Text Classification: Naïve Bayes Algorithm

SEEM5680
Document Classification

Test Data:

Classes:
- (AI)
- (Programming)
- (HCI)

Training Data:
- learning
- intelligence
- temporal
- algorithm
- reasoning
- reinforcement
- network...

(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.”)
Categorization/Classification

- Given:
  - A description of an instance, \( d \in X \)
    - \( X \) is the *instance language* or *instance space*.
      - Issue: how to represent text documents.
      - Usually some type of high-dimensional space
  - A fixed set of classes:
    \[ C = \{c_1, c_2, \ldots, c_J\} \]

- Determine:
  - The category of \( d \): \( \gamma(d) \in C \), where \( \gamma(d) \) is a *classification function* whose domain is \( X \) and whose range is \( C \).
    - We want to know how to build classification functions ("classifiers").
Supervised Classification

- Given:
  - A description of an instance, \( d \in X \)
    - \( X \) is the *instance language* or *instance space*.
  - A fixed set of classes:
    - \( C = \{c_1, c_2, \ldots, c_J\} \)
  - A training set \( D \) of labeled documents with each labeled document \( \langle d, c \rangle \in X \times C \)

- Determine:
  - A learning method or algorithm which will enable us to learn a classifier \( \gamma : X \rightarrow C \)
  - For a test document \( d \), we assign it the class \( \gamma(d) \in C \)
More Text Classification Examples

Many search engine functionalities use classification

Assigning labels to documents or web-pages:

- Labels are most often topics such as Yahoo-categories
  - "finance," "sports," "news>world>asia>business"
- Labels may be genres
  - "editorials" "movie-reviews" "news"
- Labels may be opinion on a person/product
  - “like”, “hate”, “neutral”
- Labels may be domain-specific
  - "interesting-to-me" : "not-interesting-to-me”
  - “contains adult language” : “doesn’t”
  - language identification: English, French, Chinese, …
  - search vertical: about Linux versus not
  - “link spam” : “not link spam”
Classification Methods (1)

- Manual classification
  - Used by the original Yahoo! Directory
  - 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.
    - It’s what Google Alerts is doing
      - Widely deployed in government and enterprise
    - Companies 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 + score 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
A Verity topic
A complex classification rule

- Note:
  - maintenance issues (author, etc.)
  - Hand-weighting of terms

[Verity was bought by Autonomy.]
Classification Methods (3)

- Supervised learning of a document-label assignment function
  - Many systems partly rely on machine learning (Autonomy, Microsoft, Enkata, Yahoo!, Google News, ...)
    - 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

- Many commercial systems use a mixture of methods
Recall a few probability basics

For events $a$ and $b$:

- Bayes’ Rule

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

$$p(\overline{a} \mid b) p(b) = p(b \mid \overline{a}) p(\overline{a})$$

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

- Odds:

$$O(a) = \frac{p(a)}{p(\overline{a})} = \frac{p(a)}{1 - p(a)}$$
Probabilistic Methods

- Learning and classification methods based on probability theory.
- Bayes theorem plays a critical role in probabilistic learning and classification.
- Builds 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 for text classification

- For a document \( d \) and a class \( c \)

\[
P(c,d) = P(c \mid d)P(d) = P(d \mid c)P(c)
\]

\[
P(c \mid d) = \frac{P(d \mid c)P(c)}{P(d)}
\]
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 \( c_j \in C \)

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

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

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

MAP is “maximum a posteriori” = most likely class
Naive Bayes Classifier: Naive Bayes Assumption

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

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

Naive 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 Naive Bayes Classifier

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

  \[ P(X_1, \ldots, X_5 \mid C) = P(X_1 \mid C) \cdot P(X_2 \mid C) \cdot \cdots \cdot P(X_5 \mid C) \]
First Naive Bayes Model

- Model 1: Multivariate Bernoulli
  - One feature \( X_w \) for each word in dictionary
  - \( X_w = \text{true} \) 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

- Model Learning

\[
\hat{P}(X_w = \text{true}|c_j) = \frac{\text{fraction of documents of topic } c_j \text{ in which word } w \text{ appears}}{}
\]
Multivariate Bernoulli Model

Learning the Model

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

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

\[
\hat{P}(X_i = t \mid c_j) = \frac{N(X_i = t, C = c_j)}{N(C = c_j)}
\]
Problem with Maximum Likelihood

What if we have seen no training documents with the word **muscle-ache** and classified in the topic **Flu**?

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

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

\[
\ell = \arg\max_c \hat{P}(c) \prod_i \hat{P}(X_i = t \mid c)
\]
Smoothing to Avoid Overfitting

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

# of values of \( X_i \)
Second Model

- Model 2: Multinomial = Class conditional unigram
  - One feature $X_i$ for each word position in document
    - feature’s values are all words in dictionary
  - Value of $X_i$ is the word in position $i$
  - Naive 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
Multinomial Naïve Bayes Model

\[ \hat{P}(X_i = w | c_j) = \text{fraction of times in which word } w \text{ appears among all words in documents of topic } c_j \]

- Can create a mega-document for topic \( j \) by concatenating all documents in this topic
- Use frequency of \( w \) in mega-document
Using Multinomial Naive Bayes Classifiers to Classify Text: Basic method

- Attributes are text positions, values are words.

$$c_{NB} = \arg\max_{c_j \in C} P(c_j) \prod_i P(x_i \mid c_j)$$

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

- Still too many possibilities
- Assume that classification is independent of the positions of the words
  - Use same parameters for each position
  - Result is bag of words model
Multinomial Naive Bayes: Learning

- From training corpus, extract *Vocabulary*
- Calculate required $P(c_j)$ and $P(x_k | c_j)$ terms
  - For each $c_j$ in $C$ do
    - $docs_j$ ← 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 $Text_j$
    - $P(x_k | c_j) \leftarrow \frac{n_k + 1}{n+|\text{Vocabulary}|}$
Multinomial Naive Bayes: Classifying

- $positions \leftarrow$ all word positions in current document which contain tokens found in $Vocabulary$

- Return $c_{NB}$, where

$$
c_{NB} = \arg\max_{c_j \in C} P(c_j) \prod_{i \in positions} P(x_i \mid c_j)
$$
Multinomial Naive Bayes: Example

<table>
<thead>
<tr>
<th>docID</th>
<th>words in document</th>
<th>in c = China?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Chinese Beijing Chinese</td>
<td>yes</td>
</tr>
<tr>
<td>2</td>
<td>Chinese Chinese Shanghai</td>
<td>yes</td>
</tr>
<tr>
<td>3</td>
<td>Chinese Macao</td>
<td>yes</td>
</tr>
<tr>
<td>4</td>
<td>Tokyo Japan Chinese</td>
<td>no</td>
</tr>
<tr>
<td>5</td>
<td>Chinese Chinese Chinese Tokyo Japan</td>
<td>?</td>
</tr>
</tbody>
</table>

\[
P(c) = \frac{3}{4} \quad P(\bar{c}) = \frac{1}{4}
\]

\[
P(\text{Chinese}|c) = \frac{(5 + 1)}{(8 + 6)} = \frac{6}{14} = \frac{3}{7}
\]

\[
P(\text{Toyko}|c) = P(\text{Japan}|c) = \frac{(0 + 1)}{(8 + 6)} = \frac{1}{14}
\]

\[
P(\text{Chinese}|\bar{c}) = \frac{(1 + 1)}{(3 + 6)} = \frac{2}{9}
\]

\[
P(\text{Toyko}|\bar{c}) = P(\text{Japan}|\bar{c}) = \frac{(1 + 1)}{(3 + 6)} = \frac{2}{9}
\]
Multinomial Naive Bayes: Example

\[ P(c) = \frac{3}{4}, \quad P(\bar{c}) = \frac{1}{4} \]

\[ P(\text{Chinese}|c) = \frac{(5 + 1)}{(8 + 6)} = \frac{6}{14} = \frac{3}{7} \quad P(\text{Toyko}|c) = P(\text{Japan}|c) = \frac{(0 + 1)}{(8 + 6)} = \frac{1}{14} \]

\[ P(\text{Chinese}|\bar{c}) = \frac{(1 + 1)}{(3 + 6)} = \frac{2}{9} \quad P(\text{Toyko}|\bar{c}) = P(\text{Japan}|\bar{c}) = \frac{(1 + 1)}{(3 + 6)} = \frac{2}{9} \]

\[ P(c|d_5) \propto \frac{3}{4} \cdot \left(\frac{3}{7}\right)^3 \cdot \frac{1}{14} \cdot \frac{1}{14} \approx 0.0003 \]

\[ P(\bar{c}|d_5) \propto \frac{1}{4} \cdot \left(\frac{2}{9}\right)^3 \cdot \frac{2}{9} \cdot \frac{2}{9} \approx 0.0001 \]

The classifier assigns the test document to \( c = \text{China} \)
Naive Bayes: Time Complexity

- **Training Time**: \( O(|D|L_{ave} + |C||V|) \)
  - where \( L_{ave} \) is the average length of a document in \( D \).
    - Assumes all counts are pre-computed in \( O(|D|L_{ave}) \) time during one pass through all of the data.
    - Generally just \( O(|D|L_{ave}) \) since usually \( |C||V| < |D|L_{ave} \)
- **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: using logs

- 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.

\[
c_{NB} = \arg\max_{c_j \in C} \left[ \log P(c_j) + \sum_{i \in \text{positions}} \log P(x_i | c_j) \right]
\]

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

\[ c_{NB} = \arg\max_{c_j \in C} [\log P(c_j) + \sum_{i \in \text{positions}} \log P(x_i | c_j)] \]

- Simple interpretation: Each conditional parameter \( \log P(x_i | c_j) \) is a weight that indicates how good an indicator \( x_i \) is for \( c_j \).
- The prior \( \log P(c_j) \) is a weight that indicates the relative frequency of \( c_j \).
- The sum is then a measure of how much evidence there is for the document being in the class.
- We select the class with the most evidence for it.
Feature Selection: Why?

- Text collections have a large number of features
  - 10,000 – 1,000,000 unique words … and more
- May allow 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 ($\chi^2$)
  - 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 $\chi^2$ measures confidence in association, (based on available statistics), while MI measures extent of association (assuming perfect knowledge of probabilities)
**$\chi^2$ 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).

<table>
<thead>
<tr>
<th></th>
<th>Term = jaguar</th>
<th>Term ≠ jaguar</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Class = auto</strong></td>
<td>2 (0.25)</td>
<td>500 (502)</td>
</tr>
<tr>
<td><strong>Class ≠ auto</strong></td>
<td>3 (4.75)</td>
<td>9500 (9498)</td>
</tr>
</tbody>
</table>

| 5          | 10000         |

**expected:** $f_e$

**observed:** $f_o$
There is a simpler formula for 2x2 $\chi^2$:

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

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>$C$</td>
</tr>
<tr>
<td>$B$</td>
<td>$D$</td>
</tr>
</tbody>
</table>

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

$N = A + B + C + D$

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 \( w \) and a class \( c \) is:

\[
l(w,c) = \sum_{e_w \in \{0,1\}} \sum_{e_c \in \{0,1\}} p(e_w, e_c) \log \frac{p(e_w, e_c)}{p(e_w)p(e_c)}
\]

where \( e_w = 1 \) when the document contains the word \( w \) (0 otherwise); \( e_c = 1 \) when the document is in class \( c \) (0 otherwise)
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
Feature Selection

- Mutual Information
  - Clear information-theoretic interpretation
  - May select very slightly informative frequent terms that are not very useful for classification

- Chi-square
  - Statistical foundation
  - May select rare uninformative terms

- 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).
  - Sometimes use cross-validation (averaging results over multiple training and test splits of the overall data)
- It’s easy to get good performance on a test set that was available to the learner during training (e.g., just memorize the test set).
- Measures: precision, recall, F1, classification accuracy
- **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
Naive Bayes vs. other methods

<table>
<thead>
<tr>
<th></th>
<th>NB</th>
<th>Rocchio</th>
<th>kNN</th>
<th>trees</th>
<th>SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>micro-avg-L (90 classes)</td>
<td>80</td>
<td>85</td>
<td>86</td>
<td>89</td>
<td></td>
</tr>
<tr>
<td>macro-avg (90 classes)</td>
<td>47</td>
<td>59</td>
<td>60</td>
<td></td>
<td>60</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>NB</th>
<th>Rocchio</th>
<th>kNN</th>
<th>trees</th>
<th>SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>earn</td>
<td>96</td>
<td>93</td>
<td>97</td>
<td>98</td>
<td>98</td>
</tr>
<tr>
<td>acq</td>
<td>88</td>
<td>65</td>
<td>92</td>
<td>90</td>
<td>94</td>
</tr>
<tr>
<td>money-fx</td>
<td>57</td>
<td>47</td>
<td>78</td>
<td>66</td>
<td>75</td>
</tr>
<tr>
<td>grain</td>
<td>79</td>
<td>68</td>
<td>82</td>
<td>85</td>
<td>95</td>
</tr>
<tr>
<td>crude</td>
<td>80</td>
<td>70</td>
<td>86</td>
<td>85</td>
<td>89</td>
</tr>
<tr>
<td>trade</td>
<td>64</td>
<td>65</td>
<td>77</td>
<td>73</td>
<td>76</td>
</tr>
<tr>
<td>interest</td>
<td>65</td>
<td>63</td>
<td>74</td>
<td>67</td>
<td>78</td>
</tr>
<tr>
<td>ship</td>
<td>85</td>
<td>49</td>
<td>79</td>
<td>74</td>
<td>86</td>
</tr>
<tr>
<td>wheat</td>
<td>70</td>
<td>69</td>
<td>77</td>
<td>93</td>
<td>92</td>
</tr>
<tr>
<td>corn</td>
<td>65</td>
<td>48</td>
<td>78</td>
<td>92</td>
<td>90</td>
</tr>
<tr>
<td>micro-avg (top 10)</td>
<td>82</td>
<td>65</td>
<td>82</td>
<td>88</td>
<td>92</td>
</tr>
<tr>
<td>micro-avg-D (118 classes)</td>
<td>75</td>
<td>62</td>
<td>n/a</td>
<td>n/a</td>
<td>87</td>
</tr>
</tbody>
</table>

Evaluation measure: $F_1$

Naive Bayes does pretty well, but some methods beat it consistently (e.g., SVM).
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)

Results:

<table>
<thead>
<tr>
<th></th>
<th>Student</th>
<th>Faculty</th>
<th>Person</th>
<th>Project</th>
<th>Course</th>
<th>Departmt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extracted</td>
<td>180</td>
<td>66</td>
<td>246</td>
<td>99</td>
<td>28</td>
<td>1</td>
</tr>
<tr>
<td>Correct</td>
<td>130</td>
<td>28</td>
<td>194</td>
<td>72</td>
<td>25</td>
<td>1</td>
</tr>
<tr>
<td>Accuracy</td>
<td>72%</td>
<td>42%</td>
<td>79%</td>
<td>73%</td>
<td>89%</td>
<td>100%</td>
</tr>
</tbody>
</table>
NB Model Comparison: WebKB
<table>
<thead>
<tr>
<th>Faculty</th>
<th>Students</th>
<th>Courses</th>
</tr>
</thead>
<tbody>
<tr>
<td>associate</td>
<td>resume</td>
<td>homework</td>
</tr>
<tr>
<td>chair</td>
<td>advisor</td>
<td>syllabus</td>
</tr>
<tr>
<td>member</td>
<td>student</td>
<td>assignments</td>
</tr>
<tr>
<td>ph</td>
<td>working</td>
<td>exam</td>
</tr>
<tr>
<td>director</td>
<td>stuff</td>
<td>grading</td>
</tr>
<tr>
<td>fax</td>
<td>links</td>
<td>midterm</td>
</tr>
<tr>
<td>journal</td>
<td>homepage</td>
<td>pm</td>
</tr>
<tr>
<td>recent</td>
<td>interests</td>
<td>instructor</td>
</tr>
<tr>
<td>received</td>
<td>personal</td>
<td>due</td>
</tr>
<tr>
<td>award</td>
<td>favorite</td>
<td>final</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Departments</th>
<th>Research Projects</th>
<th>Others</th>
</tr>
</thead>
<tbody>
<tr>
<td>departmental</td>
<td>investigators</td>
<td>type</td>
</tr>
<tr>
<td>colloquia</td>
<td>group</td>
<td>jan</td>
</tr>
<tr>
<td>department</td>
<td>members</td>
<td>enter</td>
</tr>
<tr>
<td>seminars</td>
<td>researchers</td>
<td>random</td>
</tr>
<tr>
<td>schedules</td>
<td>laboratory</td>
<td>program</td>
</tr>
<tr>
<td>webmaster</td>
<td>develop</td>
<td>net</td>
</tr>
<tr>
<td>events</td>
<td>related</td>
<td>time</td>
</tr>
<tr>
<td>facilities</td>
<td>arpa</td>
<td>format</td>
</tr>
<tr>
<td>people</td>
<td>affiliated</td>
<td>access</td>
</tr>
<tr>
<td>postgraduate</td>
<td>project</td>
<td>begin</td>
</tr>
</tbody>
</table>
Naive Bayes on spam email
SpamAssassin

- Naive 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

- The independence assumptions do not really hold of documents written in natural language.
  - Conditional independence
  - Positional independence
Naive Bayes Posterior Probabilities

- Classification results of naive 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

- Naive Bayes won 1st and 2nd place in KDD-CUP 97 competition out of 16 systems
  Goal: Financial services industry direct mail response prediction model: Predict if the recipient of mail will actually respond to the advertisement – 750,000 records.

- More robust to irrelevant features than many learning methods
  Irrelevant Features cancel each other without affecting results
  Decision Trees can suffer heavily from this.

- More robust to concept drift (changing class definition over time)

- 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: Bayes Optimal Classifier
  Never true for text, but possible in some domains

- Very Fast Learning and Testing (basically just count the data)

- Low Storage requirements
Resources

  - Clear simple explanation of Naive Bayes
- Open Calais: Automatic Semantic Tagging
  - Free *(but they can keep your data)*, provided by Thompson/Reuters (ex-ClearForest)
- Weka: A data mining software package that includes an implementation of Naive Bayes
- Reuters-21578 – the most famous text classification evaluation set
  - Still widely used by lazy people *(but now it’s too small for realistic experiments – you should use Reuters RCV1)*