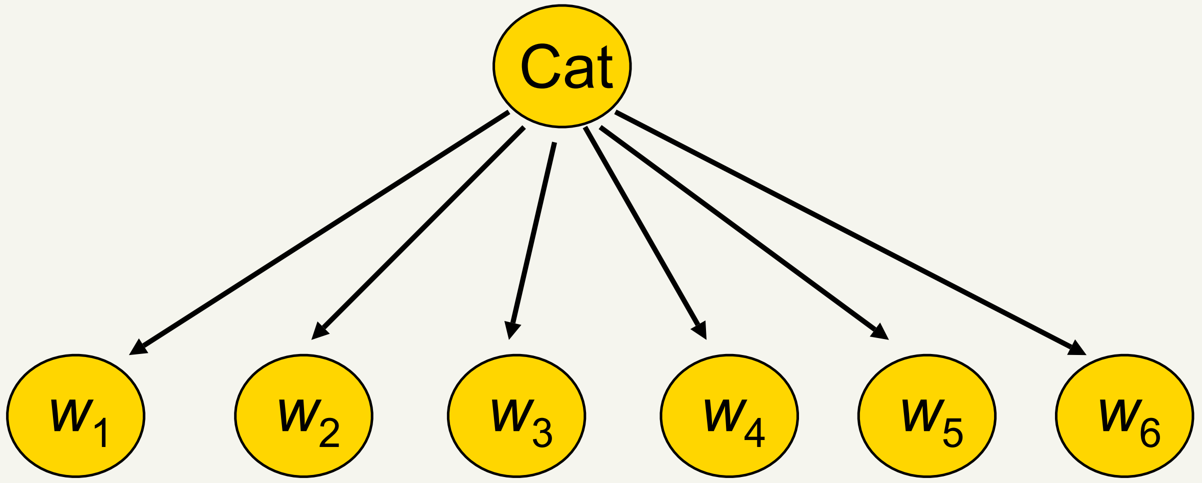
- Bigram (generally, *n*-gram) Language Models

  P (🔴) P (🟡 | 🔴) P (🔴 | 🟡) P (🔵 | 🔴)

- Other Language Models
  - Grammar-based models (PCFGs), etc.
    - Probably not the first thing to try in IR

13.2.1

Naïve Bayes via a class conditional language model = multinomial NB



- Effectively, the probability of each class is done as a class-specific unigram language model

Using Multinomial Naive Bayes Classifiers to Classify Text: Basic method

- Attributes are text positions, values are words.

$$
\begin{aligned}
c_{NB} &= \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_i P(x_i \mid c_j) \\
&= \underset{c_j \in C}{\operatorname{argmax}} P(c_j) P(x_1 = \text{"our"} \mid c_j) \cdots P(x_n = \text{"text"} \mid c_j)
\end{aligned}
$$

- Still too many possibilities
- Assume that classification is *independent* of the positions of the words
  - Use same parameters for each position
  - Result is bag of words model (over tokens not types)

Naïve Bayes: Learning

- From training corpus, extract *Vocabulary*
- Calculate required $P(c_j)$ and $P(x_k \mid c_j)$ terms
  - For each c_j in C do
    - $docs_j \leftarrow$ subset of documents for which the target class is c_j
    - $$P(c_j) \neg \frac{|docs_j|}{|\text{total \# documents}|}$$

    - $Text_j \leftarrow$ single document containing all $docs_j$
    - for each word x_k in *Vocabulary*
      - $n_k \leftarrow$ number of occurrences of x_k in $Text_j$
      - $$P(x_k \mid c_j) \neg \frac{n_k + \alpha}{n + \alpha\,|Vocabulary|}$$

Naïve Bayes: Classifying

- positions ← all word positions in current document which contain tokens found in *Vocabulary*
- Return c_{NB}, where

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_{i \in positions} P(x_i \mid c_j)$$

Naive Bayes: Time Complexity

- Training Time: $O(|D|L_d + |C||V|)$)
  where L_d is the average length of a document in *D.*
  - Assumes *V* and all D_i , n_i, and n_{ij} pre-computed in $O(|D|L_d)$ time during one pass through all of the data.

Why?

  - Generally just $O(|D|L_d)$ since usually $|C||V| < |D|L_d$
- Test Time: $O(|C|\ L_t)$
  where L_t is the average length of a test document.
- Very efficient overall, linearly proportional to the time needed to just read in all the data.

Underflow Prevention: log space

- Multiplying lots of probabilities, which are between 0 and 1 by definition, can result in floating-point underflow.
- Since log(*xy*) = log(*x*) + log(*y*), it is better to perform all computations by summing logs of probabilities rather than multiplying probabilities.
- Class with highest final un-normalized log probability score is still the most probable.

$$c_{NB} = \underset{c_j \in C}{\mathrm{argmax}} \log P(c_j) + \sum_{i \in positions} \log P(x_i \mid c_j)$$

- Note that model is now just max of sum of weights…

Note: Two Models

- Model 1: Multivariate Bernoulli
  - One feature X_w for each word in dictionary
  - X_w = true in document d if w appears in d
  - Naive Bayes assumption:
    - Given the document's topic, appearance of one word in the document tells us nothing about chances that another word appears
- This is the model used in the binary independence model in classic probabilistic relevance feedback in hand-classified data (Maron in IR was a very early user of NB)

Two Models

- Model 2: Multinomial = Class conditional unigram
  - One feature X_i for each word pos in document
    - feature's values are all words in dictionary
  - Value of X_i is the word in position *i*
  - Naïve Bayes assumption:
    - Given the document's topic, word in one position in the document tells us nothing about words in other positions
  - Second assumption:
    - Word appearance does not depend on position

$$P(X_i = w \mid c) = P(X_j = w \mid c)$$

for all positions *i,j*, word *w*, and class *c*

    - Just have one multinomial feature predicting all words

Parameter estimation

- Multivariate Bernoulli model:

$$\hat{P}(X_w = t \mid c_j) =$$ fraction of documents of topic c_j in which word *w* appears

- Multinomial model:

$$\hat{P}(X_i = w \mid c_j) =$$ fraction of times in which word *w* appears across all documents of topic c_j

  - Can create a mega-document for topic *j* by concatenating all documents in this topic
  - Use frequency of *w* in mega-document

Classification

- Multinomial vs Multivariate Bernoulli?

- Multinomial model is almost always more effective in text applications!
  - See results figures later

- See *IIR* sections 13.2 and 13.3 for worked examples with each model

Feature Selection: Why?

- Text collections have a large number of features
  - 10,000 – 1,000,000 unique words … and more
- May make using a particular classifier feasible
  - Some classifiers can't deal with 100,000 of features
- Reduces training time
  - Training time for some methods is quadratic or worse in the number of features
- Can improve generalization (performance)
  - Eliminates noise features
  - Avoids overfitting

Feature selection: how?

- Two ideas:
  - Hypothesis testing statistics:
    - Are we confident that the value of one categorical variable is associated with the value of another
    - Chi-square test
  - Information theory:
    - How much information does the value of one categorical variable give you about the value of another
    - Mutual information

- They're similar, but χ^2 measures confidence in association, (based on available statistics), while MI measures extent of association (assuming perfect knowledge of probabilities)

χ^2 statistic (CHI)

- $\chi 2$ is interested in $(f_o - f_e)^2/f_e$ summed over all table entries: is the observed number what you'd expect given the marginals?

$$\chi^2(j,a) = \sum (O-E)^2/E = (2-.25)^2/.25 + (3-4.75)^2/4.75$$
$$+ (500-502)^2/502 + (9500-9498)^2/9498 = 12.9 \ (p < .001)$$

- The null hypothesis is rejected with confidence .999,
- since 12.9 > 10.83 (the value for .999 confidence).

	Term = jaguar	*Term ≠ jaguar*
Class = auto	2 *(0.25)*	500 *(502)*
Class ≠ auto	3 *(4.75)*	9500 *(9498)*

expected: f_e

observed: f_o

χ^2 statistic (CHI)

There is a simpler formula for 2x2 χ^2:

$$\chi^2(t,c) = \frac{N \times (AD - CB)^2}{(A+C) \times (B+D) \times (A+B) \times (C+D)}$$

$A = \#(t,c)$	$C = \#(\neg t,c)$
$B = \#(t,\neg c)$	$D = \#(\neg t, \neg c)$

$N = A + B + C + D$

Value for complete independence of term and category?

Feature selection via Mutual Information

- In training set, choose *k* words which best discriminate (give most info on) the categories.
- The Mutual Information between a word, class is:

$$I(w,c) = \sum_{e_w \in \{0,1\}} \sum_{e_c \in \{0,1\}} p(e_w, e_c) \log \frac{p(e_w, e_c)}{p(e_w)p(e_c)}$$

  - For each word *w* and each category *c*

Feature selection via MI (contd.)

- For each category we build a list of *k* most discriminating terms.
- For example (on 20 Newsgroups):
  - ***sci.electronics:*** circuit, voltage, amp, ground, copy, battery, electronics, cooling, …
  - ***rec.autos:*** car, cars, engine, ford, dealer, mustang, oil, collision, autos, tires, toyota, …
- Greedy: does not account for correlations between terms
- Why?

Feature Selection

- Mutual Information
  - Clear information-theoretic interpretation
  - May select rare uninformative terms
- Chi-square
  - Statistical foundation
  - May select very slightly informative frequent terms that are not very useful for classification
- Just use the commonest terms?
  - No particular foundation
  - In practice, this is often 90% as good

Feature selection for NB

- In general feature selection is *necessary* for multivariate Bernoulli NB.
- Otherwise you suffer from noise, multi-counting

- “Feature selection” really means something different for multinomial NB. It means dictionary truncation
  - The multinomial NB model only has 1 feature
- This “feature selection” normally isn’t needed for multinomial NB, but may help a fraction with quantities that are badly estimated

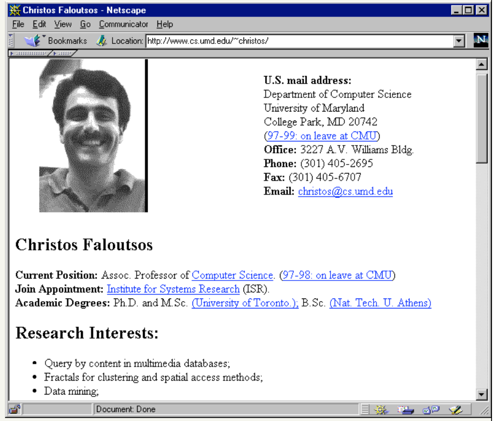
Evaluating Categorization

- Evaluation must be done on test data that are independent of the training data (usually a disjoint set of instances).
- *Classification accuracy*: *c*/*n* where *n* is the total number of test instances and *c* is the number of test instances correctly classified by the system.
  - Adequate if one class per document
  - Otherwise F measure for each class
- Results can vary based on sampling error due to different training and test sets.
- Average results over multiple training and test sets (splits of the overall data) for the best results.
- See *IIR* 13.6 for evaluation on Reuters-21578

13.6

WebKB Experiment (1998)

- Classify webpages from CS departments into:
  - student, faculty, course,project
- Train on ~5,000 hand-labeled web pages
  - Cornell, Washington, U.Texas, Wisconsin
- Crawl and classify a new site (CMU)


Christos Faloutsos - Netscape
File Edit View Go Communicator Help
Bookmarks Location: http://www.cs.umd.edu/~christos/
Document: Done

U.S. mail address:
Department of Computer Science
University of Maryland
College Park, MD 20742
(97-99: on leave at CMU)
Office: 3227 A.V. Williams Bldg.
Phone: (301) 405-2695
Fax: (301) 405-6707
Email: christos@cs.umd.edu

Christos Faloutsos

Current Position: Assoc. Professor of Computer Science. (97-98: on leave at CMU)
Join Appointment: Institute for Systems Research (ISR).
Academic Degrees: Ph.D. and M.Sc. (University of Toronto.); B.Sc. (Nat. Tech. U. Athens)

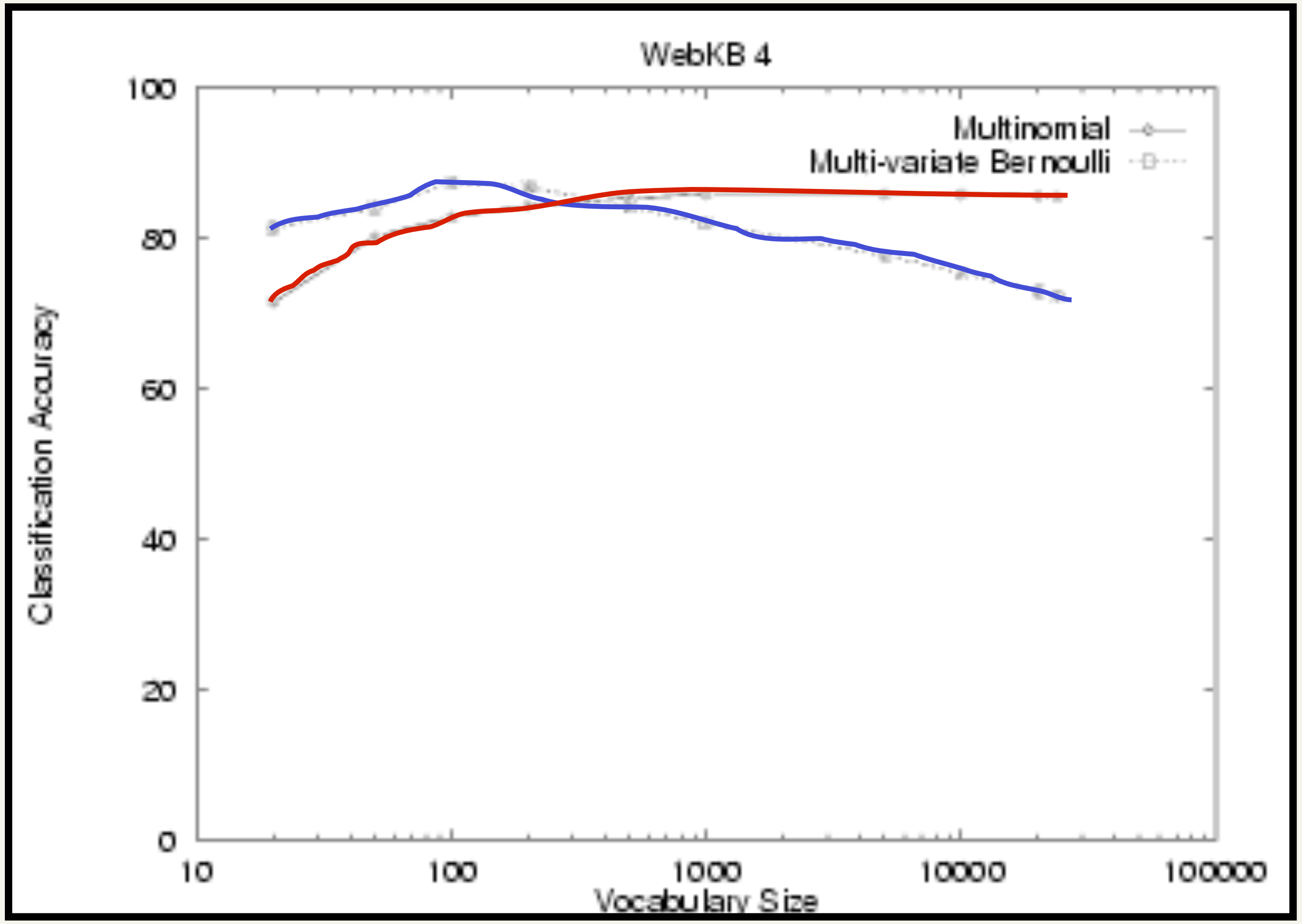
Research Interests:

- Query by content in multimedia databases;
- Fractals for clustering and spatial access methods;
- Data mining;

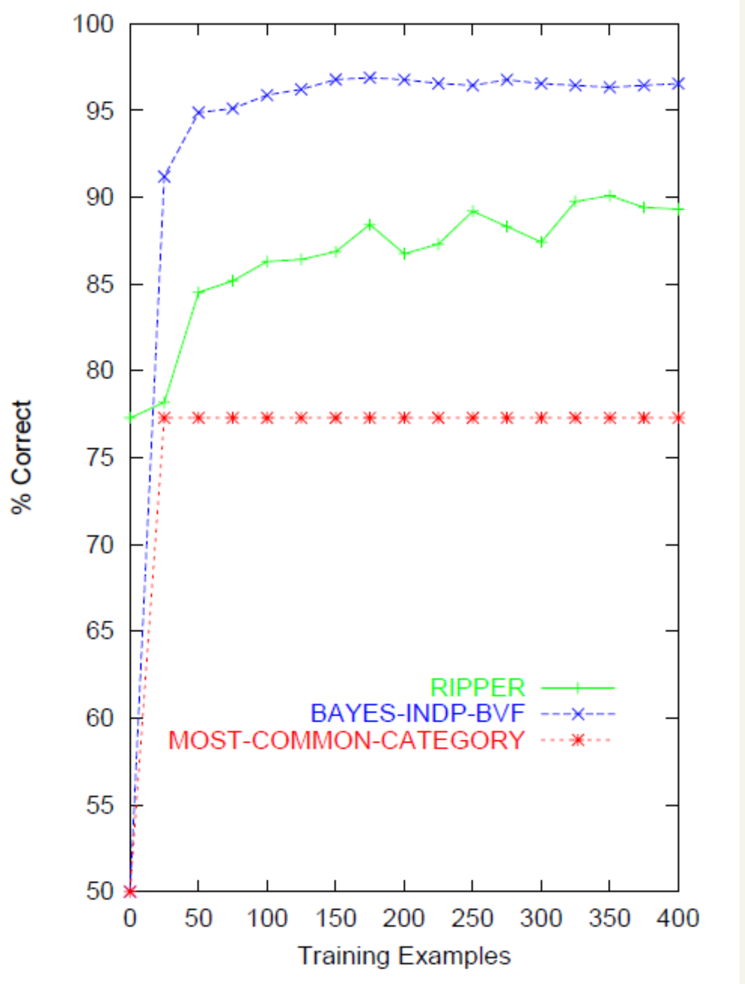
- Results:

	Student	Faculty	Person	Project	Course	Departmt
Extracted	180	66	246	99	28	1
Correct	130	28	194	72	25	1
Accuracy:	72%	42%	79%	73%	89%	100%

NB Model Comparison: WebKB



Naïve Bayes on spam email



13.6

SpamAssassin

- Naïve Bayes has found a home in spam filtering
  - Paul Graham's *A Plan for Spam*
    - A mutant with more mutant offspring...
  - Naive Bayes-like classifier with weird parameter estimation
  - Widely used in spam filters
    - Classic Naive Bayes superior when appropriately used
      - According to David D. Lewis
  - But also many other things: black hole lists, etc.

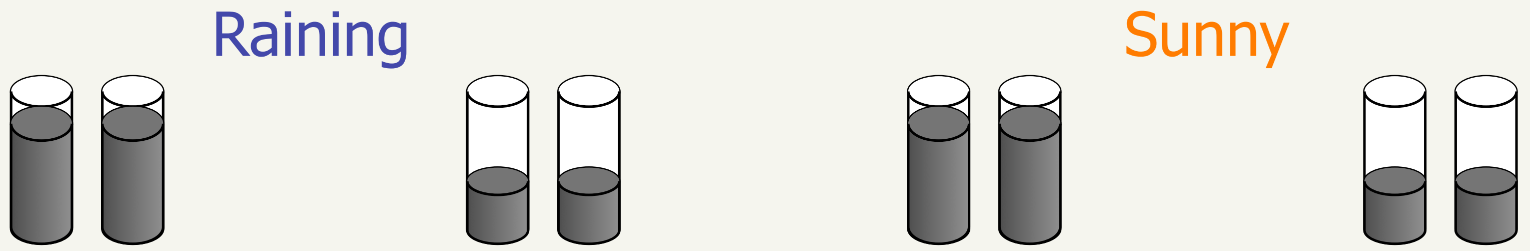
- Many email topic filters also use NB classifiers

Violation of NB Assumptions

- Conditional independence
- “Positional independence”
- Examples?

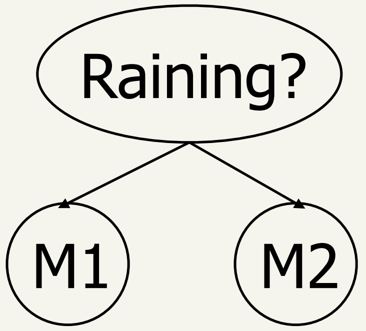
Example: Sensors

Reality



P(+,+,r) = 3/8 P(-,-,r) = 1/8 P(+,+,s) = 1/8 P(-,-,s) = 3/8

NB Model



NB FACTORS:

- P(s) = 1/2
- P(+|s) = 1/4
- P(+|r) = 3/4

PREDICTIONS:

- P(r,+,+) = (½)(¾)(¾)
- P(s,+,+) = (½)(¼)(¼)
- P(r|+,+) = 9/10
- P(s|+,+) = 1/10

Naïve Bayes Posterior Probabilities

- Classification results of naïve Bayes (the class with maximum posterior probability) are usually fairly accurate.
- However, due to the inadequacy of the conditional independence assumption, the actual posterior-probability numerical estimates are not.
  - Output probabilities are commonly very close to 0 or 1.
- Correct estimation $\Rightarrow$ accurate prediction, but correct probability estimation is NOT necessary for accurate prediction (just need right ordering of probabilities)

Naive Bayes is Not So Naive

- Naïve Bayes: First and Second place in KDD-CUP 97 competition, among 16 (then) state of the art algorithms

  Goal: Financial services industry direct mail response prediction model: Predict if the recipient of mail will actually respond to the advertisement – 750,000 records.

- Robust to Irrelevant Features

  Irrelevant Features cancel each other without affecting results

  Instead Decision Trees can heavily suffer from this.

- Very good in domains with many equally important features

  Decision Trees suffer from *fragmentation* in such cases – especially if little data

- A good dependable baseline for text classification (but not the best)!
- Optimal if the Independence Assumptions hold: If assumed independence is correct, then it is the Bayes Optimal Classifier for problem
- Very Fast: Learning with one pass of counting over the data; testing linear in the number of attributes, and document collection size
- Low Storage requirements

Resources

- IIR 13
- Fabrizio Sebastiani. Machine Learning in Automated Text Categorization. *ACM Computing Surveys*, 34(1):1-47, 2002.
- Yiming Yang & Xin Liu, A re-examination of text categorization methods. *Proceedings of SIGIR*, 1999.
- Andrew McCallum and Kamal Nigam. A Comparison of Event Models for Naive Bayes Text Classification. In *AAAI/ICML-98 Workshop on Learning for Text Categorization*, pp. 41-48.
- Tom Mitchell, Machine Learning. McGraw-Hill, 1997.
  - Clear simple explanation of Naïve Bayes
- Open Calais: Automatic Semantic Tagging
  - Free (but they can keep your data), provided by Thompson/Reuters
- Weka: A data mining software package that includes an implementation of Naive Bayes
- Reuters-21578 – the most famous text classification evaluation set and still widely used by lazy people (but now it's too small for realistic experiments – you should use Reuters RCV1)