The Bernoulli Model

CS5154/6054

Yizong Cheng

10/18/2022

13 *Text classification and Naive Bayes*

13.1 The text classification problem

13.3 The Bernoulli model

Machine Learning Algorithms

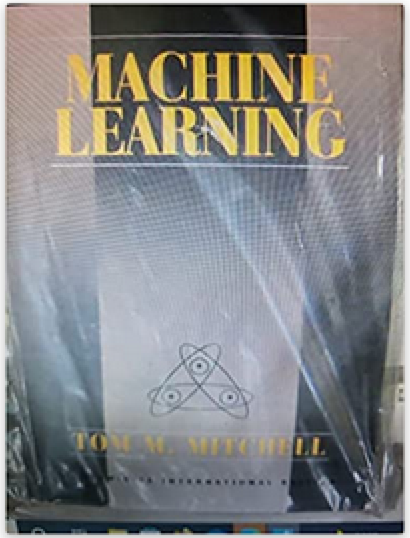
- A computer program is said to learn from experience *E* with respect to some class of tasks *T* and performance measure *P,* if its performance at tasks in *T*, as measured by *P*, improves with experience *E*. -- Tom Mitchell (1997)

File Edit View History Bookmarks Tools Help

Machine Learning (McGraw-Hil

https://www.amazon.com/Learning-McGraw-Hill-International-Editions-Computer/dp/

Look inside



See all 2 images

Follow the Author

Machine Learning (McGraw-Hill International Editions Computer Science Series) Paperback – January 1, 1997

by Tom M. (Tom Michael) Mitchell (Author)

112 ratings

See all formats and editions

Hardcover	Paperback
$286.00	**$85.00**
16 Used from $87.32	14 Used from $39.68
1 New from $286.00	1 New from $85.00

This book covers the field of machine learning, which is the study of algorithms that allow computer programs to automatically improve through experience. The book is intended to support upper level undergraduate and introductory level graduate courses in machine learning.

Buy new: $85.00

FREE Returns

FREE delivery **Sunday, October 23**

Or fastest delivery **Wednesday, October 19**. Order within 6 hrs 26 mins

Select delivery location

In Stock.

Qty: 1

Add to Cart

Buy Now

Secure transaction

Ships from Amazon

Sold by collegebook4u

Seminar

Computer Science Department
College of Engineering and Applied Science

University of Cincinnati
COLLEGE OF ENGINEERING AND APPLIED SCIENCE

SPEAKER: Prof. Tom Mitchell, Carnegie Mellon University
TITLE: Using Machine Learning to Study How Brains Process Natural Language
ROOM: 1210 Carl H Lindner Hall
DATE/TIME: October 20th, 11AM-12:15PM

ABSTRACT: How does the human brain use neural activity to create and represent meanings of words, phrases, sentences and stories? One way to study this question is to give people text to read, while recording their brain activity with fMRI (1 mm spatial resolution) and MEG (1 msec time resolution). We have been doing this, and developing novel machine learning approaches to analyze our data. As a result, we have learned intriguing insights into questions such as "Are the neural encodings of word meaning the same in your brain and mine?", "What sequence of neurally encoded information flows through the brain during the half-second in which the brain comprehends a word?," "How are meanings of multiple words combined when reading sentences, and stories?," and "How does our understanding of the brain align with current AI approaches to natural language processing?" This talk will summarize our machine learning approaches, some of what we have learned, and newer questions we are currently studying.

The Classification Problem

- Given a set of classes, we seek to determine which class(es) a given object belongs to.
- Class: topics, standing query
- Text classification, text categorization, topic spotting, routing, filtering
- Machine learning-based text classification, statistical text classification
  - Needs training set: labeling that may be manual annotation
  - Supervised learning
  - Evaluation with test samples.

Learning a Classifier with a Training Set

13.1 The text classification problem

DOCUMENT SPACE
CLASS

In text classification, we are given a description $d \in \mathbb{X}$ of a document, where $\mathbb{X}$ is the *document space*; and a fixed set of *classes* $\mathbb{C} = \{c_1, c_2, \ldots, c_J\}$. Classes are also called *categories* or *labels*. Typically, the document space $\mathbb{X}$ is some type of high-dimensional space, and the classes are human defined for the needs of an application, as in the examples *China* and *documents that talk about multicore computer chips* above. We are given a *training set* $\mathbb{D}$ of labeled documents $\langle d, c \rangle$,where $\langle d, c \rangle \in \mathbb{X} \times \mathbb{C}$. For example:

TRAINING SET

$$\langle d, c \rangle = \langle \text{Beijing joins the World Trade Organization}, \textit{China} \rangle$$

for the one-sentence document *Beijing joins the World Trade Organization* and the class (or label) *China*.

LEARNING METHOD
CLASSIFIER

Using a *learning method* or *learning algorithm*, we then wish to learn a classifier or *classification function* γ that maps documents to classes:

(13.1)
$$\gamma : \mathbb{X} \rightarrow \mathbb{C}$$

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

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.
- From now on, we may consider “relevant” as a class, and “non-relevant” as its complement.

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 $\vec{x} = (x_1, \ldots, x_M)$ 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.

```
TRAINBERNOULLINB(ℂ, 𝔻)
V ← EXTRACTVOCABULARY(𝔻)
N ← COUNTDOCS(𝔻)
for each c ∈ ℂ
do N_c ← COUNTDOCSINCLASS(𝔻, c)
    prior[c] ← N_c/N
    for each t ∈ V
    do N_ct ← COUNTDOCSINCLASSCONTAININGTERM(𝔻, c, t)
        condprob[t][c] ← (N_ct + 1)/(N_c + 2)
return V, prior, condprob
```

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** $\arg\max_{c \in \mathbb{C}} score[c]$

▶ **Figure 13.3** NB algorithm (Bernoulli model): Training and testing. The add-one smoothing in Line 8 (top) is in analogy to Equation (13.7) with $B = 2$.

▶ **Table 13.1** Data for parameter estimation examples.

	docID	words in document	in $c = China$?
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 13.2: Applying the Bernoulli model to the example in Table 13.1, we have the same estimates for the priors as before: $\hat{P}(c) = 3/4$, $\hat{P}(\overline{c}) = 1/4$. The conditional probabilities are:

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

The denominators are $(3+2)$ and $(1+2)$ because there are three documents in c and one document in $\overline{c}$ and because the constant B in Equation (13.7) is 2 – there are two cases to consider for each term, occurrence and nonoccurrence.

The scores of the test document for the two classes are

$$\begin{aligned}
\hat{P}(c|d_5) &\propto \hat{P}(c) \cdot \hat{P}(\text{Chinese}|c) \cdot \hat{P}(\text{Japan}|c) \cdot \hat{P}(\text{Tokyo}|c) \\
&\quad \cdot (1 - \hat{P}(\text{Beijing}|c)) \cdot (1 - \hat{P}(\text{Shanghai}|c)) \cdot (1 - \hat{P}(\text{Macao}|c)) \\
&= 3/4 \cdot 4/5 \cdot 1/5 \cdot 1/5 \cdot (1{-}2/5) \cdot (1{-}2/5) \cdot (1{-}2/5) \\
&\approx 0.005
\end{aligned}$$

and, analogously,

$$\begin{aligned}
\hat{P}(\overline{c}|d_5) &\propto 1/4 \cdot 2/3 \cdot 2/3 \cdot 2/3 \cdot (1{-}1/3) \cdot (1{-}1/3) \cdot (1{-}1/3) \\
&\approx 0.022
\end{aligned}$$

Thus, the classifier assigns the test document to $\overline{c}$ = *not-China*. When looking only at binary occurrence and not at term frequency, Japan and Tokyo are indicators for $\overline{c}$ $(2/3 > 1/5)$ and the conditional probabilities of Chinese for c and $\overline{c}$ are not different enough (4/5 vs. 2/3) to affect the classification decision.

13.4 Properties of Naive Bayes

To gain a better understanding of the two models and the assumptions they make, let us go back and examine how we derived their classification rules in Chapters 11 and 12. We decide class membership of a document by assigning it to the class with the maximum a posteriori probability (cf. Section 11.3.2, page 226), which we compute as follows:

$$
\begin{aligned}
c_{\text{map}} &= \arg\max_{c \in \mathbb{C}} P(c|d) \\
&= \arg\max_{c \in \mathbb{C}} \frac{P(d|c)P(c)}{P(d)} && (13.9) \\
&= \arg\max_{c \in \mathbb{C}} P(d|c)P(c), && (13.10)
\end{aligned}
$$

where Bayes' rule (Equation (11.4), page 220) is applied in (13.9) and we drop the denominator in the last step because $P(d)$ is the same for all classes and does not affect the argmax.

We can interpret Equation (13.10) as a description of the generative process we assume in Bayesian text classification. To generate a document, we first choose class c with probability $P(c)$ (top nodes in Figures 13.4 and 13.5). The two models differ in the formalization of the second step, the generation of the document given the class, corresponding to the conditional distribution $P(d|c)$:

$$\textbf{Multinomial} \quad P(d|c) = P(\langle t_1, \ldots, t_k, \ldots, t_{n_d} \rangle | c) \tag{13.11}$$

$$\textbf{Bernoulli} \quad P(d|c) = P(\langle e_1, \ldots, e_i, \ldots, e_M \rangle | c), \tag{13.12}$$

where $\langle t_1, \ldots, t_{n_d} \rangle$ is the sequence of terms as it occurs in d (minus terms that were excluded from the vocabulary) and $\langle e_1, \ldots, e_i, \ldots, e_M \rangle$ is a binary vector of dimensionality M that indicates for each term whether it occurs in d or not.

It should now be clearer why we introduced the document space $\mathbb{X}$ in Equation (13.1) when we defined the classification problem. A critical step in solving a text classification problem is to choose the document representation. $\langle t_1, \ldots, t_{n_d} \rangle$ and $\langle e_1, \ldots, e_M \rangle$ are two different document representations. In the first case, $\mathbb{X}$ is the set of all term sequences (or, more precisely, sequences of term tokens). In the second case, $\mathbb{X}$ is $\{0, 1\}^M$.

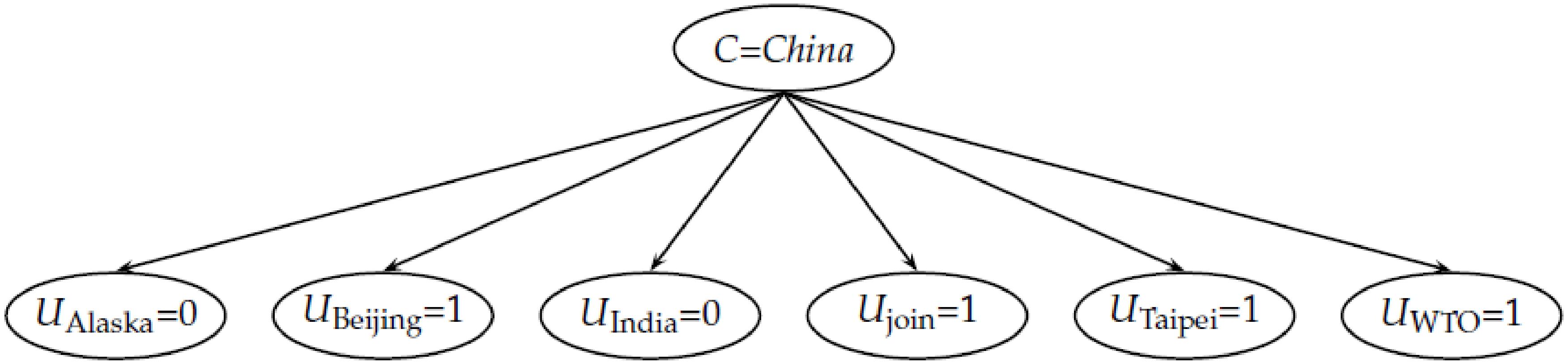
We cannot use Equations (13.11) and (13.12) for text classification directly. For the Bernoulli model, we would have to estimate $2^M|\mathbb{C}|$ different parameters, one for each possible combination of M values e_i and a class. The number of parameters in the multinomial case has the same order of magnitude.[3] This being a very large quantity, estimating these parameters reliably is infeasible.

CONDITIONAL INDEPENDENCE ASSUMPTION

To reduce the number of parameters, we make the Naive Bayes *conditional independence assumption*. We assume that attribute values are independent of each other given the class:

$$\textbf{Multinomial} \quad P(d|c) = P(\langle t_1, \ldots, t_{n_d} \rangle|c) = \prod_{1 \le k \le n_d} P(X_k = t_k|c) \tag{13.13}$$

$$\textbf{Bernoulli} \quad P(d|c) = P(\langle e_1, \ldots, e_M \rangle|c) = \prod_{1 \le i \le M} P(U_i = e_i|c). \tag{13.14}$$



▶ **Figure 13.5** The Bernoulli NB model.

```
# IR13A.py CS5154/6054 cheng 2022
# use the first and the last 1000 lines of bible.txt as two classes
# find top terms according to mutual information
# Usage: python IR13A.py

import numpy as np
import sklearn.feature_selection as fs
from sklearn.feature_extraction.text import CountVectorizer

f = open("bible.txt", "r")
docs = f.readlines()
f.close()
N =len(docs)
trainX = np.concatenate([docs[0:1000], docs[N-1000:N]])
y = np.concatenate([np.zeros(1000, dtype=np.int16), np.ones(1000,
dtype=np.int16)])

cv = CountVectorizer(binary=True, max_df=0.4, min_df=4)
X = cv.fit_transform(trainX).toarray()
print(X.shape)
voc = np.array(cv.get_feature_names())
```

```
mi = fs.mutual_info_classif(X, y)
sorted = np.argsort(mi)[::-1]
for i in range(10):
    index = sorted[i]
    print(voc[index], mi[index])

kbest = fs.SelectKBest(fs.mutual_info_classif)
kbest.fit(X, y)
support = np.array(kbest.get_support())
print(voc[support])
```

```
(2000, 1127)
said 0.06429532778705038
own 0.03471605459970184
liar 0.03361734171128305 5
wife 0.0331491147453513 16
mount 0.03280430038169757
darkness 0.032595691331536614
guile 0.03252702498251803
saints 0.03233423552008863
took 0.032092780916850305
set 0.0318478682931797 6
['amen' 'beloved' 'faith' 'jacob' 'jesus' 'noah' 'said'
'thee' 'things’ 'thy']
```

scikit learn

1.13. Feature selection

The classes in the `sklearn.feature_selection` module can be used for feature selection/dimensionality reduction on sample sets, either to improve estimators' accuracy scores or to boost their performance on very high-dimensional datasets.

1.13.1. Removing features with low variance

Toggle Menu

`hold` is a simple baseline approach to feature selection. It removes all fea-
riance doesn't meet some threshold. By default, it removes all zero-variance

File Edit View History Bookmarks Tools Help

sklearn.feature_selection.mutua

https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.mutua 150%

sklearn.feature_selection.mutual_info_classif

`sklearn.feature_selection.`**`mutual_info_classif`**(*X, y, *, discrete_features='auto', n_neighbors=3, copy=True, random_state=None*) [source]

Estimate mutual information for a discrete target variable.

Mutual information (MI) [1] between two random variables is a non-negative value, which measures the dependency between the variables. It is equal to zero if and only if two random … independent, and higher values mean higher dependency.

Toggle Menu

sklearn.feature_selection.chi2

`sklearn.feature_selection.chi2(X, y)` [source]

Compute chi-squared stats between each non-negative feature and class.

This score can be used to select the n_features features with the highest values for the test chi-squared statistic from X, which must contain only non-negative features such as booleans or frequencies (e.g., term counts in document classification), relative to the classes.

Recall that the chi-square test measures dependence between stochastic variables, so using this function “weeds out” the features that are the most likely to be independent of class and

Toggle Menu elevant for classification.

File Edit View History Bookmarks Tools Help

sklearn.feature_selection.chi2 — ×

https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.chi2.h

150%

Parameters::

X : *{array-like, sparse matrix} of shape (n_samples, n_features)*

Sample vectors.

y : *array-like of shape (n_samples,)*

Target vector (class labels).

Returns::

chi2 : *ndarray of shape (n_features,)*

Chi2 statistics for each feature.

p_values : *ndarray of shape (n_features,)*

P-values for each feature.

Toggle Menu

sklearn.feature_selection.SelectKBest

class sklearn.feature_selection.**SelectKBest**(*score_func=<function f_classif>*, ***, *k=10*)

[source]

Select features according to the k highest scores.

Read more in the User Guide.

Parameters::

score_func : *callable, default=f_classif*

Function taking two arrays X and y, and returning a pair of arrays (scores, pvalues) or a single array with scores. Default is f_classif (see below "See Also"). The default function only works ...fication tasks.

Methods

`fit`(X, y)	Run score function on (X, y) and get the appropriate features.
`fit_transform`(X[, y])	Fit to data, then transform it.
`get_feature_names_out`([input_features])	Mask feature names according to selected features.
`get_params`([deep])	Get parameters for this estimator.
`get_support`([indices])	Get a mask, or integer index, of the features selected.
`inverse_transform`(X)	Reverse the transformation operation.
`set_params`(**params)	Set the parameters of this estimator.
`transform`(X)	Reduce X to the selected features.

Toggle Menu

[source]

```
# IR13B.py CS5154/6054 cheng 2022
# BernoulliNB classification
# Usage: python IR13B.py

import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import BernoulliNB
from sklearn.metrics import accuracy_score

f = open("bible.txt", "r")
docs = f.readlines()
f.close()
N =len(docs)
trainX = np.concatenate([docs[0:1000], docs[N-1000:N]])
y = np.concatenate([np.zeros(1000, dtype=np.int16), np.ones(1000, dtype=np.int16)])
testX = np.concatenate([docs[1000:1100], docs[N-1100:N-1000]])
testY = np.concatenate([np.zeros(100, dtype=np.int16), np.ones(100, dtype=np.int16)])
```

```
cv = CountVectorizer(binary=True, max_df=0.4, min_df=4)
X = cv.fit_transform(trainX).toarray()
print(X.shape)
voc = cv.get_feature_names()
T = cv.transform(testX).toarray()

model = BernoulliNB()
model.fit(X, y)
pred = model.predict(T)
print(pred)
print ('Accuracy Score - ', accuracy_score(testY, pred))
```

```
(2000, 1127)
[0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 1 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1]
Accuracy Score -  0.92
```

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.

Read more in the User Guide.

Methods

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

Toggle Menu *ple_weight=None*) [source]