# Data-analysis and Retrieval
# Introduction/Text Classification and Naive Bayes

Ad Feelders

Universiteit Utrecht

# DAR Part 2: Overview

- People
  - Lecturer: Ad Feelders
  - TAs: Wijnand van Woerkom, Athos van Kralingen and Lammert Westerdijk
- Data Mining / Text Mining
  - Classification
  - Regression
  - Word embeddings
  - Ordinal Classification / Ranking
  - Clustering
- Practical assignment
  - Learn a model that can rank search results for customers of Home Depot

# Literature for part 2

- Christopher D. Manning et al., Introduction to Information Retrieval, Cambridge University Press, 2008.
- Gareth James et al., An introduction to statistical learning with applications in R, Springer, 2013.
- Dan Jurafsky en James H. Martin, Speech and Language Processing (3rd ed. draft).
- Lecture notes ordinal classification.
- Slides of the lectures.

# What is Data Mining?

Data Mining as a subdiscipline of computer science:

is concerned with the *development and analysis of algorithms* for the (efficient) extraction of patterns and models from (large) data bases.

# Text Mining

- Text Mining is data mining applied to text data.
- Text data requires substantial pre-processing.
- This typically results in a large number of attributes (for example, the size of the dictionary).

# Data Mining Tasks

- Regression

  *Predict a numeric target variable.*
- Classification

  *Predict a categorical target variable.*
- Ranking / Ordinal Classification

  *Rank objects (e.g. documents with respect to relevance to a query).*
- Clustering

  *Find groups of similar objects.*

# Examples of Classification Problems

- Credit Scoring: will applicant default on the loan?
- Handwritten character recognition.
- SPAM filter: is e-mail message HAM or SPAM?
- Classification of news stories into topics.
- ...

# Formal Description of Classification Problems

In *classification* problems there is a class variable $C$ that assumes values in an (unordered) discrete set $\mathbb{C} = \{c_1, c_2, \ldots, c_J\}$.

The goal is to predict the class label given a vector of attribute values $\vec{x} = (x_1, \ldots, x_M)$ measured on the same object.

More formally, a classification function $\gamma$ is a function

$$\gamma : \mathbb{X} \to \mathbb{C} \tag{13.1}$$

that maps attribute vectors to classes.

A *learning algorithm* produces a classification function from a training set $\mathbb{D}$ of labeled attribute vectors $(\vec{x}, c) \in \mathbb{X} \times \mathbb{C}$.

# Problem is not deterministic!

- Consider all objects with the same attribute values.
- In most problems of interest these objects do not all have the same class label.
- Given an attribute vector $\vec{x}$ there is a probability distribution $P(c|\vec{x})$ over the class labels, where (typically) all classes have non-zero probability.
- Example: we do not expect all loan applicants with attribute values (age $=$ 25 , income $=$ 5000, job=data scientist) to have the same class label. Some will default, some will not.

# Different Approaches to Learning Classifiers from Data

1. Probabilistic
   - Generative Methods estimate the joint distribution $P(\vec{x}, c)$ and use the result to determine $P(c|\vec{x})$.
     Example: naive Bayes, discriminant analysis.
   - Discriminative methods estimate $P(c|\vec{x})$ directly, don't bother with modeling $P(\vec{x})$. Example: logistic regression, k-nearest neighbor.
2. Non-probabilistic learn a function that assigns $\vec{x}$ to a class directly, without estimation of probability distributions.
   Example: Support Vector Machine.

# Some Important Rules of Probability

1. Sum Rule:

$$P(X) = \sum_{y \in \mathbb{Y}} P(X, y)$$

2. Product Rule:

$$P(X, Y) = P(X \mid Y)P(Y)$$

3. If $X$ and $Y$ are independent, then:

$$P(X, Y) = P(X)P(Y), \text{ or equivalently: } P(X \mid Y) = P(X)$$

4. Likewise, if $X$ and $Y$ are independent given $Z$, then:

$$P(X, Y \mid Z) = P(X \mid Z)P(Y \mid Z)$$

or equivalently:

$$P(X \mid Y, Z) = P(X \mid Z)$$

# Generative Methods

To predict $C$ from $\vec{x}$, the quantity of interest is $P(C \mid \vec{x})$.

In the generative approach we use Bayes' rule:

$$P(C = c|\vec{x}) = \frac{P(\vec{x}, c)}{P(\vec{x})} = \frac{P(\vec{x}|c)P(c)}{P(\vec{x})} = \frac{P(\vec{x}|c)P(c)}{\sum_{c' \in \mathbb{C}} P(\vec{x}|c')P(c')} \tag{1}$$

to compute posterior class probabilities.

To minimize the probability of misclassification, assign $\vec{x}$ to class

$$c_{\text{map}}(\vec{x}) = \arg\max_{c \in \mathbb{C}} P(c|\vec{x}) \qquad \text{(cf. 13.3)}$$

The quantities on the right hand side of equation (1) have to be estimated from data. We have a training set $\mathbb{D}$ with $N$ examples:

$$\mathbb{D} = \{(\vec{x}_1, c_1), \ldots, (\vec{x}_N, c_N)\}$$

To estimate $P(c)$ we typically use

$$\hat{P}(c) = \frac{N_c}{N}, \qquad \text{(cf. 13.5)}$$

that is, the fraction of examples with class $c$ in the training set.

# Generative Methods

The estimation of $P(\vec{x}|c)$ is more complicated.

Suppose that all $M$ components of $\vec{x} = (x_1, \ldots, x_M)$ are binary, and we want to estimate $P(\vec{x}|c)$ for all possible values of $\vec{x}$.

How many probabilities would we have to estimate for each class?

# Generative Methods

The estimation of $P(\vec{x}|c)$ is more complicated.

Suppose that all $M$ components of $\vec{x} = (x_1, \ldots, x_M)$ are binary, and we want to estimate $P(\vec{x}|c)$ for all possible values of $\vec{x}$.

How many probabilities would we have to estimate for each class?

The attribute vector $\vec{x}$ has $2^M$ possible values!

For $M = 30$ this is already more than 1 billion possibilities!

This is one of the manifestations of the *curse of dimensionality*.

# Naive Bayes assumption

Assume all attributes are *independent* within each class.

Then we can factorize as follows:

$$P(\vec{x}|c) = P(x_1|c) \cdot P(x_2|c) \cdots P(x_M|c) = \prod_{i=1}^{M} P(x_i|c) \qquad (2)$$

How does this solve our problem?

# Naive Bayes assumption

Assume all attributes are *independent* within each class.

Then we can factorize as follows:

$$P(\vec{x}|c) = P(x_1|c) \cdot P(x_2|c) \cdots P(x_M|c) = \prod_{i=1}^{M} P(x_i|c) \qquad (2)$$

How does this solve our problem?

Now we only have to estimate $M = 30$ probabilities per class!

# Naive Bayes

Plugging the NB assumption into equation (1) we get:

$$P(C = c|\vec{x}) = \frac{\prod_{i=1}^{M} P(x_i|c)P(c)}{\sum_{c' \in \mathbb{C}} \prod_{i=1}^{M} P(x_i|c')P(c')} \qquad (3)$$

The denominator of (3) is the same for each class so it can be ignored if we only want to determine the class with highest posterior probability.

The NB allocation rule then becomes:

$$c_{\text{NB}}(\vec{x}) = \arg\max_{c \in \mathbb{C}} P(c) \prod_{i=1}^{M} P(x_i|c) \qquad \text{(cf. 13.3)}$$

# Estimation of NB with categorical attributes

Assume that all attributes are categorical.

The Maximum Likelihood estimate of $P(x_i|c)$ is:

$$\hat{P}(x_i|c) = \frac{\hat{P}(x_i, c)}{\hat{P}(c)} = \frac{N_{x_i,c}/N}{N_c/N} = \frac{N_{x_i,c}}{N_c}, \tag{4}$$

where $N_{x_i,c}$ denotes the number of examples in the data set with $X_i = x_i$ and $C = c$.

Laplace smoothing to prevent zero probability estimates:

$$\hat{P}(x_i|c) = \frac{N_{x_i,c} + 1}{\sum_{x_i' \in \mathbb{X}_i} N_{x_i',c} + 1} = \frac{N_{x_i,c} + 1}{N_c + |\mathbb{X}_i|}, \tag{5}$$

where $|\mathbb{X}_i|$ denotes the size of the domain of $X_i$.

# Example: Text Classification

Table 13.1 book:

| | docID | words in document | $c = China$? |
|---|---|---|---|
| training set | 1 | Chinese Beijing Chinese | yes |
| | 2 | Chinese Chinese Shanghai | yes |
| | 3 | Chinese Macao | yes |
| | 4 | Tokyo Japan Chinese | no |
| test set | 5 | Chinese Chinese Chinese Tokyo Japan | ? |

# Example: Text Classification

Representation in the Bernoulli model:

| docID | Chinese | Beijing | Shanghai | Macao | Tokyo | Japan | Topic = China? |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | yes |
| 2 | 1 | 0 | 1 | 0 | 0 | 0 | yes |
| 3 | 1 | 0 | 0 | 1 | 0 | 0 | yes |
| 4 | 1 | 0 | 0 | 0 | 1 | 1 | no |
| 5 | 1 | 0 | 0 | 0 | 1 | 1 | ? |

*We only record whether or not a word occurs in a document, not how many times it occurs.*

# Example: Text Classification

Notation: $c$ = about China; $\bar{c}$ = not about China, and $\hat{P}(Chinese|c)$ is shorthand for $\hat{P}(Chinese = 1|c)$, etc.

Parameter estimates (with Laplace smoothing):

$$\begin{aligned}
\hat{P}(Chinese|c) &= (3+1)/(3+2) = 4/5 \\
\hat{P}(Japan|c) = \hat{P}(Tokyo|c) &= (0+1)/(3+2) = 1/5 \\
\hat{P}(Beijing|c) = \hat{P}(Macao|c) = \hat{P}(Shanghai|c) &= (1+1)/(3+2) = 2/5
\end{aligned}$$

$$\begin{aligned}
\hat{P}(Chinese|\bar{c}) &= (1+1)/(1+2) = 2/3 \\
\hat{P}(Japan|\bar{c}) = \hat{P}(Tokyo|\bar{c}) &= (1+1)/(1+2) = 2/3 \\
\hat{P}(Beijing|\bar{c}) = \hat{P}(Macao|\bar{c}) = \hat{P}(Shanghai|\bar{c}) &= (0+1)/(1+2) = 1/3
\end{aligned}$$

Furthermore, $\hat{P}(c) = \frac{3}{4}$ and $\hat{P}(\bar{c}) = \frac{1}{4}$.

# Example: Text Classification

Naive Bayes prediction for document 5:

$$\begin{aligned}\hat{P}(c|(1,0,0,0,1,1)) &\propto \hat{P}(c)\cdot\hat{P}(\mathit{Chinese}|c)\cdot(1-\hat{P}(\mathit{Beijing}|c))\\ &\cdot(1-\hat{P}(\mathit{Shanghai}|c)\cdot(1-\hat{P}(\mathit{Macao}|c))\\ &\cdot\hat{P}(\mathit{Tokyo}|c)\cdot\hat{P}(\mathit{Japan}|c)\\ &= 3/4\cdot 4/5\cdot 3/5\cdot 3/5\cdot 3/5\cdot 1/5\cdot 1/5\approx 0.005\end{aligned}$$

Analogously

$$\hat{P}(\bar{c}|(1,0,0,0,1,1)) \propto 1/4\cdot(2/3)^6 \approx 0.022$$

Hence document 5 is classified as not being about China.

# Naive Bayes: Multinomial Model

- A potential disadvantage of the Bernoulli model for text classification is that it ignores multiple occurrences of terms in a document.
- This might be ok for very short documents (e.g. Tweets), but probably not for long documents.
- The Multinomial model is an alternative Naive Bayes model that takes term frequencies into account.

# Multinomial Model

In the multinomial model we assume

$$P(c|d) \propto P(c) \prod_{k=1}^{n_d} P(t_k|c), \tag{13.2}$$

where $d = \langle t_1, \ldots, t_{n_d} \rangle$, $t_k$ is the term occurring in position $k$ of document $d$, $P(t_k|c)$ is the conditional probability of term $t_k$ occurring (in any position) in a document of class $c$, and $n_d$ is the total number of terms occurring in document $d$.

For all $t \in V$, to estimate $P(t|c)$, we can use

$$\hat{P}(t|c) = \frac{T_{ct}}{\sum_{t' \in V} T_{ct'}} \tag{13.6}$$

where $T_{ct}$ is the number of occurrences of $t$ in training documents from class $c$, and $V$ is the vocabulary (collection of all terms considered).

# Multinomial Model: smoothing

Interpretation of (13.6): if we draw a word at random from a document of class $c$, the probability that we draw $t$ is $\hat{P}(t \mid c)$.

Again, (13.6) may produce unwanted zero probability estimates, so we apply Laplace smoothing:

$$\hat{P}(t|c) = \frac{T_{ct} + 1}{\sum_{t' \in V}(T_{ct'} + 1)} = \frac{T_{ct} + 1}{\sum_{t' \in V} T_{ct'} + |V|} \qquad (13.7)$$

where $|V|$ is the size of the vocabulary.

# Example: Text Classification

Representation in the Multinomial model:

| docID | Chinese | Beijing | Shanghai | Macao | Tokyo | Japan | Topic = China? |
|---|---|---|---|---|---|---|---|
| 1 | 2 | 1 | 0 | 0 | 0 | 0 | yes |
| 2 | 2 | 0 | 1 | 0 | 0 | 0 | yes |
| 3 | 1 | 0 | 0 | 1 | 0 | 0 | yes |
| 4 | 1 | 0 | 0 | 0 | 1 | 1 | no |
| 5 | 3 | 0 | 0 | 0 | 1 | 1 | ? |

# Multinomial Model: estimation

Parameter estimates:

$$
\begin{aligned}
\hat{P}(Chinese|c) &= (5+1)/(8+6) = 3/7 \\
\hat{P}(Japan|c) = \hat{P}(Tokyo|c) &= (0+1)/(8+6) = 1/14 \\
\hat{P}(Beijing|c) = \hat{P}(Macao|c) = \hat{P}(Shanghai|c) &= (1+1)/(8+6) = 1/7 \\
\\
\hat{P}(Chinese|\bar{c}) &= (1+1)/(3+6) = 2/9 \\
\hat{P}(Japan|\bar{c}) = \hat{P}(Tokyo|\bar{c}) &= (1+1)/(3+6) = 2/9 \\
\hat{P}(Beijing|\bar{c}) = \hat{P}(Macao|\bar{c}) = \hat{P}(Shanghai|\bar{c}) &= (0+1)/(3+6) = 1/9
\end{aligned}
$$

Furthermore, $\hat{P}(c) = \frac{3}{4}$ and $\hat{P}(\bar{c}) = \frac{1}{4}$.

# Multinomial Model: prediction

Multinomial Naive Bayes prediction for document 5:

$$\hat{P}(c|d_5) \propto 3/4 \cdot (3/7)^3 \cdot 1/14 \cdot 1/14 \approx 0.0003$$
$$\hat{P}(\bar{c}|d_5) \propto 1/4 \cdot (2/9)^3 \cdot 2/9 \cdot 2/9 \approx 0.0001$$

Hence, in contrast to the Bernoulli model, document 5 is now classified as being about China. How come?

# Multinomial Model: prediction

Multinomial Naive Bayes prediction for document 5:

$$\hat{P}(c|d_5) \propto 3/4 \cdot (3/7)^3 \cdot 1/14 \cdot 1/14 \approx 0.0003$$
$$\hat{P}(\bar{c}|d_5) \propto 1/4 \cdot (2/9)^3 \cdot 2/9 \cdot 2/9 \approx 0.0001$$

Hence, in contrast to the Bernoulli model, document 5 is now classified as being about China. How come?

The fact that the term *Chinese* occurs 3 times in the document now weighs more heavily than the negative indicators *Tokyo* and *Japan*.

What if we hadn't performed Laplace smoothing?

# Naive Bayes: Training

```
TrainMultinomialNB(ℂ, 𝔻)
  V ← ExtractVocabulary(𝔻)
  N ← CountDocs(𝔻)
  for each c ∈ ℂ
  do N_c ← CountDocsInClass(𝔻, c)
      prior[c] ← N_c / N
      text_c ← ConcatenateTextOfAllDocsInClass(𝔻, c)
      for each t ∈ V
      do T_ct ← CountTokensOfTerm(text_c, t)
      for each t ∈ V
      do condprob[t][c] ← (T_ct + 1) / Σ_t'(T_ct' + 1)
  return V, prior, condprob
```

# Naive Bayes: Testing

$\textsc{ApplyMultinomialNB}(\mathbb{C}, V, \mathit{prior}, \mathit{condprob}, d)$

$W \leftarrow \textsc{ExtractTokensFromDoc}(V, d)$
**for each** $c \in \mathbb{C}$
**do** $\mathit{score}[c] \leftarrow \log \mathit{prior}[c]$
    **for each** $t \in W$
    **do** $\mathit{score}[c] += \log \mathit{condprob}[t][c]$
**return** $\arg\max_{c \in \mathbb{C}} \mathit{score}[c]$

$$c_{\text{MNB}}(d) = \arg\max_{c \in \mathbb{C}} P(c) \prod_{k=1}^{n_d} P(t_k|c),$$

or (same thing)

$$c_{\text{MNB}}(d) = \arg\max_{c \in \mathbb{C}} \log P(c) + \sum_{k=1}^{n_d} \log P(t_k|c).$$

# Time complexity of Naive Bayes

| mode | time complexity |
|---|---|
| training | $\Theta(\lvert\mathbb{D}\rvert L_{\text{ave}} + \lvert\mathbb{C}\rvert\lvert V\rvert)$ |
| testing | $\Theta(\lvert\mathbb{C}\rvert L_{\text{a}})$ |

- $L_{\text{ave}}$: average length of a training doc, $L_{\text{a}}$: length of the test doc, $\mathbb{D}$: training set, $V$: vocabulary, $\mathbb{C}$: set of classes
- $\Theta(|\mathbb{D}|L_{\text{ave}})$ is the time it takes to compute all counts.
- $\Theta(|\mathbb{C}||V|)$ is the time it takes to compute the parameters from the counts.
- Generally: $|\mathbb{C}||V| < |\mathbb{D}|L_{\text{ave}}$
- Test time is also linear (in the length of the test document).
- Thus: Naive Bayes is linear in the size of the training set (training) and the test document (testing).

# Violation of Naive Bayes independence assumptions

- Conditional independence:

$$P(\langle t_1, \ldots, t_{n_d} \rangle | c) = \prod_{1 \leq k \leq n_d} P(t_k | c)$$

- Positional independence: probability of a term is the same, regardless of the position in the document.
- The independence assumptions do not really hold for documents written in natural language.
- How can Naive Bayes work if it makes such "heroic" assumptions?

# Why does Naive Bayes work?

- Naive Bayes can work well even though independence assumptions are *badly* violated.
- Example:

| | $c_1$ | $c_2$ | class selected |
|---|---|---|---|
| true probability $P(c\|d)$ | 0.6 | 0.4 | $c_1$ |
| $\hat{P}(c) \prod_{1 \leq k \leq n_d} \hat{P}(t_k\|c)$ | 0.00099 | 0.00001 | |
| NB estimate $\hat{P}(c\|d)$ | 0.99 | 0.01 | $c_1$ |

- Double counting of evidence causes underestimation (0.01) and overestimation (0.99).
- Classification is about predicting the correct class, *not* about accurate estimation.

# Naive Bayes is not so naive

- Probability estimates may be way off, but that doesn't have to hurt classification performance (much).
- Requires the estimation of relatively few parameters, which may be beneficial if you have a small training set.
- Fast, low storage requirements

# Single-label and Multi-label classification

- In single-label (one-of) classification each object is assigned a single label from a set of mutually exclusive labels.
- This is the standard setting for the literature on classification.
- In multi-label (any-of) classification each object is assigned 0 or more labels. This is quite common in document classification where a document can be about more than 1 topic (e.g. sports and politics).
- This is often handled by making a binary classifier for each class label. For example, in training the classifier for *sports*, all documents that do not have the label sports are used as negative examples.

# Evaluating classification

- To get a reliable assessment of the performance of a classifier, it must be tested on data that was not used for training.
- The performance estimate on the training data is usually too optimistic, in particular if the classifier is able to adapt well to peculiarities of the training sample. This phenomenon is called *overfitting*.
- Performance measures: classification accuracy, precision, recall, $F_1$.

# Accuracy, Precision and Recall

| prediction/truth | in the class | not in the class |
|---|---|---|
| in the class | true positives (TP) | false positives (FP) |
| not in the class | false negatives (FN) | true negatives (TN) |

TP, FP, FN, TN are counts of documents. The sum of these four counts is the total number of test documents $N_{\text{test}}$.

- Accuracy is the fraction of correct predictions:

$$\text{Accuracy} = \frac{TP + TN}{N_{\text{test}}}$$

- Precision:

$$P = TP/(TP + FP)$$

- Recall:

$$R = TP/(TP + FN)$$

# A combined measure: $F$

- Precision and recall only measure a single aspect of performance. We can easily get a recall of 1 simply by classifying all documents as *in the class*.
- $F_1$ allows us to trade off precision against recall.
- Definition:

$$F_1 = \frac{2P \times R}{P + R}$$

- This is the harmonic mean of $P$ and $R$.

# Averaging: Micro vs. Macro

- We now have an evaluation measure ($F_1$) for each class.
- But we also want a single number that measures the aggregate performance over all classes in the collection.
- Macroaveraging
  - Compute $F_1$ for each of the $C$ classes
  - Average these $C$ numbers
- Microaveraging
  - Compute TP, FP, FN for each of the $C$ classes
  - Sum these $C$ numbers (e.g., all TP to get aggregate TP)
  - Compute $F_1$ for aggregate TP, FP, FN

# Naive Bayes vs. other methods on Reuters-21578 Corpus

(a)

| | NB | Rocchio | kNN | | SVM |
|---|---|---|---|---|---|
| micro-avg-L (90 classes) | 80 | 85 | 86 | | 89 |
| macro-avg (90 classes) | 47 | 59 | 60 | | 60 |

(b)

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

Evaluation measure: $F_1$

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

# Feature selection

- In text classification, we usually represent documents in a high-dimensional space, with each dimension corresponding to a term.
- Many dimensions correspond to rare words.
- Rare words can mislead the classifier.
- Rare misleading features are called noise features.
- Eliminating noise features from the representation increases efficiency and effectiveness of text classification.
- Eliminating features is called feature selection.

# Example for a noise feature

- Let's say we're doing text classification for the class *China*.
- Suppose a rare term, say ARACHNOCENTRIC, has no information about *China* . . .
- . . . but all instances of ARACHNOCENTRIC happen to occur in *China* documents in our training set.
- Then we may learn a classifier that incorrectly interprets ARACHNOCENTRIC as evidence for the class *China*.
- Such an incorrect generalization from an accidental property of the training set is called overfitting.
- Feature selection reduces overfitting and improves the accuracy of the classifier.

# Different feature selection methods

- A feature selection method is mainly defined by the feature utility measure it employs
- Feature utility measures:
  - Frequency – select the most frequent terms
  - Mutual information – select the terms that have the highest mutual information with the class label
  - Chi-square (see book)
- Sort features by utility and select top $M$.
- Can we miss good sets of features this way?

## Entropy and Conditional Entropy

Entropy is the average amount of information generated by observing the value of a random variable:

$$H(X) = \sum_{x \in \mathbb{X}} P(x) \log_2 \frac{1}{P(x)} = -\sum_{x \in \mathbb{X}} P(x) \log_2 P(x)$$

We can also interpret it as a measure of the uncertainty about the value of $X$ prior to observation.

Conditional entropy:

$$H(X \mid Y) = \sum_{x,y} P(x,y) \log_2 \frac{1}{P(x \mid y)} = -\sum_{x,y} P(x,y) \log_2 P(x \mid y)$$

# Mutual Information

For (categorical) random variables $X$ and $Y$, their mutual information is given by

$$I(X;Y) = H(X) - H(X \mid Y) = H(Y) - H(Y \mid X)$$
$$= \sum_{x \in \mathbb{X}} \sum_{y \in \mathbb{Y}} P(x,y) \log_2 \frac{P(x,y)}{P(x)P(y)}$$

- Mutual information measures the reduction in uncertainty about $X$ achieved by observing the value of $Y$ (and vice versa).
- If $X$ and $Y$ are independent, then for all $x, y$ we have $P(x,y) = P(x)P(y)$, so $I(X,Y) = 0$.
- Otherwise $I(X;Y)$ is a positive quantity.

# Mutual Information

To estimate $I(X;Y)$ from data we compute

$$I(X;Y) = \sum_x \sum_y \hat{P}(x,y) \log_2 \frac{\hat{P}(x,y)}{\hat{P}(x)\hat{P}(y)},$$

where

$$\hat{P}(x,y) = \frac{N_{xy}}{N} \qquad \hat{P}(x) = \frac{N_x}{N},$$

and $N_{xy}$ denotes the number of records with $X = x$ and $Y = y$. Plugging-in these estimates we get:

$$\begin{aligned} I(X;Y) &= \sum_x \sum_y \frac{N_{xy}}{N} \log_2 \frac{N_{xy}/N}{(N_x/N)(N_y/N)} \\ &= \sum_x \sum_y \frac{N_{xy}}{N} \log_2 \frac{N\,N_{xy}}{N_x\,N_y} \end{aligned}$$

# Mutual information between term presence and class

- Compute the feature utility $A(t, c)$ as the mutual information of term $t$ and class $c$.
- Definition:

$$I(U;C) = \sum_{e_t, e_c \in \{1,0\}^2} P(U = e_t, C = e_c) \log_2 \frac{P(U = e_t, C = e_c)}{P(U = e_t)P(C = e_c)} \quad (13.16)$$

Where $e_t = 0$ means the document does not contain term $t$, and $e_c = 0$ means the document does not belong to class $c$, etc.

# How to compute MI values

- Based on maximum likelihood estimates, the formula we actually use is:

$$
\begin{aligned}
I(U;C) &= \frac{N_{11}}{N}\log_2\frac{NN_{11}}{N_{1.}N_{.1}} + \frac{N_{01}}{N}\log_2\frac{NN_{01}}{N_{0.}N_{.1}} \\
&\quad + \frac{N_{10}}{N}\log_2\frac{NN_{10}}{N_{1.}N_{.0}} + \frac{N_{00}}{N}\log_2\frac{NN_{00}}{N_{0.}N_{.0}}
\end{aligned}
$$

- $N_{10}$: number of documents that contain $t$ ($e_t = 1$) and are not in $c$ ($e_c = 0$); etc. $N_{0.} = N_{00} + N_{01}$ and $N = N_{00} + N_{01} + N_{10} + N_{11}$.

# MI example for *poultry*/EXPORT in Reuters

| | $e_c = e_{poultry} = 1$ | $e_c = e_{poultry} = 0$ |
|---|---|---|
| $e_t = e_{\text{EXPORT}} = 1$ | $N_{11} = 49$ | $N_{10} = 27{,}652$ |
| $e_t = e_{\text{EXPORT}} = 0$ | $N_{01} = 141$ | $N_{00} = 774{,}106$ |

Plug these values into formula:

$$\begin{aligned} I(U;C) &= \frac{49}{801{,}948}\log_2\frac{801{,}948\cdot 49}{(49+27{,}652)(49+141)} \\ &\quad+\frac{141}{801{,}948}\log_2\frac{801{,}948\cdot 141}{(141+774{,}106)(49+141)} \\ &\quad+\frac{27{,}652}{801{,}948}\log_2\frac{801{,}948\cdot 27{,}652}{(49+27{,}652)(27{,}652+774{,}106)} \\ &\quad+\frac{774{,}106}{801{,}948}\log_2\frac{801{,}948\cdot 774{,}106}{(141+774{,}106)(27{,}652+774{,}106)} \\ &\approx 0.000105 \end{aligned}$$

# Example Reuters document


```
<?xml version="1.0"?>
<REUTERS TOPICS="YES" LEWISSPLIT="TRAIN" CGISPLIT="TRAINING-SET" OLDID="353" NEWID="12125">
 <DATE>1-APR-1987 13:07:30.08</DATE>
 <TOPICS>
  <D>interest</D>
 </TOPICS>
 <PLACES>
  <D>usa</D>
 </PLACES>
 <PEOPLE/>
 <ORGS/>
 <EXCHANGES/>
 <COMPANIES/>
 <UNKNOWN>A
   f1811 reute
r f BC-SUNTRUST-BANKS-&lt;STI&gt;   04-01 0043</UNKNOWN>
 <TEXT>
  <TITLE>SUNTRUST BANKS &lt;STI&gt; RAISES PRIME TO 7-3/4 PCT</TITLE>
  <DATELINE>NEW YORK, April 1 -</DATELINE>
  <BODY>SunTrust Banks said that Sun Banks in
Florida and Trust Co banks in Georgia have raised their prime
rate to 7-3/4 pct from 7-1/2 pct.
    The company said the action is effective immediately.
 Reuter</BODY>
 </TEXT>
</REUTERS>
```

# MI feature selection on Reuters

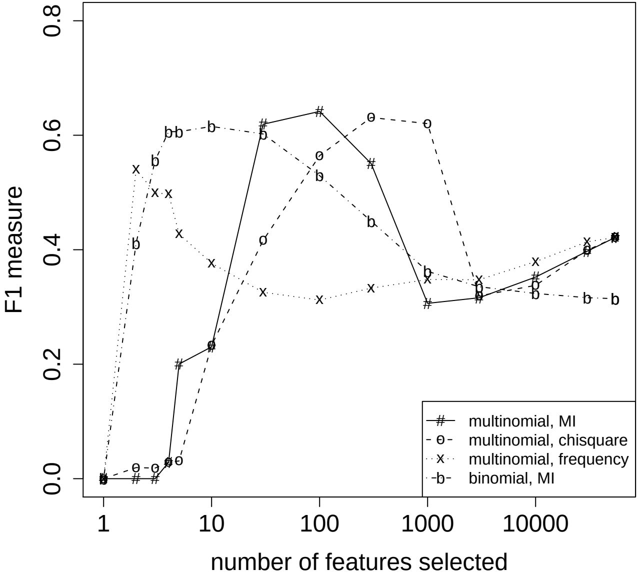
Class: *coffee*

| term | MI |
|---|---|
| COFFEE | 0.0111 |
| BAGS | 0.0042 |
| GROWERS | 0.0025 |
| KG | 0.0019 |
| COLOMBIA | 0.0018 |
| BRAZIL | 0.0016 |
| EXPORT | 0.0014 |
| EXPORTERS | 0.0013 |
| EXPORTS | 0.0013 |
| CROP | 0.0012 |

Class: *sports*

| term | MI |
|---|---|
| SOCCER | 0.0681 |
| CUP | 0.0515 |
| MATCH | 0.0441 |
| MATCHES | 0.0408 |
| PLAYED | 0.0388 |
| LEAGUE | 0.0386 |
| BEAT | 0.0301 |
| GAME | 0.0299 |
| GAMES | 0.0284 |
| TEAM | 0.0264 |

# Naive Bayes: Effect of feature selection



# An Example Analysis in R

- Can you predict music genre from song lyrics?
- Data: 20 Blues and 20 Metal songs.
- Metal lyrics obtained from www.darklyrics.com
- Blues lyrics obtained from www.bluesforpeace.com
- Analysis in R with `tm` package for text mining and package `e1071` with naive Bayes function.

# Reading in Text Data

```
# load package tm
> library(tm)
# read in texts from specified directory
> lyrics.metal <- VCorpus(DirSource("C:/Lyrics/Metal"),
readerControl=list(reader=readLyrics))
> lyrics.blues <- VCorpus(DirSource("C:/Lyrics/Blues"),
readerControl=list(reader=readLyrics))
# merge into one corpus
> lyrics <- c(lyrics.metal,lyrics.blues)
```

# Example document


```
<?xml version="1.0"?>
<Lyrics>
<genre>Metal</genre>
<artist>Neurosis</artist>
<album>Given To The Rising</album>
<track>Given To The Rising</track>
<text>
We stand encircled by wing and fire
Our deepest ties return and turn upon us
The shrouded reason, the bleeding answer
The human plague in womb
Bring clouds of war
Let us rest
Our future breed is the last
In the conscience waits
Dreams of the new sun
We're blood in the dust
Given to the Rising
Through this we claw roots
Of trees in the world of iron
Our father's steps fueled the boiling sea
The wretched harvest reaped by the hands of dawning
Our pain cannot forgive the silent machine of the fatal flaw in man
That brings us to the end
</text>
</Lyrics>
```

# Blues or Metal?

```
All tears, restrained for years
Their grief is confined
Which destroys my mind

An ode to their plight is this dirge
Some yearn for lugubrious silence
Serenity in the image of the coffins

Shall life renew these bodies of a truth?
All death will he annul, all tears assuage?
Fill the void veins of life, again with youth
And wash with an immortal water, age
They die.
```

# Blues or Metal or neither?

```
Ooh-ooh-ooh, ooh-ooh, ooh-ooh
Ooh-ooh-ooh, ooh-ooh, ooh-ooh
A broken heart is all that's left
I'm still fixing all the cracks
Lost a couple of pieces when
I carried it, carried it, carried it home
I'm afraid of all I am
My mind feels like a foreign land
Silence ringing inside my head
Please, carry me, carry me, carry me home
I spent all of the love I've saved
We were always a losing game
Small-town boy in a big arcade
I got addicted to a losing game
Oh-oh-oh-oh, oh-oh-oh-oh
All I know, all I know
Loving you is a losing game
How many pennies in the slot?
Giving us up didn't take a lot
I saw the end 'fore it begun
Still I carried, I carried, I carried on
Oh-oh-oh-oh, oh-oh-oh-oh
All I know, all I know
Loving you is a losing game
Oh-oh-oh-oh, oh-oh-oh-oh
```

# Preprocessing

```
# perform pre-processing on the corpus
> lyrics <- tm_map(lyrics,stripWhitespace)
> lyrics <- tm_map(lyrics,
removeWords,stopwords("english"))
> lyrics <- tm_map(lyrics, removePunctuation)
# sample indices of training data
> lyrics.train <- c(sample(1:20,15),sample(21:40,15))
# extract document-term matrix from training corpus
> lyrics.train.dtm <-
DocumentTermMatrix(lyrics[lyrics.train])
> dim(lyrics.train.dtm)
[1] 30 1102
# remove terms that occur in less than 15% of documents
> lyrics.train.dtm <-
removeSparseTerms(lyrics.train.dtm,0.85)
> dim(lyrics.train.dtm)
[1] 30 43
```

# Extracting Terms from Test Corpus

```
# extract column names from document-term matrix
> lyrics.dict <- dimnames(lyrics.train.dtm)[[2]]
# use them as a dictionary to extract terms from test
corpus
> lyrics.test.dtm <-
DocumentTermMatrix(lyrics[-lyrics.train],
list(dictionary=lyrics.dict))
```

# The Dictionary (Vocabulary)

```
> lyrics.dict

 [1] "aint"  "all"   "and"   "baby"
 [5] "back"  "bad"   "blood" "but"
 [9] "cause" "come"  "death" "even"
[13] "eyes"  "find"  "from"  "get"
[17] "gonna" "got"   "home"  "hope"
[21] "just"  "keep"  "know"  "life"
[25] "like"  "love"  "made"  "make"
[29] "man"   "mind"  "night" "now"
[33] "one"   "pain"  "place" "still"
[37] "the"   "they"  "will"  "woman"
[41] "world" "yeah"  "you"
```

# Conversion for Bernoulli NB modeling

```
# convert training data
# convert dtm to "normal" matrix
> lyrics.train.dat <- as.matrix(lyrics.train.dtm)
# make the matrix binary for Bernoulli model
> lyrics.train.bin <- lyrics.train.dat > 0
# required for naiveBayes function:
# convert matrix to data frame
> lyrics.train.bin <- as.data.frame(lyrics.train.bin)
# make all attributes (columns) categorical
> for(i in 1:43){lyrics.train.bin[,i] <-
as.factor(lyrics.train.bin[,i])}
# extract class labels
> lyrics.lab <-
as.vector(unlist(lapply(lyrics,meta,tag="genre")))
> lyrics.lab <- as.factor(lyrics.lab)
```

# Model fitting and prediction with naive Bayes

```
# load package with naive Bayes function
> library(e1071)
# fit model with Laplace smoothing
lyrics.nb <-
naiveBayes(lyrics.train.bin,lyrics.lab[lyrics.train],
laplace=1)
# make predictions on test sample
> lyrics.nb.pred <- predict(lyrics.nb,lyrics.test.bin)
# make confusion matrix
> table(lyrics.nb.pred,lyrics.lab[-lyrics.train])

lyrics.nb.pred Blues Metal
         Blues     5     0
         Metal     0     5
```

# Conditional Probability Tables

```
# P(blood|Genre)
> lyrics.nb$table$blood

                       blood
lyrics.lab[lyrics.train]      FALSE       TRUE
                   Blues 0.94117647 0.05882353
                   Metal 0.64705882 0.35294118


# P(baby|Genre)
> lyrics.nb$table$baby

                       baby
lyrics.lab[lyrics.train]      FALSE       TRUE
                   Blues 0.35294118 0.64705882
                   Metal 0.94117647 0.05882353
```

# Contributions of terms to score

For a given attribute vector $\vec{x}$ the "score" of class $c$ is:

$$\hat{P}(C = c|\vec{x}) \propto \hat{P}(c) \prod_{i=1}^{M} \hat{P}(x_i|c)$$

We commonly use the log version:

$$\text{score}(c; \vec{x}) = \log \hat{P}(c) + \sum_{i=1}^{M} \log \hat{P}(x_i|c)$$

Relative contribution of $X_i = x_i$ to class $c$ compared to $c'$:

$$\log \hat{P}(x_i|c) - \log \hat{P}(x_i|c') = \log \frac{\hat{P}(x_i|c)}{\hat{P}(x_i|c')}$$

# Relative contribution of terms to Blues compared to Metal

For example, the relative contribution of “blood” to Blues is

$$\log \hat{P}(\text{blood}|\text{Blues}) - \log \hat{P}(\text{blood}|\text{Metal}) = \log \frac{0.05882353}{0.35294118} = -2.57$$

The relative contribution of “baby” to Blues is

$$\log \hat{P}(\text{baby}|\text{Blues}) - \log \hat{P}(\text{baby}|\text{Metal}) = \log \frac{0.64705882}{0.05882353} = 3.46$$