

Probabilistic IR

CS5154/6054

Yizong Cheng

9/22/2022

11 *Probabilistic information retrieval*



CS 276 / LING 286: Information Retrieval and Web Search





Class time & location

Spring quarter 2019

Lecture times: Tues/Thurs, 4:30–5:50pm, April 1 to June 5

Location: Gates B1 (Basement)

Required textbook

Introduction to Information Retrieval, by C. Manning, P. Raghavan, and H. Schütze (Cambridge University Press, 2008).

This book is available from [Amazon](#), the Stanford bookstore, or your favorite book purveyor. You can also download and print chapters for free at the [book website](#). (We'd appreciate any reports of typos or of higher-level problems for the third printing.)

This book will be referred to as **IIR** in the reading assignments listed in the [course schedule](#) section.

Grading & course policies

See the [course policies](#) page for details on grading, late days, and other policies.

Other useful references

- **(MG)** *Managing Gigabytes*, by I. Witten, A. Moffat, and T. Bell.
- **(IRAH)** *Information Retrieval: Algorithms and Heuristics*, by D. Grossman and O. Frieder.
- **(MIR)** *Modern Information Retrieval*, by R. Baeza-Yates and B. Ribeiro-Neto.
- **(FSNLP)** *Foundations of Statistical Natural Language Processing*, by C. Manning and H. Schütze.
- **(SE)** *Search Engines: Information Retrieval in Practice*, by B. Croft, D. Metzler, and T. Strohman.
- **(IRIE)** *Information Retrieval: Implementing and Evaluating Search Engines*, by S. Büttcher, C. Clarke, and G. Cormack.



4/20

due

Week 4

Tues.
4/23

PA1 due

Programming assignment #1 dueTues.
4/23

Guest lecture

*Guest lecture by Joachim Kupke (Principal Software Engineer, Google)***NOTE: attendance required** for on-campus studentsTues.
4/23

PA2 release

Programming assignment #2 releasedThurs.
4/25Lecture
(Chris)**Probabilistic IR: the binary independence model, BM25, BM25F**

- [IIR chapter 6](#)
- [IIR chapter 11](#)

- Videos: "Vector Space Model"
- Slides: [PPT](#) | [PDF/6](#) | [PDF/1](#)

Week 5

Tues.
4/30

PS1 due

Problem set #1 due

Tues.

Lecture

Evaluation methods & NDCG

- [IIR chapter 8](#)

Free 7-Day Trial

Event, Probability, Odds, Model

- An **event** A is a subset of the space of possible **outcomes**.
- The set **complement** of A , \bar{A} , is also an event.
- A **probability** $P(A)$ may be assigned to an event.
- We must have $P(A) + P(\bar{A}) = 1$.
- The **odds** is (11.5) $O(A) = P(A) / P(\bar{A}) = P(A) / (1 - P(A))$.
- Given a query q and a document d , R_{dq} is the event that d is relevant with respect to q . (Sometimes we simply write R instead of R_{dq} .)
- Its probability is $P(R = 1 | d, q)$. It is $1 - P(R = 0 | d, q)$.
- Need a probabilistic **model** to compute $P(R = 1 | d, q)$.

11.1 Review of basic probability theory

RANDOM VARIABLE

We hope that the reader has seen a little basic probability theory previously. We will give a very quick review; some references for further reading appear at the end of the chapter. A variable A represents an event (a subset of the space of possible outcomes). Equivalently, we can represent the subset via a *random variable*, which is a function from outcomes to real numbers; the subset is the domain over which the random variable A has a particular value. Often we will not know with certainty whether an event is true in the world. We can ask the probability of the event $0 \leq P(A) \leq 1$. For two events A and B , the joint event of both events occurring is described by the joint probability $P(A, B)$. The conditional probability $P(A|B)$ expresses the probability of event A given that event B occurred. The fundamental relationship between joint and conditional probabilities is given by the *chain rule*:

CHAIN RULE

$$(11.1) \quad P(A, B) = P(A \cap B) = P(A|B)P(B) = P(B|A)P(A)$$

Without making any assumptions, the probability of a joint event equals the probability of one of the events multiplied by the probability of the other event conditioned on knowing the first event happened.

Conditional Probability, Bayes' Rule

- $P(R = 1 | d, q) = P(d | R = 1, q) P(R = 1 | q) / P(d | q)$ (11.8)
- Chain rule, or the definition of conditional probability:
- $P(R = 1 | d, q) P(d | q) = P(R = 1, d | q) = P(d | R = 1, q) P(R = 1 | q)$
- Think that q is given, we have chain rule
- $P(R = 1 | d) P(d) = P(R = 1, d) = P(d | R = 1) P(R = 1)$
- to relate two events $R = 1$, and d and their intersection $R = 1, d$.
- Odds $O(R | d, q) = P(R = 1 | d, q) / P(R = 0 | d, q) =$
- $O(R | q) P(d | R = 1, q) / P(d | R = 0, q)$. (11.10)

Writing $P(\overline{A})$ for the complement of an event, we similarly have:

$$(11.2) \quad P(\overline{A}, B) = P(B|\overline{A})P(\overline{A})$$

PARTITION RULE Probability theory also has a *partition rule*, which says that if an event B can be divided into an exhaustive set of disjoint subcases, then the probability of B is the sum of the probabilities of the subcases. A special case of this rule gives that:

$$(11.3) \quad P(B) = P(A, B) + P(\overline{A}, B)$$

BAYES' RULE

From these we can derive *Bayes' Rule* for inverting conditional probabilities:

$$(11.4) \quad P(A|B) = \frac{P(B|A)P(A)}{P(B)} = \left[\frac{P(B|A)}{\sum_{X \in \{A, \bar{A}\}} P(B|X)P(X)} \right] P(A)$$

PRIOR PROBABILITY

POSTERIOR
PROBABILITY

This equation can also be thought of as a way of updating probabilities. We start off with an initial estimate of how likely the event A is when we do not have any other information; this is the *prior probability* $P(A)$. Bayes' rule lets us derive a *posterior probability* $P(A|B)$ after having seen the evidence B , based on the *likelihood* of B occurring in the two cases that A does or does not hold.¹

ODDS

Finally, it is often useful to talk about the *odds* of an event, which provide a kind of multiplier for how probabilities change:

$$(11.5) \quad \text{Odds:} \quad O(A) = \frac{P(A)}{P(\bar{A})} = \frac{P(A)}{1 - P(A)}$$

11.2 The Probability Ranking Principle

Using a probabilistic model, the obvious order in which to present documents to the user is to rank documents by their estimated probability of relevance with respect to the information need: $P(R = 1|d, q)$. This is the basis of the *Probability Ranking Principle* (PRP) (van Rijsbergen 1979, 113–114):

PROBABILITY
RANKING PRINCIPLE

“If a reference retrieval system’s response to each request is a ranking of the documents in the collection in order of decreasing probability of relevance to the user who submitted the request, where the probabilities are estimated as accurately as possible on the basis of whatever data have been made available to the system for this purpose, the overall effectiveness of the system to its user will be the best that is obtainable on the basis of those data.”

Optimal Decision Rule, the decision which minimizes the risk of loss, is to simply return documents that are more likely relevant than nonrelevant:

$$(11.6) \quad d \text{ is relevant iff } P(R = 1|d, q) > P(R = 0|d, q)$$

Theorem 11.1. *The PRP is optimal, in the sense that it minimizes the expected loss (also known as the Bayes risk) under 1/0 loss.*

BAYES RISK

Probability Ranking is Classification

- When ranking is by probability $P(R=1 | d, q)$, the Bayes optimal decision rule (11.6) is indeed doing classification.
- Each document is classified as “relevant” if the probability is larger than that of the complement event “nonrelevant”.
- In terms of odds, the document is “relevant” if the odds (ratio of $P(R=1 | d, q)$ and $P(R=0 | d, q)$) is larger than 1.
- Many classification tools also provide this probability and thus the ranking.

11.3 The Binary Independence Model

BINARY INDEPENDENCE MODEL

The *Binary Independence Model* (BIM) we present in this section is the model that has traditionally been used with the PRP. It introduces some simple assumptions, which make estimating the probability function $P(R|d, q)$ practical. Here, “binary” is equivalent to Boolean: documents and queries are both represented as binary term incidence vectors. That is, a document d is represented by the vector $\vec{x} = (x_1, \dots, x_M)$ where $x_t = 1$ if term t is present in document d and $x_t = 0$ if t is not present in d . With this representation, many possible documents have the same vector representation. Similarly, we represent q by the incidence vector \vec{q} (the distinction between q and \vec{q} is less central since commonly q is in the form of a set of words). “Independence” means that terms are modeled as occurring in documents independently. The model recognizes no association between terms. This assumption is far from correct, but it nevertheless often gives satisfactory results in practice; it is the “naive” assumption of Naive Bayes models, discussed further in Section 13.4 (page 265). Indeed, the Binary Independence Model is exactly the same as the multivariate Bernoulli Naive Bayes model presented in Section 13.3 (page 263). In a sense this assumption is equivalent to an assumption of the vector space model, where each term is a dimension that is orthogonal to all other terms.

Binary Vectors as Documents

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth	...
ANTHONY	1	1	0	0	0	1	
BRUTUS	1	1	0	1	0	0	
CAESAR	1	1	0	1	1	1	
CALPURNIA	0	1	0	0	0	0	
CLEOPATRA	1	0	0	0	0	0	
MERCY	1	0	1	1	1	1	
WORSER	1	0	1	1	1	0	
...							

Each document is represented as a **binary vector** $\in \{0, 1\}^{|V|}$.



To make a probabilistic retrieval strategy precise, we need to estimate how terms in documents contribute to relevance, specifically, we wish to know how term frequency, document frequency, document length, and other statistics that we can compute influence judgments about document relevance, and how they can be reasonably combined to estimate the probability of document relevance. We then order documents by decreasing estimated probability of relevance.

$$(11.10) \quad O(R|\vec{x}, \vec{q}) = \frac{P(R = 1|\vec{x}, \vec{q})}{P(R = 0|\vec{x}, \vec{q})} = \frac{\frac{P(R=1|\vec{q})P(\vec{x}|R=1,\vec{q})}{P(\vec{x}|\vec{q})}}{\frac{P(R=0|\vec{q})P(\vec{x}|R=0,\vec{q})}{P(\vec{x}|\vec{q})}} = \frac{P(R = 1|\vec{q})}{P(R = 0|\vec{q})} \cdot \frac{P(\vec{x}|R = 1, \vec{q})}{P(\vec{x}|R = 0, \vec{q})}$$

NAIVE BAYES
ASSUMPTION

we make the *Naive Bayes conditional independence assumption* that the presence or absence of a word in a document is independent of the presence or absence of any other word (given the query):

$$(11.11) \quad \frac{P(\vec{x}|R = 1, \vec{q})}{P(\vec{x}|R = 0, \vec{q})} = \prod_{t=1}^M \frac{P(x_t|R = 1, \vec{q})}{P(x_t|R = 0, \vec{q})}$$

So:

$$(11.12) \quad O(R|\vec{x}, \vec{q}) = O(R|\vec{q}) \cdot \prod_{t=1}^M \frac{P(x_t|R = 1, \vec{q})}{P(x_t|R = 0, \vec{q})}$$

Since each x_t is either 0 or 1, we can separate the terms to give:

$$(11.13) \quad O(R|\vec{x}, \vec{q}) = O(R|\vec{q}) \cdot \prod_{t:x_t=1} \frac{P(x_t = 1|R = 1, \vec{q})}{P(x_t = 1|R = 0, \vec{q})} \cdot \prod_{t:x_t=0} \frac{P(x_t = 0|R = 1, \vec{q})}{P(x_t = 0|R = 0, \vec{q})}$$

Henceforth, let $p_t = P(x_t = 1|R = 1, \vec{q})$ be the probability of a term appearing in a document relevant to the query, and $u_t = P(x_t = 1|R = 0, \vec{q})$ be the probability of a term appearing in a nonrelevant document. These quantities can be visualized in the following contingency table where the columns add to 1:

(11.14)

document		relevant ($R = 1$)	nonrelevant ($R = 0$)
Term present	$x_t = 1$	p_t	u_t
Term absent	$x_t = 0$	$1 - p_t$	$1 - u_t$

BIM

- The binary independence model (BIM, 1976).
- Document d is subset of the vocabulary.
- The naïve Bayes conditional independence assumption.
- $P(d | R = 1, q) = \prod_{t \text{ in } d} P(t \text{ in } d | R = 1, q) \prod_{t \text{ not in } d} P(t \text{ not in } d | R = 1, q)$
- $P(d | R = 0, q) = \prod_{t \text{ in } d} P(t \text{ in } d | R = 0, q) \prod_{t \text{ not in } d} P(t \text{ not in } d | R = 0, q)$
- Define $p_t = P(t \text{ in } d | R = 1, q)$, then $1 - p_t = P(t \text{ not in } d | R = 1, q)$.
- Also $u_t = P(t \text{ in } d | R = 0, q)$ and $1 - u_t = P(t \text{ not in } d | R = 0, q)$.
- $O(R | d, q) = O(R | q) \prod_{t \text{ in } d} p_t / u_t \prod_{t \text{ not in } d} (1 - p_t) / (1 - u_t)$. (11.13)

Let us make an additional simplifying assumption that terms not occurring in the query are equally likely to occur in relevant and nonrelevant documents: that is, if $q_t = 0$ then $p_t = u_t$. (This assumption can be changed, as when doing relevance feedback in Section 11.3.4.) Then we need only consider terms in the products that appear in the query, and so,

$$(11.15) \quad O(R|\vec{q}, \vec{x}) = O(R|\vec{q}) \cdot \prod_{t:x_t=q_t=1} \frac{p_t}{u_t} \cdot \prod_{t:x_t=0, q_t=1} \frac{1-p_t}{1-u_t}$$

The left product is over query terms found in the document and the right product is over query terms not found in the document.

We can manipulate this expression by including the query terms found in the document into the right product, but simultaneously dividing through by them in the left product, so the value is unchanged. Then we have:

$$(11.16) \quad O(R|\vec{q}, \vec{x}) = O(R|\vec{q}) \cdot \prod_{t:x_t=q_t=1} \frac{p_t(1-u_t)}{u_t(1-p_t)} \cdot \prod_{t:q_t=1} \frac{1-p_t}{1-u_t}$$

Simplifying Assumption and Manipulation

- Terms not occurring in the query are equally likely to occur in relevant and nonrelevant documents.
- Query q is also a subset of the vocabulary.
- If t not in q , then $p_t = u_t$ or $P(t \text{ in } d | R = 1, q) = P(t \text{ in } d | R = 0, q)$.
- $O(R|d, q) =$
 - $O(R|q) \prod_{t \text{ in } q \text{ and } d} p_t / u_t \prod_{t \text{ in } q \text{ but not in } d} (1 - p_t) / (1 - u_t). \quad (11.15)$
 - Multiply this with $\prod_{t \text{ in } q \text{ and } d} (1 - p_t) / (1 - u_t) \prod_{t \text{ in } q \text{ and } d} (1 - u_t) / (1 - p_t).$
 - $O(R|d, q) =$
 - $O(R|q) \prod_{t \text{ in } q \text{ and } d} p_t / u_t (1 - u_t) / (1 - p_t) \prod_{t \text{ in } q} (1 - p_t) / (1 - u_t). \quad (11.16)$

$$(11.16) \quad O(R|\vec{q}, \vec{x}) = O(R|\vec{q}) \cdot \prod_{t:x_t=q_t=1} \frac{p_t(1-u_t)}{u_t(1-p_t)} \cdot \prod_{t:q_t=1} \frac{1-p_t}{1-u_t}$$

The left product is still over query terms found in the document, but the right product is now over all query terms. That means that this right product is a constant for a particular query, just like the odds $O(R|\vec{q})$. So the only quantity that needs to be estimated to rank documents for relevance to a query is the left product. We can equally rank documents by the logarithm of this term, since log is a monotonic function. The resulting quantity used for ranking is called the *Retrieval Status Value* (RSV) in this model:

RETRIEVAL STATUS
VALUE

$$(11.17) \quad RSV_d = \log \prod_{t:x_t=q_t=1} \frac{p_t(1-u_t)}{u_t(1-p_t)} = \sum_{t:x_t=q_t=1} \log \frac{p_t(1-u_t)}{u_t(1-p_t)}$$

So everything comes down to computing the *RSV*. Define c_t :

$$(11.18) \quad c_t = \log \frac{p_t(1-u_t)}{u_t(1-p_t)} = \log \frac{p_t}{(1-p_t)} + \log \frac{1-u_t}{u_t}$$

ODDS RATIO

The c_t terms are log odds ratios for the terms in the query. We have the odds of the term appearing if the document is relevant ($p_t/(1-p_t)$) and the odds of the term appearing if the document is nonrelevant ($u_t/(1-u_t)$). The *odds ratio* is the ratio of two such odds, and then we finally take the log of that quantity. The value will be 0 if a term has equal odds of appearing in relevant and nonrelevant documents, and positive if it is more likely to appear in relevant documents. The c_t quantities function as term weights in the model, and the document score for a query is $RSV_d = \sum_{x_t=q_t=1} c_t$. Operationally, we

Retrieval Status Value (RSV)

- $O(R|d, q) =$
- $O(R|q) \prod_{t \text{ in } q \text{ and } d} p_t / u_t (1 - u_t) / (1 - p_t) \prod_{t \text{ in } q} (1 - p_t) / (1 - u_t).$
(11.16)
- For a given query, ranking documents is the same as ranking the red term in (11.16).
- Or, its logarithm $RSV_d = \log \prod_{t \text{ in } q \text{ and } d} p_t / u_t (1 - u_t) / (1 - p_t) =$
- $\sum_{t \text{ in } q \text{ and } d} \log p_t / u_t (1 - u_t) / (1 - p_t) = \sum_{t \text{ in } q \text{ and } d} c_t$ where
- $C_t = \log p_t / (1 - p_t) + \log (1 - u_t) / u_t$ (11.18)

11.3.2 Probability estimates in theory

For each term t , what would these c_t numbers look like for the whole collection? (11.19) gives a contingency table of counts of documents in the collection, where df_t is the number of documents that contain term t :

(11.19)

documents		relevant	nonrelevant	Total
Term present	$x_t = 1$	s	$df_t - s$	df_t
Term absent	$x_t = 0$	$S - s$	$(N - df_t) - (S - s)$	$N - df_t$
Total		S	$N - S$	N

Using this, $p_t = s/S$ and $u_t = (df_t - s)/(N - S)$ and

(11.20)

$$c_t = K(N, df_t, S, s) = \log \frac{s/(S - s)}{(df_t - s)/((N - df_t) - (S - s))}$$

To avoid the possibility of zeroes (such as if every or no relevant document has a particular term) it is fairly standard to add $\frac{1}{2}$ to each of the quantities in the center 4 terms of (11.19), and then to adjust the marginal counts (the totals) accordingly (so, the bottom right cell totals $N + 2$). Then we have:

(11.21)

$$\hat{c}_t = K(N, df_t, S, s) = \log \frac{(s + \frac{1}{2})/(S - s + \frac{1}{2})}{(df_t - s + \frac{1}{2})/(N - df_t - S + s + \frac{1}{2})}$$

11.3.3 Probability estimates in practice

Under the assumption that relevant documents are a very small percentage of the collection, it is plausible to approximate statistics for nonrelevant documents by statistics from the whole collection. Under this assumption, u_t (the probability of term occurrence in nonrelevant documents for a query) is df_t/N and

$$(11.22) \quad \log[(1 - u_t)/u_t] = \log[(N - df_t)/df_t] \approx \log N/df_t$$

In other words, we can provide a theoretical justification for the most frequently used form of idf weighting, which we saw in Section 6.2.1.

The approximation technique in Equation (11.22) cannot easily be extended to relevant documents. The quantity p_t can be estimated in various ways:

1. We can use the frequency of term occurrence in known relevant documents (if we know some). This is the basis of probabilistic approaches to relevance feedback weighting in a feedback loop, discussed in the next subsection.
2. Croft and Harper (1979) proposed using a constant in their combination match model. For instance, we might assume that p_t is constant over all terms x_t in the query and that $p_t = 0.5$. This means that each term has even odds of appearing in a relevant document, and so the p_t and $(1 - p_t)$ factors cancel out in the expression for RSV . Such an estimate is weak, but doesn't disagree violently with our hopes for the search terms appearing in many but not all relevant documents. Combining this method with our earlier approximation for u_t , the document ranking is determined simply by which query terms occur in documents scaled by their idf weighting.

Ad-hoc Retrieval: No Relevance Judgments

- Ad-hoc retrieval: no user-supplied relevance judgments available
- In this case: assume constant $p_t = 0.5$ for all terms x_t in the query
- Each query term is equally likely to occur in a relevant document, and so the p_t and $(1 - p_t)$ factors cancel out in the expression for RSV.
- Weak estimate, but doesn't disagree violently with expectation that query terms appear in many but not all relevant documents.
- Weight c_t in this case: $c_t = \log \frac{p_t}{(1-p_t)} - \log \frac{u_t}{1-u_t} \approx \log N/df_t$
- For short documents (titles or abstracts), this simple version of BIM works well. □

IDF Only in ad-hoc BIM Retrieval (11.30)

- In the simplest version of BIM, the score for document d is just idf weighting of the query terms present in the document:



$$RSV_d = \sum_{t \in q \cap d} \log \frac{N}{df_t}$$



Okapi BM25 Basic Weighting (11.32)

- Improve idf term $[\log N/df]$ by factoring in term frequency and document length.

$$RSV_d = \sum_{t \in q} \log \left[\frac{N}{df_t} \right] \cdot \frac{(k_1 + 1)tf_{td}}{k_1((1 - b) + b \times (L_d/L_{ave})) + tf_{td}}$$

- tf_{td} : term frequency in document d
- L_d (L_{ave}): length of document d (average document length in the whole collection)
- k_1 : tuning parameter controlling scaling of term frequency
- b : tuning parameter controlling the scaling by document length



11.4.3 Okapi BM25: a non-binary model

BM25 WEIGHTS
OKAPI WEIGHTING

The BIM was originally designed for short catalog records and abstracts of fairly consistent length, and it works reasonably in these contexts, but for modern full-text search collections, it seems clear that a model should pay attention to term frequency and document length, as in Chapter 6. The *BM25 weighting scheme*, often called *Okapi weighting*, after the system in which it was first implemented, was developed as a way of building a probabilistic model sensitive to these quantities while not introducing too many additional parameters into the model (Spärck Jones et al. 2000). We will not develop the full theory behind the model here, but just present a series of forms that build up to the standard form now used for document scoring. The simplest score for document d is just idf weighting of the query terms present, as in Equation (11.22):

$$(11.30) \quad RSV_d = \sum_{t \in q} \log \frac{N}{df_t}$$

We can improve on Equation (11.30) by factoring in the frequency of each term and document length:

$$(11.32) \quad RSV_d = \sum_{t \in q} \log \left[\frac{N}{df_t} \right] \cdot \frac{(k_1 + 1)tf_{td}}{k_1((1 - b) + b \times (L_d/L_{ave})) + tf_{td}}$$

Here, tf_{td} is the frequency of term t in document d , and L_d and L_{ave} are the length of document d and the average document length for the whole collection. The variable k_1 is a positive tuning parameter that calibrates the document term frequency scaling. A k_1 value of 0 corresponds to a binary model (no term frequency), and a large value corresponds to using raw term frequency. b is another tuning parameter ($0 \leq b \leq 1$) which determines the scaling by document length: $b = 1$ corresponds to fully scaling the term weight by the document length, while $b = 0$ corresponds to no length normalization.

If the query is long, then we might also use similar weighting for query terms. This is appropriate if the queries are paragraph long information needs, but unnecessary for short queries.


$$(11.33) \quad RSV_d = \sum_{t \in q} \left[\log \frac{N}{df_t} \right] \cdot \frac{(k_1 + 1)tf_{td}}{k_1((1 - b) + b \times (L_d/L_{ave})) + tf_{td}} \cdot \frac{(k_3 + 1)tf_{tq}}{k_3 + tf_{tq}}$$

with tf_{tq} being the frequency of term t in the query q , and k_3 being another positive tuning parameter that this time calibrates term frequency scaling

File Edit View History Bookmarks Tools Help

1.9. Naive Bayes — scikit-learn 1X

https://scikit-learn.org/stable/modules/naive_bayes.html 150%



1.9. Naive Bayes

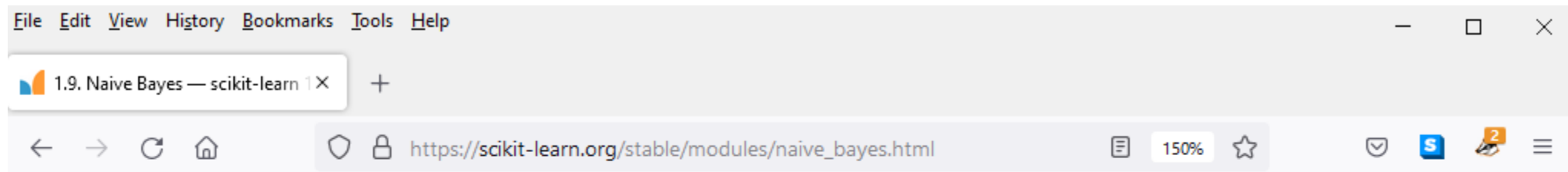
Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of features given the value of the class variable. Bayes' theorem states the following relationship, given class variable y and dependent feature vector x_1 through x_n :

$$P(y \mid x_1, \dots, x_n) = \frac{P(y)P(x_1, \dots, x_n \mid y)}{P(x_1, \dots, x_n)}$$

Using the naive conditional independence assumption that

Toggle Menu

$P(x_1, \dots, x_n \mid y) = P(x_1 \mid y) \dots P(x_n \mid y)$



1.9.4. Bernoulli Naive Bayes

BernoulliNB implements the naive Bayes training and classification algorithms for data that is distributed according to multivariate Bernoulli distributions; i.e., there may be multiple features but each one is assumed to be a binary-valued (Bernoulli, boolean) variable. Therefore, this class requires samples to be represented as binary-valued feature vectors; if handed any other kind of data, a **BernoulliNB** instance may binarize its input (depending on the **binarize** parameter).

The decision rule for Bernoulli naive Bayes is based on

Toggle Menu

$$P(x_i | y) = P(x_i = 1 | y)x_i + (1 - P(x_i = 1 | y))(1 - x_i)$$

File Edit View History Bookmarks Tools Help

sklearn.naive_bayes.BernoulliNB X +

← → ↻ 🏠 🔒 https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.Ber 150% ☆ 📧 📄 ☰

sklearn.naive_bayes.BernoulliNB

```
class sklearn.naive_bayes.BernoulliNB(*, alpha=1.0, binarize=0.0, fit_prior=True,
class_prior=None)
```

[\[source\]](#)

Naive Bayes classifier for multivariate Bernoulli models.

Like MultinomialNB, this classifier is suitable for discrete data. The difference is that while MultinomialNB works with occurrence counts, BernoulliNB is designed for binary/boolean features.

Toggle Menu

the User Guide.

FileEditViewHistoryBookmarksToolsHelp

sklearn.naive_bayes.BernoulliNB X

←→↺🏠🔒https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.Ber150%★📧📌☰

Methods

<code>fit(X, y[, sample_weight])</code>	Fit Naive Bayes classifier according to X, y.
<code>get_params([deep])</code>	Get parameters for this estimator.
<code>partial_fit(X, y[, classes, sample_weight])</code>	Incremental fit on a batch of samples.
<code>predict(X)</code>	Perform classification on an array of test vectors X.
<code>predict_log_proba(X)</code>	Return log-probability estimates for the test vector X.
<code>predict_proba(X)</code>	Return probability estimates for the test vector X.
<code>score(X, y[, sample_weight])</code>	Return the mean accuracy on the given test data and labels.
<code>set_params(**params)</code>	Set the parameters of this estimator.

Toggle Menu

► Table 13.1 Data for parameter estimation examples.

	docID	words in document	in $c = \text{China}$?
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	?

(11.19)

	documents	relevant	nonrelevant	Total
Term present	$x_t = 1$	s	$df_t - s$	df_t
Term absent	$x_t = 0$	$S - s$	$(N - df_t) - (S - s)$	$N - df_t$
	Total	S	$N - S$	N

Using this, $p_t = s/S$ and $u_t = (df_t - s)/(N - S)$ and

$$(11.20) \quad c_t = K(N, df_t, S, s) = \log \frac{s/(S - s)}{(df_t - s)/((N - df_t) - (S - s))}$$

To avoid the possibility of zeroes (such as if every or no relevant document has a particular term) it is fairly standard to add $\frac{1}{2}$ to each of the quantities in the center 4 terms of (11.19), and then to adjust the marginal counts (the totals) accordingly (so, the bottom right cell totals $N + 2$). Then we have:

$$(11.21) \quad \hat{c}_t = K(N, df_t, S, s) = \log \frac{(s + \frac{1}{2})/(S - s + \frac{1}{2})}{(df_t - s + \frac{1}{2})/(N - df_t - S + s + \frac{1}{2})}$$

Adding $\frac{1}{2}$ in this way is a simple form of smoothing. For trials with cat-

```
# IR10A.py CS5154/6054 cheng 2022
# CountVectorizer (binary=True) makes a document-term matrix
# or the X vectors to be used by BernoulliNB along with an Y vector
# Test documents are transformed and the model is applied
# the probability ranking score for relevance or being about China is reported.
# Usage: python IR10A.py
```

```
docs = ["Chinese Beijing Chinese", "Chinese Chinese Shanghai", "Chinese Macao", "Tokyo Japan Chinese"]
R = [1, 1, 1, 0] # R=1: relevant, R=0: nonrelevant
tests = ["Chinese Chinese Chinese Tokyo Japan", "Shanghai Tokyo"]
```

```
cv = CountVectorizer(binary=True)
X = cv.fit_transform(docs).toarray()
voc = cv.get_feature_names()
doclen, voclen = X.shape
T = # your code to transform (but not fit) tests with cv
print(X)
print(T)
```

```
model = BernoulliNB()
model.fit(X, R)
# your code to report the probability  $P(R=1 | q, x)$ , of course q is irrelevant now
```