Knowledge Discovery and Data Mining 1 (VO) (707.003)

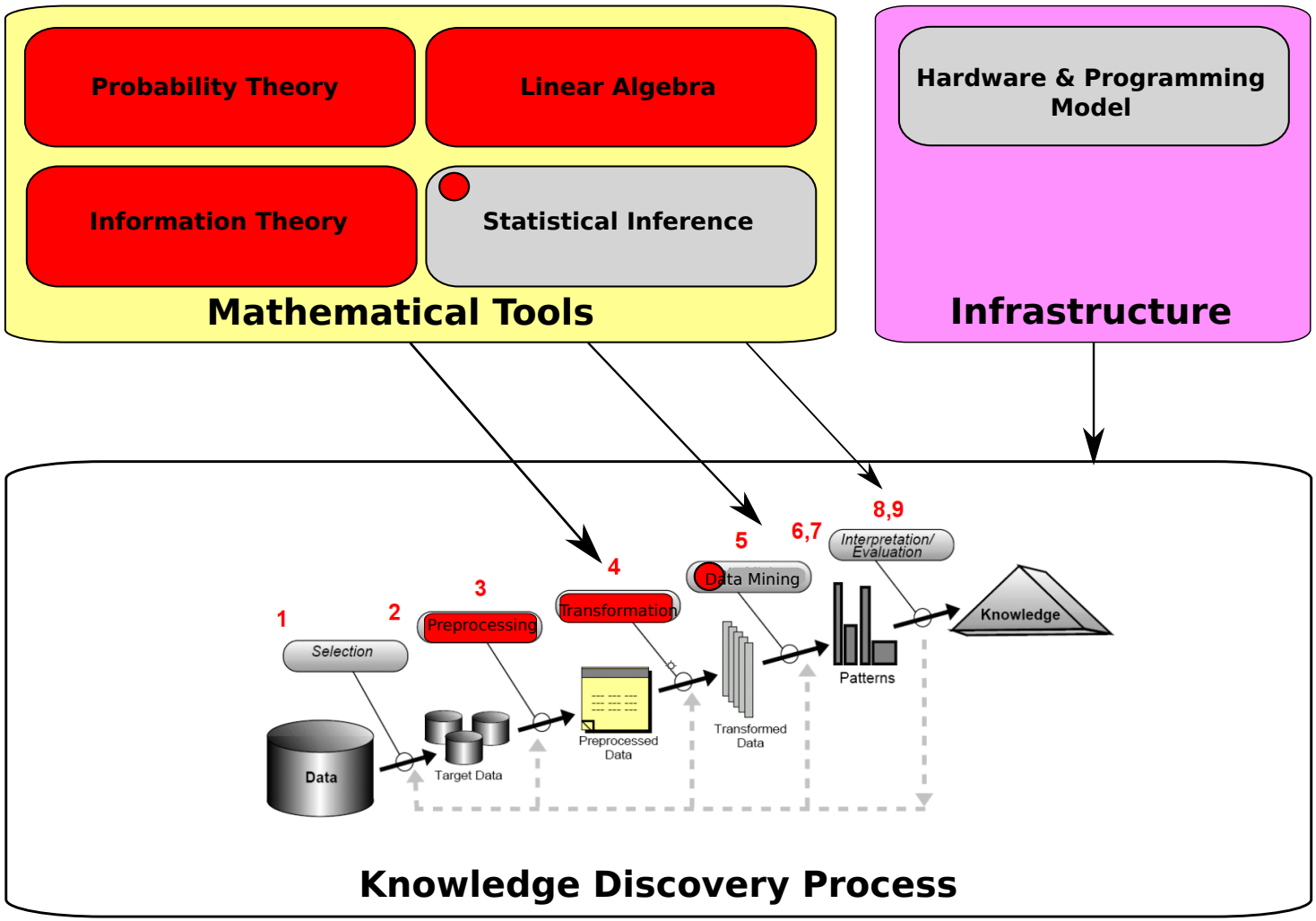
Classification

Denis Helic

KTI, TU Graz

Dec 11, 2014

Big picture: KDDM



Outline

Slides

Slides are partially based on "Machine Learning" course from University of Washington by Pedro Domingos, "Machine Learning" book by Tom Mitchell, and "Introduction to Information Retrieval" by Manning, Raghavan and Schütze

Classification as a Machine Learning task

- Classification problems: classify an example into given set of categories
- Typically we have a labeled set of training examples
- We apply an algorithm to learn from the training set and devise a classification rule
- We apply the rule to classify new examples

Types of learning

- Supervised (inductive) learning: training data includes the desired outputs
- Unsupervised learning: training data does not include the desired outputs
- Semi-supervised learning: training data includes a few desired outputs
- Reinforcement learning: rewards from sequence of actions

Supervised learning

- Given examples of a function $(\mathbf{x}, f(\mathbf{x}))$
- Predict function $f(\mathbf{x})$ for new examples $\mathbf{x}$
- Depending on $f(\mathbf{x})$ we have:
  1. Classification if $f(\mathbf{x})$ is a discrete function
  2. Regression if $f(\mathbf{x})$ is a continuous function
  3. Probability estimation if $f(\mathbf{x})$ is a probability distribution

Classification

- **Given:** Training examples $(\mathbf{x}, f(\mathbf{x}))$ for some unknown *discrete* function f
- **Find:** A good approximation for f
- $\mathbf{x}$ is data, e.g. a vector, text documents, emails, words in text, etc.

Classification: example applications

- **Credit risk assessment:**
  - $\mathbf{x}$: properties of customers and proposed purchase
  - $f(\mathbf{x})$: approve purchase or not
- **Disease diagnosis:**
  - $\mathbf{x}$: properties of patients, lab tests
  - $f(\mathbf{x})$: disease

Classification: example applications

- **Email spam filter:**
  - $\mathbf{x}$: email text (or features extracted from emails)
  - $f(\mathbf{x})$: spam or not spam
- **Text categorization:**
  - $\mathbf{x}$: documents or words (features) from a document
  - $f(\mathbf{x})$: thematic category

Terminology

- **Training example**: an example of the form $(\mathbf{x}, f(\mathbf{x}))$
- **Target function**: the true function f
- **Hypothesis**: a proposed function h believed to be similar to f
- **Classifier**: a discrete-valued function. The possible values $f(\mathbf{x}) \in 1, 2, \ldots, K$ are called **classes** or **labels**
- **Hypothesis space**: a space of all possible hypothesis

Classification approaches

- Logistic regression
- Decision trees
- Support vector machines
- Nearest neighbor algorithms
- Neural networks
- Statistical inference such as Bayesian learning (e.g. text categorization)

Model-based methods

- Statistical inference is based on fitting a probabilistic model of data
- The idea is based on a probabilistic or *generative* model
- Such models assign a probability for observing specific data examples, e.g. observing words in a text document
- Generative models are a powerful method to encode specific assumptions of how unknown parameters interact to create data

Generative models

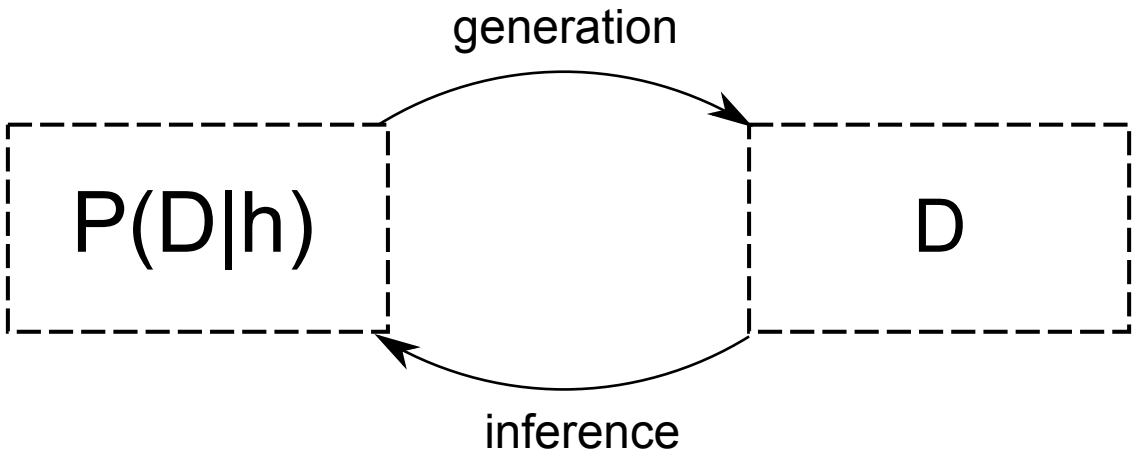
- Generative models have many advantages:
  1. they make our assumptions about the world explicit (rather than encoding them within a procedure or algorithm)
  2. their parameters can be directly interpreted with respect to certain hypothesis about data
  3. comparison of different parameterizations is based on likelihood, which is a fundamental principle in probability theory and statistics
  4. they make probabilistic statements about observation (lack-of) of specific data examples
  5. they allow for prediction of future features based on the past observations

Generative models

- The main disadvantage is that fitting of the models can be more complicated than an algorithmic approach
- How does a generative network model work?
- It defines a conditional probability distribution over data given a hypothesis $P(D|h)$
- Given h we generate data from the conditional distribution $P(D|h)$

Inference

- (Statistical) *inference* is the reverse of the generation process
- We are given some data D, e.g. a collection of documents
- We want to estimate the model, or more precisely the parameters of the hypothesis h that are most likely to have generated the data



Bayesian learning

- Bayesian learning is a method for model selection
- I.e. we might have competing models, or competing hypotheses h
- Which of these hypotheses is the most likely one?
- For example, we have competing classes for a text document
- Which is the most probable class?

Bayesian learning

- Bayesian learning can calculate explicit probabilities for hypotheses (e.g. classes)
- Each observed training example can incrementally decrease or increase the estimated probability (updating)
- Prior knowledge can be combined with observed data
- Prior knowledge consists of a prior probability for each candidate hypothesis and $P(D|h)$
- New instances can be classified by weighting the predictions of multiple hypotheses

Bayes rule

Theorem

Suppose $P(A) > 0$ *and* $P(B) > 0$. *Then,*

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)}$$

- Proof left for exercise ;)

Application in model selection

- We are given some hypothesis space H and the observed data D, and some **prior** probabilities of various hypotheses
- We are interested in determining the best hypothesis from H
- The best means the most probable given the data plus the prior
- In other words we are interested in **posterior** probabilities of various hypotheses given the data and their prior probabilities

Application in model selection

Bayes Theorem

For some given joint probability distribution $P(D, h)$ we have the prior probability of h given as $P(h)$, the probability of observing the data $P(D)$ regardless of which hypothesis holds, and the conditional probability of data given a hypothesis $P(D|h)$. The posterior probability of a hypothesis h is given by:

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)} \tag{1}$$

Application in model selection

- $P(h|D)$ increases with $P(h)$ and with $P(D|h)$
- Also, $P(h|D)$ decreases as $P(D)$ increases because the more probable it is that D will be observed independent of h, the less evidence D provides in support of h
- In many learning scenarios the learner considers some set of candidate hypotheses H and selects the most probable hypothesis from this set
- Such maximally probably hypotheses is called *maximum a posteriori* (MAP) hypothesis

Maximum a posteriori (MAP) hypothesis

MAP

$$
\begin{aligned}
h_{MAP} &\equiv \underset{h \in H}{\operatorname{argmax}} P(h|D) \\
&= \underset{h \in H}{\operatorname{argmax}} \frac{P(D|h)P(h)}{P(D)} \\
&= \underset{h \in H}{\operatorname{argmax}} P(D|h)P(h) \qquad (2)
\end{aligned}
$$

- In the last step we dropped $P(D)$ because it is a constant independent of h

Maximum likelihood (ML) hypothesis

- In some cases we can assume that every hypothesis is equally probable a priori
- The MAP equation simplifies to:

ML

$$h_{ML} = \underset{h \in H}{\operatorname{argmax}} P(D|h) \tag{3}$$

- $P(D|h)$ is the likelihood of the data given a hypothesis
- This is then maximum likelihood hypothesis

MAP: example

Medical diagnosis

Consider a medical diagnosis problem in which there are two alternative hypotheses:

1. The patient has a particular form of cancer
2. The patient does not have cancer.

The available data is from a particular laboratory test with two possible outcomes: P (positive) and N (negative). We have prior knowledge that over the entire population only .008 people have this disease. Furthermore, the test returns a correct positive result in only 98% of the cases in which the disease is actually present and a correct negative test in only 97% of the cases in which the disease is not present. What is h_{MAP}?

MAP: example

- $P(cancer) = 0.008$
- $P(cancer^c) = 0.992$
- $P(P|cancer) = 0.98$
- $P(N|cancer) = 0.02$
- $P(P|cancer^c) = 0.03$
- $P(N|cancer^c) = 0.97$
- $h_{MAP} = ?$

MAP: example

- If the test is positive

$$\begin{aligned} P(P|cancer)P(cancer) &= 0.00784 \\ P(P|cancer^c)P(cancer^c) &= 0.02976 \\ h_{MAP} &= cancer^c \end{aligned}$$

- If the test is negative

$$\begin{aligned} P(N|cancer)P(cancer) &= 0.00016 \\ P(N|cancer^c)P(cancer^c) &= 0.96224 \\ h_{MAP} &= cancer^c \end{aligned}$$

Bayes classifier

- We considered the question: "What is the most probable hypothesis given the training data?"
- We can also ask: "What is the most probable classification of the new instance given the training data?"
- In other words we can apply Bayes theorem for the classification problem
- The most well-known example of such an approach is the Naive Bayes Classifier

Naive Bayes Classifier

- The Naive Bayes approach is based on the MAP approach applied to the set of possible classifications of new data instances
- Given a data instance as a tuple of values $\mathbf{x} = \langle x_1, x_2, \ldots, x_n \rangle$ and a finite set of classes C c_{MAP} is given by:

$$c_{MAP} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j | x_1, x_2, \ldots, x_n) \tag{4}$$

$$\begin{aligned} c_{MAP} &= \underset{c_j \in C}{\operatorname{argmax}} \frac{P(x_1, x_2, \ldots, x_n | c_j) P(c_j)}{P(x_1, x_2, \ldots, x_n} \\ &= \underset{c_j \in C}{\operatorname{argmax}} P(x_1, x_2, \ldots, x_n | c_j) P(c_j) \end{aligned} \tag{5}$$

Naive Bayes Classifier

- We should estimate the two terms from the previous equation based on the training data
- $P(c_j)$ is easy: we just count the frequency with which each class occurs in the training data
- Estimating $P(x_1, x_2, \ldots, x_n|c_j)$ is difficult unless we have very very large dataset
- The number of those terms is equal to the number of possible instances times the number of classes
- We need to see every instance in the instance space to obtain reliable estimates

Naive Bayes Classifier

- Using the chain rule we can rewrite $P(x_1, x_2, \ldots, x_n|c_j)$ as:

$$P(x_1, \ldots, x_n|c_j) = P(x_1|c_j)P(x_2|x_1, c_j)P(x_3|x_1, x_2, c_j)\ldots P(x_n|x_1, \ldots, x_{n-1}, c_j) \quad (6)$$

Naive Bayes Classifier

- The naive Bayes classifier is based on the simplifying assumption that the attribute values are conditionally independent given the class

$$P(x, y|c) = P(x|c)P(y|c)$$

$$P(x|y, c) = P(x|c), \forall x, y, c$$

$$\begin{aligned} P(x_1, \ldots, x_n|c_j) &= P(x_1|c_j)P(x_2|c_j)P(x_3|c_j)\ldots P(x_n|c_j) \\ &= \prod_i P(x_i|c_j) \end{aligned} \tag{7}$$

Naive Bayes Classifier

- The c_{MAP} is then given by:

$$c_{MAP} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_i P(x_i|c_j) \tag{8}$$

- Estimating $P(x_i|c_j)$ from the training data is much easier than estimation of $P(x_1, x_2, \ldots, x_n|c_j)$
- The number of parameters to estimate is the number of distinct attribute values times the number of distinct classes
- Estimation of a single parameter is typically easy: e.g. based on frequency of their occurrence in the training data

Naive Bayes Classifier: Example

- *PlayTennis* problem from the book "Machine Learning" by Tim Mitchell
- Can we play tennis on a given day with given attributes
- It is a classification problem with two classes: "yes" and "no"
- The classification is based on the values of attributes such as temperature, humidity, etc.
- We have 14 training examples

Naive Bayes Classifier: example

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	sunny	hot	high	weak	no
D2	sunny	hot	high	strong	no
D3	overcast	hot	high	weak	yes
D4	rain	mild	high	weak	yes
D5	rain	cool	normal	weak	yes
D6	rain	cool	normal	strong	no
D7	overcast	cool	normal	strong	yes
D8	sunny	mild	high	weak	no
D9	sunny	cool	normal	weak	yes
D10	rain	mild	normal	weak	yes
D11	sunny	mild	normal	strong	yes
D12	overcast	mild	high	strong	yes
D13	overcast	hot	normal	weak	yes
D14	rain	mild	high	strong	no

Naive Bayes Classifier: Example

- The task is to classify the new instance: $\langle$sunny, cool, high, strong$\rangle$
- The class is then given by:

$$
\begin{aligned}
c_{MAP} &= \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_i P(x_i|c_j) \\
&= \underset{c_j \in C}{\operatorname{argmax}} P(c_j) P(sunny|c_j) P(cool|c_j) P(high|c_j) P(strong|c_j)
\end{aligned}
$$

Naive Bayes Classifier: example

PlayTennis \ Outlook	sunny	overcast	rain	n
yes	2	4	3	9
no	3	0	2	5

P \ Outlook	sunny	overcast	rain
$P(Outlook\|yes)$	$2/9$	$4/9$	$3/9$
$P(Outlook\|no)$	$3/5$	0	$2/5$

Table : Frequencies and conditional probability of "Outlook" feature

Naive Bayes Classifier: example

PlayTennis \ Temperature	hot	mild	cool	n
yes	2	4	3	9
no	2	2	1	5

P \ Temperature	hot	mild	cool
$P(Temperature \mid yes)$	$2/9$	$4/9$	$3/9$
$P(Temperature \mid no)$	$2/5$	$2/5$	$1/5$

Table : Frequencies and conditional probability of "Temperature" feature

Naive Bayes Classifier: example

PlayTennis \ Humidity	high	normal	n
yes	3	6	9
no	4	1	5

P \ Humidity	high	normal
$P(Humidity \mid yes)$	$3/9$	$6/9$
$P(Humidity \mid no)$	$4/5$	$1/5$

Table : Frequencies and conditional probability of "Humidity" feature

Naive Bayes Classifier: example

PlayTennis \ Wind	weak	strong	n
yes	6	3	9
no	2	3	5

P \ Wind	weak	strong
$P(Wind\|yes)$	$6/9$	$3/9$
$p(Wind\|no)$	$2/5$	$3/5$

Table : Frequencies and conditional probability of "Wind" feature

Naive Bayes Classifier: Example

- We calculate unnormalized a posteriori probabilities for the classes:

$$P(yes)P(sunny|yes)P(cool|yes)P(high|yes)P(strong|yes) = 0.00529$$
$$P(no)P(sunny|no)P(cool|no)P(high|no)P(strong|no) = 0.02057$$

- $c_{MAP} = no$

Naive Bayes Classifier: Example

Remarks

- These are not the probabilities
- To obtain conditional probabilities we need to normalize with the sum over classes
- $\frac{0.020507}{0.00529+0.02057} = 0.7949$
- In a real example the numbers get very small: danger of **underflow**
- Use logarithms!

Naive Bayes Classifier: Example

- The task is to classify the new instance: $\langle$overcast, cool, high, strong$\rangle$

$$P(yes)P(overcast|yes)P(cool|yes)P(high|yes)P(strong|yes) = 0.01058$$
$$P(no)P(overcast|no)P(cool|no)P(high|no)P(strong|no) = 0$$

- $P(overcast|no) = 0$
- $c_{MAP} = yes$ for any new instance with overcast as Outlook

Estimating probabilities

- We have estimated probabilities by the fraction of times the event is observed to occur over the total number of opportunities
- $P = \frac{n_c}{n}$
- The first difficulty: $\frac{n_c}{n}$ is a biased underestimate of the real probability
- Second, when the estimate is 0 a posteriori probability for that class is always 0
- Smoothing! (as discussed in feature extraction lecture)

Estimating probabilities

- We introduce fake counts
- One possibility: m-estimate of probability

$$P = \frac{n_c + mp}{n + m} \tag{9}$$

- p is prior estimate of the probability we wish to determine
- m is a constant called equivalent sample size

Estimating probabilities

- In absence of any other information we will assume uniform priors $p = \frac{1}{k}$, with k being the number of distinct values of a feature
- E.g. if a feature has two possible values $p = 0.5$
- Using this prior we produce m fake counts
- E.g. if a feature has two possible values we can produce two fake counts:

$$P = \frac{n_c + 1}{n + 2} \tag{10}$$

Classifying text

- Text classification is a standard area of classical information retrieval
- A query (e.g. "Multicore CPU") against a document collection divides the collection into two classes
- Documents about "Multicore CPU"
- Documents not about "Multicore CPU"
- A typical two-class classification problem

Classifying text

- In a general case, a class is a *subject*, a *topic* or a *category*
- Application examples of text classification include:
  1. Automatic detection of spam documents or other types of offending content
  2. Sentiment detection (positive or negative product reviews)
  3. Personal email sorting
  4. Topical or vertical search
- There are several approaches for text classification but again we concentrate on statistical text classification, in particular on Naive Bayes

Formalizing the text classification problem

- We are given a description $d \in \mathbb{X}$ of a document and a fixed set of classes $\mathbb{C} = \{c_1, c_2, \ldots, c_m\}$
- $\mathbb{X}$ is a document space and is typically some high dimensional space
- The classes are in most cases human defined
- We are given a training set $\mathbb{D}$ of labeled documents $\langle d, c \rangle \in \mathbb{X} \times \mathbb{C}$
- For example: $\langle$ Beijing joins the World Trade Organization, China $\rangle$

Formalizing the text classification problem

- Using a learning algorithm we learn a classification function γ

$$\gamma : \mathbb{X} \rightarrow \mathbb{C} \tag{11}$$

- We apply supervised learning method Γ which learns γ: $\Gamma(\mathbb{D}) = \gamma$
- We investigate now in more details Naive Bayes learning method

Naive Bayes text classification

- The MAP approach applied to the set of possible classifications of a new document d

$$c_{MAP} = \underset{c_j \in \mathbb{C}}{\operatorname{argmax}} P(c_j|d) \tag{12}$$

$$\begin{aligned} c_{MAP} &= \underset{c_j \in \mathbb{C}}{\operatorname{argmax}} \frac{P(d|c_j)P(c_j)}{P(d} \\ &= \underset{c_j \in C}{\operatorname{argmax}} P(d|c_j)P(c_j) \end{aligned} \tag{13}$$

Naive Bayes text classification

- We can interpret the last equation as a description of the **generative** process
- To generate a document we first select class c with probability $P(c)$
- In the second step we generate a document given c, corresponding to the conditional probability distribution $P(d|c)$
- How does this conditional probability distribution looks like?
- First we need to make certain simplifying assumptions

Naive Bayes text classification

- Similarly, to the naive Bayes classifier we will base our models on the simplifying assumption that the attribute values are independent, or conditionally independent given the class

$$P(t_1 t_2 t_3 t_4) = P(t_1)P(t_2)P(t_3)P(t_4)$$

- This is unigram language model
- Under this model the order of words is irrelevant: "bag of words"
- A generative model will have the same probability for any "bag of words" regardless their ordering

Naive Bayes text classification

- Assuming "bag of words" model we still need to select a particular conditional distribution over bags
- Two most common approaches are:
  1. Multinomial distribution
  2. Bernoulli distribution

Multinomial random variable

- The model is based on multinomial distribution, which is a generalization of the binomial distribution
- Each of n trials leads to a success in one of k categories, with each category having a fixed success probability
- It holds $\sum_{i=1}^{k} p_i = 1$
- A multinomial r.v. is given by the vector $X = (X_1, X_2, \ldots, X_k)$, where X_i is the number of successes in the ith category
- It holds $\sum_{i=1}^{k} X_i = n$

Multinomial random variable

PMF

$$p(x1, \ldots, x_k; n, p_1, \ldots, p_k) = \begin{cases} \frac{n!}{x_1! \ldots x_n!} p_1^{x_1} \ldots p_k^{x_k} \text{ if } \sum_{i=1}^{k} x_i = n \\ 0, \text{ otherwise} \end{cases}$$

- With $\sum_{i=1}^{k} p_i = 1$ and x_i non-negative, $\forall i$

Multinomial generative model

- In this model we assume that the documents are generated from a multinomial distribution, i.e. the number of occurrences of terms in document is a multinomial r.v.

$$P(d|c) = \begin{cases} \frac{L_d!}{tf_{t_1,d}!\ldots tf_{t_n,d}!} p(t_1|c)^{tf_{t_1,d}} \ldots p(t_M|C)^{tf_{t_M,d}} \text{ if } \sum_{i=1}^{M} tf_{t_i,d} = L_d \\ 0, \text{ otherwise} \end{cases}$$

- $tf_{t_i,d}$ is the term frequency of the term t_i in document d
- L_d is the length of the document, i.e. $L_d = \sum_{i=1}^{M} tf_{t_i,d}$
- M is the size of the document vocabulary

Inference of parameters in the multinomial model

- Likelihood of all documents belonging to a class would be the product of multinomial probabilities
- Simple calculus (setting of derivative of log-likelihood to 0) give as the maximum likelihood estimate for $p(c)$ and $p(t_i|c)$

$$\begin{aligned} p(c) &= \frac{n_c}{n} \\ p(t_i|c) &= \frac{tf_{t_i,c}}{\sum_{i=1}^{M} tf_{t_i,c}} \end{aligned}$$

- n is the total number of documents, n_c is the number of documents in the class c
- $tf_{t_i,c}$ is the term frequency of term t_i in all documents belonging to the class c

Inference of parameters in the multinomial model

- As before because of zero counts we need to apply smoothing, e.g. Laplace smoothing

$$p(t_i|c) = \frac{tf_{t_i,c} + 1}{\sum_{i=1}^{M}(tf_{t_i,c} + 1)}$$

MAP in the multinomial model

- With the estimated parameters and ignoring the irrelevant constants we obtain

$$\begin{aligned} c_{MAP} &= \underset{c_j \in \mathbb{C}}{\operatorname{argmax}}\, P(d|c_j)P(c_j) \\ &= \underset{c_j \in \mathbb{C}}{\operatorname{argmax}}\, P(c_j) \prod_{i=1}^{L_d} p(t_i|c) \end{aligned} \tag{14}$$

- Interpretation of $p(t_i, c)$: how much evidence t_i contributes that c is the correct class
- Implementation with logarithms! Underflow!

NB with multinomial model: Example

Document	Class
Chinese Beijing Chinese	China
Chinese Chinese Shanghai	China
Chinese Macao	China
Tokyo Japan Chinese	Japan

- What is the class of $d = \langle$ Chinese Chinese Chinese Tokyo Japan $\rangle$

NB with multinomial model: Example

Class \ Term	Chinese	Bejing	Shanghai	Macao	Tokyo	Japan
China	5	1	1	1	0	0
Japan	1	0	0	0	1	1

P \ Term	Chinese	Bejing	Shanghai	Macao	Tokyo	Japan
$P(t\|China)$	$6/14$	$2/14$	$2/14$	$2/14$	$1/14$	$1/14$
$p(t\|Japan)$	$2/9$	$1/9$	$1/9$	$1/9$	$2/9$	$2/9$

Table : Frequencies and conditional probability of terms (Laplace smoothing)

NB with multinomial model: Example

- The task is to classify the new instance: $d = \langle$ Chinese Chinese Chinese Tokyo Japan $\rangle$

$$P(China)P(Chinese|China)^3P(Tokyo|China)P(Japan|China) = 0.0003$$
$$P(Japan)P(Chinese|Japan)^3P(Tokyo|Japan)P(Japan|Japan) = 0.0001$$

- $c_{MAP} = China$
- Three occurrences of positive indicator outweigh two occurrences of the negative indicator

NB with multinomial model

TRAINMULTINOMIALNB($\mathbb{C}, \mathbb{D}$)

1 $V \leftarrow$ EXTRACTVOCABULARY($\mathbb{D}$)
2 $N \leftarrow$ COUNTDOCS($\mathbb{D}$)
3 **for each** $c \in \mathbb{C}$
4 **do** $N_c \leftarrow$ COUNTDOCSINCLASS($\mathbb{D}, c$)
5 $prior[c] \leftarrow N_c / N$
6 $text_c \leftarrow$ CONCATENATETEXTOFALLDOCSINCLASS($\mathbb{D}, c$)
7 **for each** $t \in V$
8 **do** $T_{ct} \leftarrow$ COUNTTOKENSOFTERM($text_c, t$)
9 **for each** $t \in V$
10 **do** $condprob[t][c] \leftarrow \frac{T_{ct}+1}{\sum_{t'}(T_{ct'}+1)}$
11 **return** $V, prior, condprob$

NB with multinomial model

APPLYMULTINOMIALNB($\mathbb{C}, V, prior, condprob, d$)
1 $W \leftarrow$ EXTRACTTOKENSFROMDOC(V, d)
2 **for each** $c \in \mathbb{C}$
3 **do** $score[c] \leftarrow \log prior[c]$
4 **for each** $t \in W$
5 **do** $score[c] \mathrel{+}= \log condprob[t][c]$
6 **return** $\arg\max_{c \in \mathbb{C}} score[c]$

NB with multinomial model: Complexity

- Estimation of parameters has $O(|\mathbb{C}||V|)$
- Product of the number of classes and the size of the vocabulary
- This is linear in the size of the vocabulary
- Preprocessing is also linear
- Applying is also linear in the size of the document
- All together linear in the time it takes to scan the data: very efficient!

Bernoulli generative model

- In this model we assume that the documents are generated from a multivariate Bernoulli distribution
- The model generates an indicator for each term from the vocabulary
- 1 indicates the presence of the term in a document
- 0 indicates the absence of the term in a document
- Different generation of the documents leads to different parameter estimation and different classification rules

Bernoulli generative model

- In particular, for each term in a vocabulary we perform a Bernoulli trial and its result decides if the term is present in a document or absent

$$P(d|c) = \prod_{t_i \in d} p(t_i|c) \prod_{t_i \notin d} (1 - p(t_i|c))$$

Inference of parameters in the Bernoulli model

- We apply MLE again to get the estimates for $p(c)$ and $p(t_i|c)$

$$
\begin{aligned}
p(c) &= \frac{n_c}{n} \\
p(t_i|c) &= \frac{\sum_{i,j} e_{t_i,d_j}}{n_c}
\end{aligned}
$$

- n is the total number of documents, n_c is the number of documents in the class c
- e_{t_i,d_j} is the indicator of presence of term t_i in document d_j belonging to the class c

Inference of parameters in the multinomial model

- As before because of zero counts we need to apply smoothing, e.g. m-estimate smoothing with $p = 0.5$ and $m = 2$

$$p(t_i|c) = \frac{\sum_{i,j} e_{t_i,d_j} + 1}{n_c + 2}$$

MAP in the Bernoulli model

- With the estimated parameters and ignoring the irrelevant constants we obtain

$$\begin{aligned} c_{MAP} &= \underset{c_j \in \mathbb{C}}{\operatorname{argmax}} P(d|c_j)P(c_j) \\ &= \underset{c_j \in \mathbb{C}}{\operatorname{argmax}} P(c_j) \prod_{t_i \in d} p(t_i|c) \prod_{t_i \notin d} (1 - p(t_i|c)) \end{aligned} \tag{15}$$

- Interpretation of $p(t_i, c)$ and $(1 - p(t_i|c))$: how much evidence the presence or absence of t_i contributes that c is the correct class
- Implementation with logarithms! Underflow!

NB with Bernoulli model: Example

Document	Class
Chinese Beijing Chinese	China
Chinese Chinese Shanghai	China
Chinese Macao	China
Tokyo Japan Chinese	Japan

- What is the class of $d = \langle$ Chinese Chinese Chinese Tokyo Japan $\rangle$

NB with Bernoulli model: Example

Class \ Term	Chinese	Bejing	Shanghai	Macao	Tokyo	Japan
China	3	1	1	1	0	0
Japan	1	0	0	0	1	1

P \ Term	Chinese	Bejing	Shanghai	Macao	Tokyo	Japan
$P(t\|China)$	$4/5$	$2/5$	$2/5$	$2/5$	$1/5$	$1/5$
$p(t\|Japan)$	$2/3$	$1/3$	$1/3$	$1/3$	$2/3$	$2/3$

Table : Frequencies and conditional probability of terms (m-estimate smoothing)

NB with Bernoulli model: Example

- The task is to classify the new instance: $d = \langle$ Chinese Chinese Chinese Tokyo Japan $\rangle$

$$\begin{aligned} P(China)P(Chinese|China)P(Tokyo|China)P(Japan|China) &\times \\ (1 - P(Bejing|China))(1 - P(Shanghai|China))(1 - P(Macao|China)) &= 0.00518 \\ P(Japan)P(Chinese|Japan)P(Tokyo|Japan)P(Japan|Japan) &\times \\ (1 - P(Bejing|Japan))(1 - P(Shanghai|Japan))(1 - P(Macao|Japan)) &= 0.02194 \end{aligned}$$

- $c_{MAP} = Japan$
- "Japan" and "Tokyo" are good indicators for "Japan" and since we do not count occurrences they outweigh "Chinese" indicator

NB with Bernoulli model

TRAINBERNOULLINB($\mathbb{C}, \mathbb{D}$)
1 $V \leftarrow$ EXTRACTVOCABULARY($\mathbb{D}$)
2 $N \leftarrow$ COUNTDOCS($\mathbb{D}$)
3 **for each** $c \in \mathbb{C}$
4 **do** $N_c \leftarrow$ COUNTDOCSINCLASS($\mathbb{D}, c$)
5 $prior[c] \leftarrow N_c / N$
6 **for each** $t \in V$
7 **do** $N_{ct} \leftarrow$ COUNTDOCSINCLASSCONTAININGTERM($\mathbb{D}, c, t$)
8 $condprob[t][c] \leftarrow (N_{ct} + 1)/(N_c + 2)$
9 **return** $V, prior, condprob$

NB with Bernoulli model

APPLYBERNOULLINB($\mathbb{C}, V, prior, condprob, d$)
1 $V_d \leftarrow$ EXTRACTTERMSFROMDOC(V, d)
2 **for each** $c \in \mathbb{C}$
3 **do** $score[c] \leftarrow \log prior[c]$
4 **for each** $t \in V$
5 **do if** $t \in V_d$
6 **then** $score[c] \mathrel{+}= \log condprob[t][c]$
7 **else** $score[c] \mathrel{+}= \log(1 - condprob[t][c])$
8 **return** $\arg\max_{c \in \mathbb{C}} score[c]$

Multinomial vs. Bernoulli model

- The Bernoulli model estimates $p(t_i|c)$ as the fraction of documents of class c that contains term t
- In contrast, the multinomial model estimates $p(t_i|c)$ as the term frequency of term t in a document of a class c relative to the term frequency of term t in all documents of class c
- When classifying a document the Bernoulli model uses binary occurrence information, and the multinomial model keeps track of multiple occurrences

Multinomial vs. Bernoulli model

- As a result the Bernoulli model tends to make more errors when classifying long documents
- E.g. it might assign a complete book to "China" class because of a single occurrence of the term "Chinese"
- Non-occurring terms do not affect classification decision in the multinomial model, whereas in the Bernoulli model they do
- Bernoulli works better with fewer features
- Complexity is same and both models work very good and accurate in practice

NB models

- The independence assumption is oversimplified
- "Bag of words" model ignores positions of the words and their context
- E.g. in "Hong Kong" are not independent of each other nor are their positions
- Bernoulli model does not even take into account the term frequencies
- This leads to very bad estimates of true document probabilities

NB models

- However, NB classification decisions are surprisingly good
- Suppose the following true document probabilities $P(c_1|d) = 0.6$ and $P(c_2|d) = 0.4$
- Assume that d contains many positive indicators for c_1 and many negative indicators for c_2
- Then the NB estimation might be: $\hat{P}(c_1|d) = 0.00099$ and $\hat{P}(c_2|d) = 0.00001$
- This is common: the winning class in NB has usually a much larger probability than the others
- In classification we care only about the winning class

Evaluating classifiers

- We want to objectively compare the results of different classifiers
- What is their error rate?
- What is the accuracy?
- Typically a standard evaluation method from information retrieval is applied
- Precision, recall, F1 measure
- Applied on two-class classifier: c and c^c

Evaluating classifiers

Prediction \ Real class	c	c^c
c	true positive (tp)	false positive (fp)
c^c	false negative (fn)	true negative (tn)

Table : Contingency table

Accuracy

- Accuracy $A = \frac{tp+tn}{tp+fp+fn+tn}$
- What happens with accuracy in a presence of a skewed class distribution?
- E.g. we have one small and one huge class
- $P(cancer) = 0.008$
- $P(cancer^c) = 0.992$

Example

- We always predict: $cancer^c$:

Prediction \ Real class	c	c^c
c	0	0
c^c	8	992

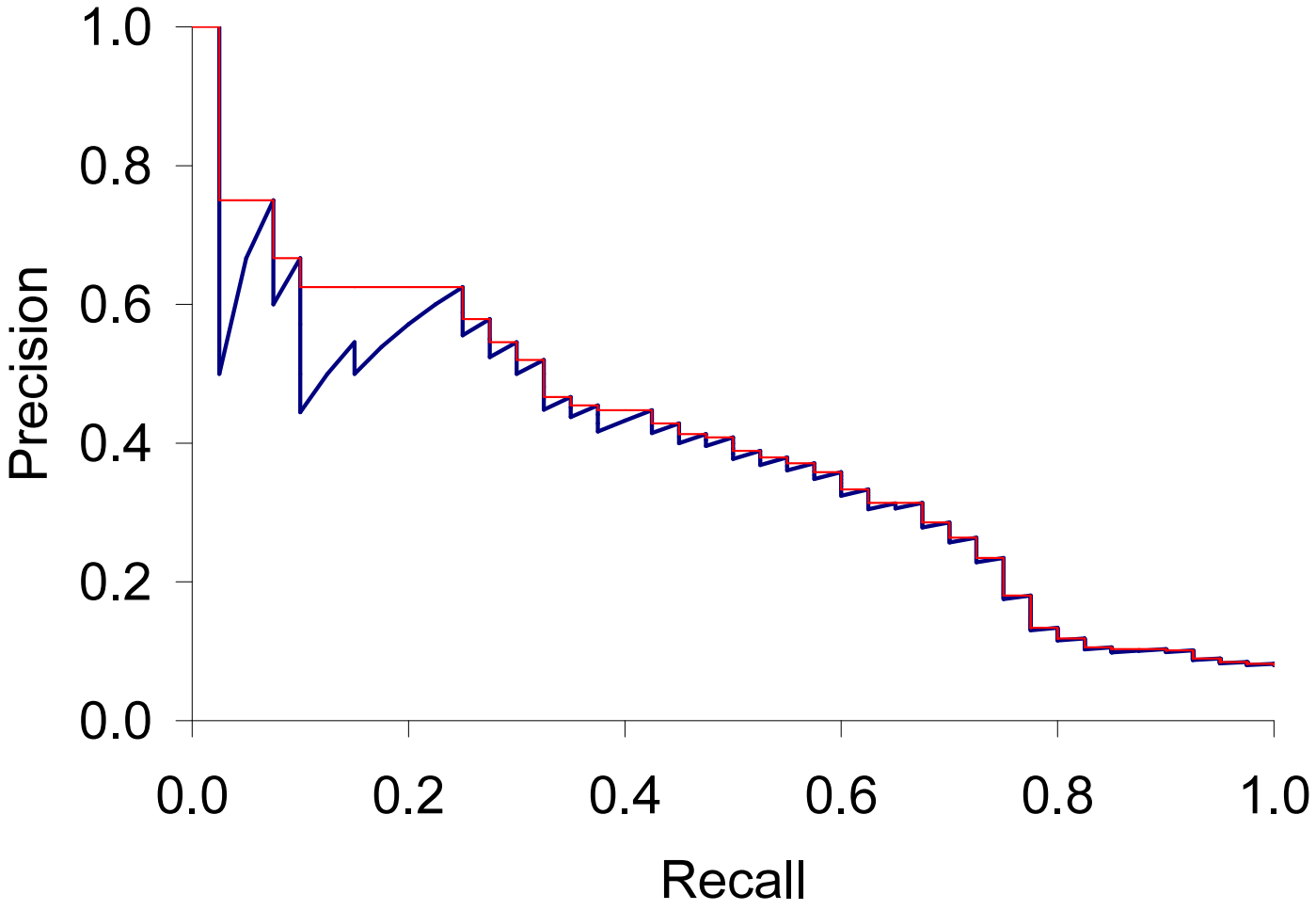
Table : Contingency table: skewed class

- $A = \frac{tp+tn}{tp+fp+fn+tn} = \frac{0+992}{0+0+8+992} = 0.992$

Recall

- But how many did we retrieve when the patient indeed has a cancer?
- We retrieved none!
- Recall $R = \frac{tp}{tp+fn}$
- I.e. recall is 0
- Precision $P = \frac{tp}{tp+fp}$

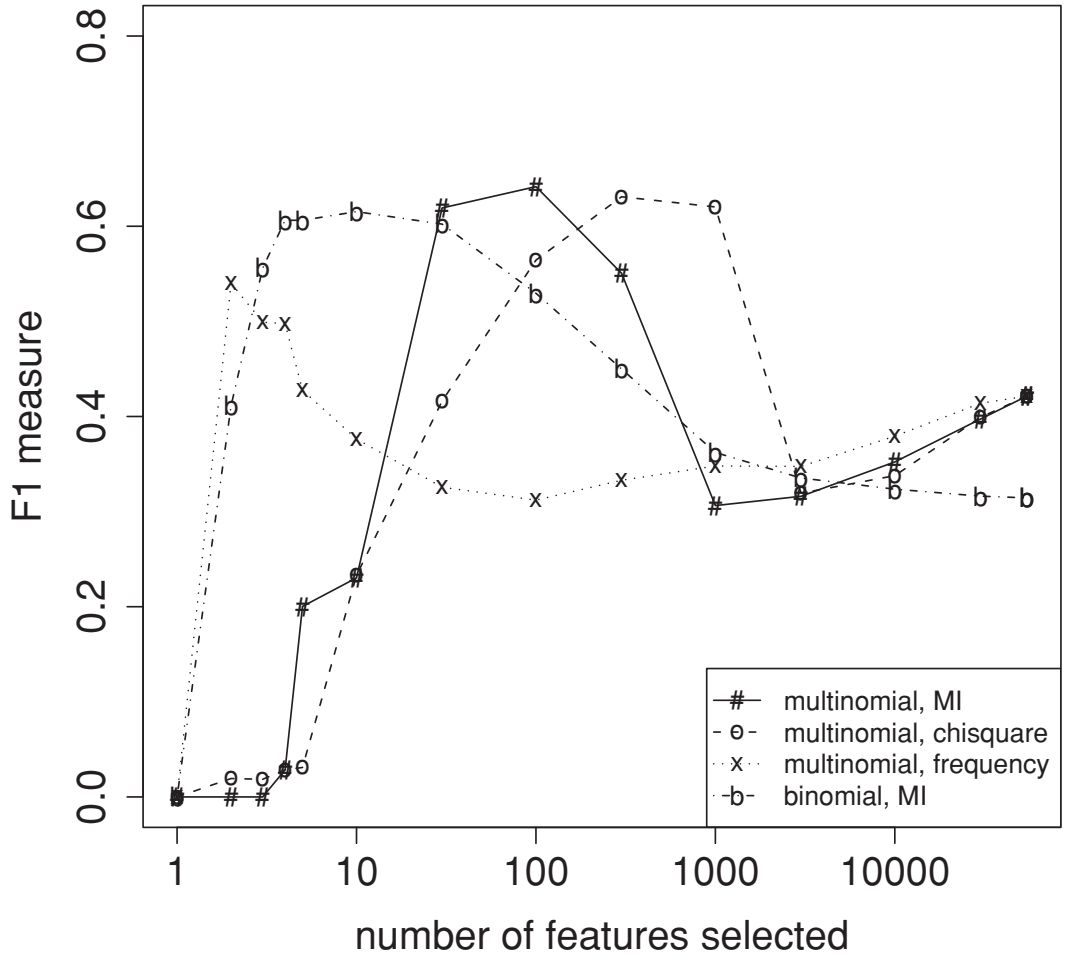
Precision-Recall trade-off



F1 Measure

- Sometimes precision is more important than recall
- E.g. in Web search precision is more important
- In local search recall might be more important
- We can quantify the trade-off by a single measure
- $F1 = \frac{2PR}{P+R}$

NB performance



NB performance

▶ **Table 13.9** Text classification effectiveness numbers on Reuters-21578 for F_1 (in percent). Results from Li and Yang (2003) (a), Joachims (1998) (b: kNN) and Dumais et al. (1998) (b: NB, Rocchio, trees, SVM).

(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

NB performance

- SVM seems to be better but it costs more to train
- kNN is better for some classes
- In theory: SVM better than kNN better than NB
- When data is nice!
- In practice with errors in data, data drifting, etc. very difficult to beat NB