machine learning for IR

some slides courtesy James Allan@umass
some slides from Chris Manning/Rada Mihalcea
Text and Machine Learning

- Information Retrieval
- Library and Information Science
- Artificial Intelligence
- Natural Language Processing
- Database Management
What is Machine Learning?

• A computer program is said to learn from experience $E$ with respect to some class of tasks $T$ and performance measure $P$, if its performance at tasks in $T$, as measured by $P$, improves with experience $E$. [Mitchell ’97]

• $T$: Classifying Text to some category
• $P$: Accuracy of Classification
• $E$: A training set
Given such a dataset one might want to:

- Learn to put instances into predefined classes (classification)
- Learn relationships between attributes (association learning)
- Groups similar instances together (clustering)

A fictional dataset:

<table>
<thead>
<tr>
<th>Name</th>
<th>Age</th>
<th>Sex</th>
<th>...</th>
<th>Risk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tom</td>
<td>32</td>
<td>M</td>
<td>...</td>
<td>Y</td>
</tr>
<tr>
<td>Mary</td>
<td>54</td>
<td>F</td>
<td>...</td>
<td>N</td>
</tr>
<tr>
<td>John</td>
<td>13</td>
<td>M</td>
<td>...</td>
<td>?</td>
</tr>
<tr>
<td>Kim</td>
<td>10</td>
<td>F</td>
<td>...</td>
<td>?</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
pattern classification

• Definitions:
  – **Instance**: Single example in the dataset \((X_i)\)
  – **Attribute**: An aspect of an instance \(x_j\)
  – **Value**: Value that an attribute can take
  – \(X = (X_1 \ldots X_n)\), a set of \(d\)-dimensional vectors (the data)
    • \(X_i = x_{1,i} \ldots x_{m,i}\)
  – \(Y = Y_1 \ldots Y_m\), a set of output classes
  – **Concept** – The thing to be learned

<table>
<thead>
<tr>
<th>Name</th>
<th>Age</th>
<th>Sex</th>
<th>...</th>
<th>Risk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tom</td>
<td>32</td>
<td>M</td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>Mary</td>
<td>54</td>
<td>F</td>
<td></td>
<td>N</td>
</tr>
<tr>
<td>John</td>
<td>13</td>
<td>M</td>
<td></td>
<td>?</td>
</tr>
<tr>
<td>Kim</td>
<td>10</td>
<td>F</td>
<td></td>
<td>?</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>
example concept
training and testing
• **Document Classification**

• **Standard datasets:**
  - **Reuters:** Reuters news articles in categories like earnings, acquisitions etc
  - **Newsgroups:** Newsgroups pages: Predict the newsgroup (comp.graphics, comp.os.ms-windows.misc, rec.sport.baseball, rec.sport.hockey etc)

<table>
<thead>
<tr>
<th>Docs</th>
<th>Features</th>
<th>w1</th>
<th>...</th>
<th>wn</th>
<th>...</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
classification

Cross Validation

- Collect data
- Choose features
- Choose model
- Train Classifier
- Evaluate Classifier
Supervised Learning

• Supervised learning
  – learning algorithm is provided with a set of inputs for the algorithm along with the corresponding correct outputs,
  – learning involves the algorithm comparing its current actual output with the correct or target outputs, so that it knows what its error is, and modify things accordingly.

• Unsupervised Learning
  – Example – regression, clustering
models

- **Discriminative Models:**
  \[ x \rightarrow g(x) \]

- **Generative models:**
  \[ x \rightarrow P(x|C) \]
  \[ P(C|x) \propto P(x|C)P(C) \]
  \[ g(x) = \frac{P(C|x)}{P(C|x)} \]
naive Bayes

\[
P(C|X) = \frac{P(X|C)P(C)}{P(X)}
\]

\[
P(X|C) = \prod_{i}^{V} p(x_i|C)
\]

- If \( P(C|X) > P(\bar{C}|X) \) then assign \( X \) to \( C \)
  - Intuitive. Also corresponds to the action where Bayes Risk is minimum

- Example of Generative Model
- Probabilities are Max likelihood with some form of smoothing
support vector machines

Find the best hyper-plane that separates the two classes

\[ w^T x + b < 0 \]
\[ w^T x + b > 0 \]
\[ w^T x + b = 0 \]

Example of a Generative Model

\[ f(x) = \text{sgn}(w^T x + b) \]
support vector machines

But what is the best hyper-plane?
support vector machines
support vector machines

- optimization problem

\[(w^*, b^*) = \arg\max_{(w, b)} \min_{X_i \in X} Y_i (w^T X_i + b)\]
Lagrange optimization
**svm**

- The solution is of the form

\[
f(x) = \text{sgn}(\sum_{i \in SV} \alpha_i y_i x_i^T x + b^*)
\]

- Support vectors are the only important data points in the training set

- Summation over number of support vectors
the kernel trick

\[ K((x, y)) = \phi(x)^T \phi(y) \]
IR as a Classification Problem

- Binary Classification
- Compare with Language Modeling Framework
Probabilistic IR models as classifiers

• BIR model: A generative classifier
  – Features are binary representing the presence or absence of each word in the vocabulary
  – Uses a multiple-Bernoulli model to model the class-conditional

\[
\log \frac{P(R \mid D)}{P(\overline{R} \mid D)} = \log \frac{P(D \mid R)P(R)}{P(D \mid \overline{R})P(\overline{R})}
\]

\[
= \log \left( \prod_{i : x_i = 1} \frac{P(x_i = 1 \mid R)}{P(x_i = 1 \mid \overline{R})} \prod_{i : x_i = 0} \frac{P(x_i = 0 \mid R)}{P(x_i = 0 \mid \overline{R})} \right)
\]
Probabilistic IR models as classifiers

• Language models

  – Appear to have abandoned the notion of IR as a binary classification problem: There is no reference to the class variable $R$!

  – However, if we imagine each document as a unique class, language models can be considered generative!

  – Language models rank the classes (documents) for each instance (query)!
Case for Discriminative models for IR

• Theoretical considerations
  – “One should solve the (classification) problem directly and never solve a more general problem (class-conditional) as an intermediate step” [Vapnik, 1998]
  – Discriminative models tend to have a lower asymptotic error as the training set size is increased [Ng and Jordan, NIPS 2002]
Case for Discriminative models for IR

• Modeling assumptions
  – Term conditional independence assumptions in LM not strictly valid
  
  – Multinomial distribution fails to model burstiness of terms [Teevan and Karger, SIGIR 2003]
  
  – Discriminative models make very few assumptions and let the data speak for itself!
Case for Discriminative models for IR

• Case for Discriminative models for IR

• Expressiveness : advanced features
  – Proximity of query terms
  – Ordering of terms
  – Presence or absence of terms

• Hard to include such features in LMs

• Discriminative models can handle arbitrary features
Case for Discriminative models for IR

• Learning arbitrary features
  – Multiple representations of documents
    • E.g.: abstract, title, anchor text, document content
  – Query-independent features
    • E.g.: Page Rank
    • User preferences

• Language models permit both but feature weights (typically) determined empirically

• Discriminative models can learn all such features automatically
IR vs. Text Classification

- IR not same as text classification!
  - IR is much harder: training data is very sparse
  - Dynamic vs. static classes: Distribution of words in the relevant class is query-specific
    - training on words as features will not help

- Features based on query-based statistics of documents instead
Unbalanced data

• Non-relevant class is represented by much larger number of training examples than the relevant class

• Discriminative classifiers trained on unbalanced data result in trivial classifiers

• Methods used to overcoming unbalanced data problem:
  – Oversampling minority class
  – Undersampling majority class
  – Adjusting misclassification cost of one of the classes
Ad-hoc Retrieval

- Task of retrieving a ranked list of relevant documents for a given free-text query
  - 4 different TREC collections used in the experiments: each collection has a set of train and test queries and relevance judgments
  - SVM and LM
  - The models trained on each collection and tested on all 4 collections: in total we have 16 runs
  - Documents and queries are pre-preprocessed using a stop-word list and the K-stemmer
Ad-hoc Retrieval

• Used title queries in all experiments

• Dirichlet smoothing is used in LM runs: training consists of finding the best value of Dirichlet parameter

• SVMs: linear kernels proved the best

• Discriminative models trained using all relevant examples and randomly sampled non-relevant examples

• *Lemur* for LMs, *SVM-light* for SVMs
Ad-hoc Retrieval

Features used in the discriminative models

1. $\sum_{i=1}^{n} \log(c(q_i, D))$

2. $\sum_{i=1}^{n} \log(1 + \frac{c(q_i, D)}{|D|})$

3. $\sum_{i:c(q_i,D)>0} \log(idf(q_i))$

4. $\sum_{i:q_i>0} \log \frac{|C|}{c(q_i, C)}$

5. $\sum_{i=1}^{n} \log \left(1 + \frac{|D|}{c(q_i, D)} idf(q_i) \right)$

6. $\sum_{i=1}^{n} \log \left(1 + \frac{|C|}{c(q_i, C)} \frac{c(q_i, D)}{|D|} \right)$
Out-Of-Vocab problem

- Words in test queries are mostly to have occurred in training queries.

- However, features are based not on words but on the term statistics.
## Adhoc Retrieval

<table>
<thead>
<tr>
<th>Train/Test</th>
<th>Disk 1-2 (151-200)</th>
<th>Disk 3 (101-150)</th>
<th>Disks 4-5 (401-450)</th>
<th>WT2G (426-450)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disk1-2 (101-150)</td>
<td></td>
<td></td>
<td>0.2377 (0.80)</td>
<td>0.2665 (0.61)</td>
</tr>
<tr>
<td>SVM</td>
<td>0.2145</td>
<td>0.1877 (0.3)</td>
<td>0.2356</td>
<td>0.2598</td>
</tr>
<tr>
<td>Disk3 (51-100)</td>
<td></td>
<td></td>
<td>0.2503 (0.21)</td>
<td>0.2666</td>
</tr>
<tr>
<td>SVM</td>
<td>0.2064</td>
<td>0.1728</td>
<td>0.2432</td>
<td>0.2750 (0.55)</td>
</tr>
<tr>
<td>Disk4-5 (301-350)</td>
<td></td>
<td></td>
<td>0.2516 (0.036)</td>
<td>0.2656</td>
</tr>
<tr>
<td>SVM</td>
<td>0.2078</td>
<td>0.1646</td>
<td>0.2355</td>
<td>0.2675 (0.89)</td>
</tr>
<tr>
<td>WT2G (401-425)</td>
<td></td>
<td></td>
<td>0.2335</td>
<td>0.2639</td>
</tr>
<tr>
<td>SVM</td>
<td>0.2199</td>
<td>0.1744</td>
<td>0.2487 (0.046)</td>
<td>0.2798 (0.037)</td>
</tr>
</tbody>
</table>
ad-hoc retrieval

• Conclusions
  – LMs, despite some inaccurate assumptions are quite robust!
  – class conditional models using a fixed distribution are relatively impervious to noise in training data
  – Simplicity helps in good generalization
    • Why use SVMs then?
  – Strength of SVMs: ability to learn relative importance of arbitrary features automatically
home page finding

• Task of retrieving the relevant document as high in the ranked list as possible.
  – Corpus is WT10G, a 10GB web collection.
  – 50 Queries for Training, 50 for development and 145 for testing

– Evaluation
  • Mean Reciprocal Rank (MRR)
  • Success rate
  • Failure rate
Features used in discriminative models

- Query-dependent features:
  - Document content
  - Anchor text
  - Title

- Query-independent features
  - Link factor
    \[ \log\left(1 + \frac{\text{num-links}(D)}{\text{Avg} - \text{num-links}}\right) \]
  - URL-depth: reciprocal of number of branches in the URL path of the document
home page finding

Results on the development set

<table>
<thead>
<tr>
<th>SVM features</th>
<th>MRR</th>
<th>Success %</th>
<th>Failure %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content + Anchor</td>
<td>0.54</td>
<td>73.0</td>
<td>5.2</td>
</tr>
<tr>
<td>Content + Anchor + Title</td>
<td>0.61</td>
<td>85.7</td>
<td>10.2</td>
</tr>
<tr>
<td>Content + Anchor + Title + URL</td>
<td>0.61</td>
<td>85.7</td>
<td>10.2</td>
</tr>
<tr>
<td>Content + Anchor + Title + URL + link</td>
<td>0.61</td>
<td>85.7</td>
<td>10.2</td>
</tr>
<tr>
<td>LM baseline</td>
<td>0.35</td>
<td>52.0</td>
<td>10.0</td>
</tr>
<tr>
<td>SVM baseline</td>
<td>0.33</td>
<td>53.06</td>
<td>12.24</td>
</tr>
</tbody>
</table>
home page finding

Results on test set
- Used all query-dependent and query-independent features

<table>
<thead>
<tr>
<th>Model</th>
<th>MRR</th>
<th>Success %</th>
<th>Failure %</th>
</tr>
</thead>
<tbody>
<tr>
<td>FullFeatured SVM</td>
<td>0.52</td>
<td>77.93</td>
<td>11.03</td>
</tr>
<tr>
<td>LM baseline</td>
<td>0.35</td>
<td>57.93</td>
<td>15.86</td>
</tr>
<tr>
<td>SVM Baseline</td>
<td>0.28</td>
<td>52.41</td>
<td>17.90</td>
</tr>
</tbody>
</table>
Different Learning Paradigms

- Inductive Learning – what you just saw
  - Learn from solved examples in a book. In-class closed book exam

- Active Learning
  - Only unsolved problems. Can ask an expert a few questions. In-class closed book exam

- Semi supervised learning
  - Book examples, back of the book questions. In-class closed book exam

- Transductive Learning
  - Book examples. Take home exam.
Active Learning

• In *Active Learning* the learner can ask an expert the labels of some of the unlabeled instances in order to improve classification accuracy.

• The objective is to ask the expert as few questions as possible.

• Uncertainty sampling is one way of Active Learning.
Active Learning

• Query by Committee [Freund, Sueng et al]
  – They prove theoretically that if a 2 member committee can achieve information gain with +ve lower bound then error decreases exponentially in the number of queries

• Uncertainty Sampling [Lewis and Gale]
  – Query on those instances that the Naïve Bayes classifier is most uncertain about \((p(Y|X) \approx 0.5)\)

• Optimize on expected future error[Roy, McCallum]

• Active Learning with Support Vector Machines [Tong, Koller]
  – Pick a sample such that the knowledge of the label reduces the version space in half.
Active Learning with a Naive Bayes Classifier

- Remember the Naïve Bayes Classifier

- The simplest way of uncertainty sampling is to query the user on instances with as close to 0.5 as possible.

\[
\frac{P(C|D)}{P(\bar{C}|D)} = \frac{P(C)}{P(\bar{C})} \times \frac{P(D|C)}{P(D|\bar{C})}
\]
active learning with SVM

- Consider a two class problem
- The SVM tries to find the best separating hyper-plane
- When all the data is labeled it's easy.
Uncertainty Sampling
active learning and SVMs

- For each instance that you pick, you halve the hypothesis space.

- In other words you halve the number of possible concepts that fit the data.
## Uncertainty Sampling

<table>
<thead>
<tr>
<th>Topic</th>
<th>SVM – Unc</th>
<th>Equivalent Random size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Earn</td>
<td>86.4</td>
<td>34</td>
</tr>
<tr>
<td>Acq</td>
<td>77.0</td>
<td>&gt;100</td>
</tr>
<tr>
<td>Money</td>
<td>93.8</td>
<td>50</td>
</tr>
<tr>
<td>Grain</td>
<td>95.5</td>
<td>13</td>
</tr>
<tr>
<td>Crude</td>
<td>95.26</td>
<td>&gt;100</td>
</tr>
</tbody>
</table>

Avg. test set accuracy on Reuters corpus. 2nd column is accuracy with 10 labeled instances using Uncertainty sampling with SVMs.
Maximum Likelihood Parameter Estimation

\[ P(X) \sim \theta \]

• For example, \( \theta = \mu, \sigma \) for a normal distribution.

• Write this as: \( P(X|\theta) \)

\[ \mathcal{D} = x_1 \ldots x_n \]

\[ p(\mathcal{D}|\theta) = \prod_{i=1}^{n} p(x_i|\theta) \]
MLE

Log Likelihood: \[ l(\theta) = \log p(D|\theta) \]

Maximum Likelihood Estimate:
\[ \hat{\theta} = \arg\max_{\theta} l(\theta) \]
FIGURE 3.1. The top graph shows several training points in one dimension, known or assumed to be drawn from a Gaussian of a particular variance, but unknown mean. Four of the infinite number of candidate source distributions are shown in dashed lines. The middle figure shows the likelihood $p(D|\theta)$ as a function of the mean. If we had a very large number of training points, this likelihood would be very narrow. The value that maximizes the likelihood is marked $\hat{\theta}$; it also maximizes the logarithm of the likelihood—that is, the log-likelihood $l(\theta)$, shown at the bottom. Note that even though they look similar, the likelihood $p(D|\theta)$ is shown as a function of $\theta$ whereas the conditional density $p(x|\theta)$ is shown as a function of $x$. Furthermore, as a function of $\theta$, the likelihood $p(D|\theta)$ is not a probability density function and its area has no significance. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.
Bayesian estimation

\[ P(x|\mathcal{D}) = \int p(x|\theta)p(\theta|\mathcal{D})d\theta \]

\[ P(\theta|\mathcal{D}) = \frac{P(\mathcal{D}|\theta)p(\theta)}{\int p(\mathcal{D})} \]

\[ P(\theta|\mathcal{D}) = \frac{P(\mathcal{D}|\theta)p(\theta)}{\int p(\mathcal{D}|\theta)p(\theta)d\theta} \]

• used for smoothing language models
text classification
Is this spam?

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
============================================================================
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 categories:
    \[ C = \{ c_1, c_2, \ldots, c_n \} \]

• Determine:
  – The category of \( x \): \( c(x) \in C \), where \( c(x) \) is a *categorization function* whose domain is \( X \) and whose range is \( C \).
    • We want to know how to build categorization functions (“classifiers”).
(Note: in real life there is often a hierarchy, not present in the above problem statement; and you get papers on ML approaches to Garb. Coll.)
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
  e.g., "like", "hate", "neutral"
- Labels may be domain-specific binary
  e.g., "interesting-to-me" : "not-interesting-to-me"
  e.g., "spam" : "not-spam"
  e.g., "is a toner cartridge ad" : "isn’t"
Methods (1)

• Manual classification
  – Used by Yahoo!, Looksmart, about.com, ODP, Medline
  – very accurate when job is done by experts
  – consistent when the problem size and team is small
  – difficult and expensive to scale

• Automatic document classification
  – Hand-coded rule-based systems
    • Used by CS dept’s spam filter, Reuters, CIA, Verity, ...
    • E.g., assign category if document contains a given boolean combination of words
    • Commercial systems have complex query languages (everything in IR query languages + *accumulators*)
Methods (2)

• Accuracy is often very high if a query has been carefully refined over time by a subject expert
• Building and maintaining these queries is expensive

• Supervised learning of document-label assignment function
  – Many new systems rely on machine learning (Autonomy, Kana, MSN, Verity, …)
    • 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 can be built (and refined) by non-experts
Text Categorization: attributes

- Representations of text are very high dimensional (one feature for each word).

- High-bias algorithms that prevent overfitting in high-dimensional space are best.

- For most text categorization tasks, there are many irrelevant and many relevant features.

- Methods that combine evidence from many or all features (e.g. naive Bayes, kNN, neural-nets) tend to work better than ones that try to isolate just a few relevant features (standard decision-tree or rule induction)*

*Although one can compensate by using many rules
Bayesian Methods

• 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.
Naive Bayes Classifiers

Task: Classify a new instance based on a tuple of attribute values

\[ \langle x_1, x_2, \ldots, x_n \rangle \]

\[ c_{MAP} = \arg\max_{c_j \in C} P(c_j \mid x_1, x_2, \ldots, x_n) \]

\[ c_{MAP} = \arg\max_{c_j \in C} \frac{P(x_1, x_2, \ldots, x_n \mid c_j)P(c_j)}{P(x_1, x_2, \ldots, x_n)} \]

\[ c_{MAP} = \arg\max_{c_j \in C} P(x_1, x_2, \ldots, x_n \mid c_j)P(c_j) \]
Naïve Bayes Classifier: Assumptions

- $P(c_j)$
  - Can be estimated from the frequency of classes in the training examples.

- $P(x_1, x_2, ..., x_n | c_j)$
  - $O(|X|^n \cdot |C|)$
  - Could only be estimated if a very, very large number of training examples was available.

Conditional Independence Assumption:

⇒ Assume that the probability of observing the conjunction of attributes is equal to the product of the individual probabilities.
The Naïve Bayes Classifier

- Conditional Independence Assumption: features are independent of each other given the class:

\[ P(X_1, \ldots, X_5 \mid C) = P(X_1 \mid C) \cdot P(X_2 \mid C) \cdot \cdots \cdot P(X_5 \mid C) \]
Learning the Model

- Common practice: maximum likelihood
  - simply use the frequencies in the data

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

\[ \hat{P}(x_i | c_j) = \frac{N(X_i = x_i, C = c_j)}{N(C = c_j)} \]
Problem with Max Likelihood

\[ P(X_1, \ldots, X_5 | C) = P(X_1 | C) \cdot P(X_2 | C) \cdot \cdots \cdot P(X_5 | C) \]

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

\[ \hat{P}(X_5 = t | C = nf) = \frac{N(X_5 = t, C = nf)}{N(C = nf)} = 0 \]

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

\[ \ell = \arg \max_c \hat{P}(c) \prod_i \hat{P}(x_i | c) \]
Smoothing to Avoid Overfitting

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

• Somewhat more subtle version

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

# of values of \(X_i\)

overall fraction in data where \(X_i=x_{i,k}\)

extent of “smoothing”
Naive Bayes Text Classification

- Attributes are text positions, values are words.

\[ c_{NB} = \arg\max_{c_j \in C} P(c_j) \prod_{i} P(x_i \mid c_j) \]

\[ = \arg\max_{c_j \in C} P(c_j) P(x_1 = "our" \mid c_j) \cdots P(x_n = "text" \mid c_j) \]

- Still too many possibilities
- Assume that classification is *independent* of the positions of the words
  
  – Use same parameters for each position
Text Classification Algorithms: 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 $c_j$
  
    • 
      $$P(c_j) \leftarrow \frac{|docs_j|}{|\text{total \# documents}|}$$

• $Text_j \leftarrow$ single document containing all $docs_j$
• for each word $x_k$ in *Vocabulary*
  – $n_k \leftarrow$ number of occurrences of $x_k$ in $Text_j$
  
  $P(x_k \mid c_j) \leftarrow \frac{n_k + 1}{n + |\text{Vocabulary}|}$
Text Classification Algorithms: Classifying

- Return $c_{NB}$, where

$$c_{NB} = \arg\max_{c_j} P(c_j) \prod_{c_j \in C} P(x_i \mid c_j)$$
General Learning Issues

- Many hypotheses are usually consistent with the training data.
  - Can derive many classification schemes

- Classification accuracy (% of instances classified correctly).
  - Measured on independent test data.

- Training time (efficiency of training algorithm).

- Testing time (efficiency of subsequent classification).
Text Categorization

Assigning documents to a fixed set of categories.

Applications:

- Web pages
  - Recommending
  - Yahoo-like classification
- Newsgroup Messages
  - Recommending
  - spam filtering
- News articles
  - Personalized newspaper
- Email messages
  - Routing
  - Prioritizing
  - Folderizing
  - spam filtering
Learning for Text Categorization

- Manual development of text categorization functions is difficult.

- Learning Algorithms:
  - Bayesian (naïve)
  - Neural network
  - Relevance Feedback (Rocchio)
  - Rule based (Ripper)
  - Nearest Neighbor (case based)
  - Support Vector Machines (SVM)
Using Relevance Feedback (Rocchio)

- Relevance feedback methods can be adapted for text categorization.

- Use standard TF/IDF weighted vectors to represent text documents (normalized by maximum term frequency).

- For each category, compute a *prototype* vector by summing the vectors of the training documents in the category.

- Assign test documents to the category with the closest prototype vector based on cosine similarity.
Assume the set of categories is \{c_1, c_2, \ldots, c_n\}
For \(i\) from 1 to \(n\) let \(p_i = <0, 0, \ldots, 0>\) (init. prototype vectors)

For each training example \(<x, c(x)> \in D\)

Let \(d\) be the frequency normalized TF/IDF term vector for doc \(x\)
Let \(i = j: (c_j = c(x))\)
(sum all the document vectors in \(c_i\) to get \(p_i\))
Let \(p_i = p_i + d\)

One vector per category
Rocchio Text Categorization Algorithm (Test)

Given test document $x$

Let $\mathbf{d}$ be the TF/IDF weighted term vector for $x$

Let $m = -2$ \hspace{1cm} (init. maximum $\text{cosSim}$)

For $i$ from 1 to $n$:

$(\text{compute similarity to prototype vector})$

Let $s = \text{cosSim}(\mathbf{d}, \mathbf{p}_i)$

if $s > m$

let $m = s$

let $r = c_i$ \hspace{1cm} (update most similar class prototype)

Return class $r$