Naive Bayes

1. Overview

Naive Bayes is a classification algorithm. The algorithm is a classification method based on Bayes' theorem and the assumption of conditional independence of features. Among other things, Bayesian algorithm is based on Bayes' principle and uses the knowledge of probability statistics to classify the sample data set. Due to its solid mathematical foundation, the misclassification rate of Bayesian classification algorithm is very low.

Bayesian algorithm is characterized by combining prior and posterior probabilities, i.e., it avoids the supervisor bias of using only prior probabilities and the overfitting phenomenon of using sample information alone. Bayesian classification algorithms show higher accuracy with larger data sets, while the algorithms themselves are simpler. The plain Bayesian algorithm, on the other hand, is a corresponding simplification of the Bayesian algorithm, i.e., it assumes that the attributes are mutually conditionally independent of each other given the target value[1].

This means that no attribute variable has a greater or lesser weight on the decision outcome. Although this simplification reduces the classification effect of Bayesian classification algorithm to some extent, it greatly simplifies the complexity of Bayesian methods in practical application scenarios.

2. Advantages

(1) Having stable classification efficiency.

(2) performs well for small-scale data, can handle multi-classification problems, and is suitable for incremental training, especially when the amount of data exceeds the memory, and can do incremental training in batches.

(3) Less sensitive to missing data, the algorithm is also relatively simple, and is often used for text classification.

3. Disadvantages

(1) The algorithm is not very effective when the number of attributes is relatively large or when there is a large correlation between attributes (so an improvement is semi-parsimonious Bayes).

(2) the need to know the prior probabilities, which many times depend on assumptions and may lead to poor predictions due to assumptions (i.e., there is subjectivity).

(3) there is a certain error rate in the classification decision since the a priori data is used to determine the posterior probabilities and hence the classification.

(4) It is sensitive to the representation of the input data[2].

4. Methods of word disambiguation

There are two basic linguistic assumptions for performing word sense disambiguation. The first assumption is that all lexical senses of a polysemous word can be obtained from a dictionary, the second assumption is that a polysemous word has a specific lexical sense in a specific context.

From a methodological point of view, many computational linguistic problems can be formalized as a classification problem. The word sense disambiguation problem is also a typical classification problem, where the senses of a polysemous word in a certain context are attributed by a finite number of sense categories. In terms of specific methods, there are mainly rule-based methods, lexicon-based methods and corpus-based methods for word sense disambiguation. The rule-based approach means that a large number of rules are stored in the system, and words that satisfy the constraints of the rules are selected according to the rules.

Thanks to the spread of Internet technology, access to vast amounts of machine-readable electronic texts is no longer a difficulty. In addition, the processing power of large-scale corpora is no longer an insurmountable bottleneck due to the exponential growth in computer processing performance. The use of a large-scale corpus-based approach allows computers to automatically acquire knowledge for word sense disambiguation. Therefore, the main approaches currently used for word sense disambiguation are corpus-based approaches or a combination of corpus and lexicon approaches.

Methods based on manual acquisition of disambiguation knowledge can usually handle only a small number of polysemous words due to bottlenecks in knowledge acquisition. In contrast, large-rule corpus-based approaches can handle the disambiguation of almost all polysemous words in a text.

5. Word sense disambiguation algorithm based on Naive Bayes

Regardless of the type of words sense disambiguation method used, it consists of two main steps.

(1) Determining the set of all possible word senses that each word can take in the context.

(2) Selecting the most appropriate word meaning from the set of word meaning candidates.

Step (1) is relatively simple and involves retrieving the lexical items for a particular word from the dictionary. Step (2) embodies the essence of different methods of word sense disambiguation. The naive Bayesian method based on corpus compares the context of the word to be disambiguated with the relevant information in the corpus. Its main idea is that the specific meaning of a polysemous word in a text is often closely related to the theme of the text. In other words, the context of the word determines the meaning of the word. For example, in financial news, "bank" often refers to "financial institutions"; In the literature on hydraulic engineering, "bank" may usually refer to "embankment". According to this idea,

The main steps of the algorithm are as follows[3]:

Step 1: Retrieve all the senses of a particular word from the dictionary.

Step 2: Divide the relevant corpus

In the corpus, the texts containing the polysemous word are divided and classified according to the meaning of the word. When classifying, there may be a situation: the word may express multiple meanings in different parts of a specific document. In this case, according to the weight of word meaning in the text (frequency of occurrence), the text is divided into the category with the largest weight of word meaning.

Step 3: semantic characterization

According to the vector space model (VSM), the text content can be regarded as the "word package" of words appearing in the text. Because the dimension of word space is too large, in order to reduce the computational complexity of filtering analysis and reduce the phenomenon of over matching, the topic words in the text are usually used to replace the word set in the text. Therefore, the content of text D_i can be represented by the vector shown in equation (1)

$$\overline{D}_i = (T_1, W_1; T_2, W_2; \dots T_n, W_n) \qquad (1)$$

In formula (1), T_i is the subject word in the text, W_i is the weight of the subject word calculated according to the weight calculation formula, which is usually calculated according to $tf * idf$ formula.

Step 4: Calculate the probability of word sense items

Suppose the corpus corresponding to the j-th sense terms S_j of the polysemy T is denoted as C_j, and the probability $P\left(S_j \middle| D_j(T)\right)$ that the polysemy T in the text D_i is a sense S_j in the text, it is obvious that there exists equation (2):

$$\sum_j P\left(S_j \middle| D_j(T)\right) = 1 \qquad (2)$$

According to Bayes' theorem, the meaning of polysemous words in the new text can be selected through the corpus sample set, as shown in formula (3), and the posterior probability can be calculated through the prior probability and conditional probability obtained from the corpus.

$$P\left(S_j \middle| D_j(T)\right) = \frac{P\left(D_i(T)\middle|S_j\right) * P(S_j)}{P(D_i(T))} \qquad (3)$$

In formula (3), $P(D_i\ (T)|)$ is a constant for all the sense terms of the word T. In other words, the sense term for which $P(D_i(T)|S_j) * P(S_j)$ obtains the maximum value is the sense term of the word T in the text D_i.

In formula (3), $P(S\ )$ is the prior probability that word T takes word sense S_j in the corpus, which can be calculated by formula (4). In Eq. (4), the C_j denotes the meaning of word T in the corpus as of the text of S_j.

$$P(S_j) = \frac{|C_j|}{\sum_j |C_j|} \qquad (4)$$

The overhead of computing $P(D_i(T)|S_j)\,P$ can be very large, and to reduce the computational overhead, the plain assumption of conditional independence of subject words can be made: in a given sample set corresponding to the word sense S_j, the subject words are assumed to be conditionally independent of each other, i.e., there are no dependencies among these words. Therefore, the conditional probability $P(D_i(T)|S_j)$ can be calculated in a simplified manner according to formula (5).

$$P\left(D_i(T)\middle|S_j\right) = \prod_{k=1}^{N} P\left(T_k\middle|S_j\right) \qquad (5)$$

Each probability component in formula (5) can be approximated based on the corpus sample set, the $P(T_k|S_j)$ is calculated in formula (6)

$$P(T_k|S_j) = \frac{|T_k|}{|C_j|} \tag{6}$$

In formula (6), $|T_k|$ denotes the number of texts in corpus C_j with subject term T_k. After calculating the probability weights of the candidate senses of a particular word in the text, the sense with the largest probability weight is selected as result output.

6. nltk.classify.naivebayes module[4]

A classifier based on the Naive Bayes algorithm. In order to find the probability for a label, this algorithm first uses the Bayes rule to express P(label|features) in terms of P(label) and P(features|label):

$$P(label|features) = \frac{P(label) * P(features|label)}{P(features)}$$

The algorithm then makes the 'naive' assumption that all features are independent, given the label:

$$P(label|features) = \frac{P(label) * P(f1|label) * \ldots * P(fn|label)}{P(features)}$$

Rather than computing P(features) explicitly, the algorithm just calculates the numerator for each label, and normalizes them so they sum to one:

$$P(label|features) = \frac{P(label) * P(f1|label) * \ldots * P(fn|label)}{SUM[l](P(l) * P(f1|l) * \ldots * P(fn|l))}$$

class **nltk.classify.naivebayes.NaiveBayesClassifier**

Bases: nltk.classify.api.ClassifierI

A Naive Bayes classifier. Naive Bayes classifiers are paramaterized by two probability distributions:

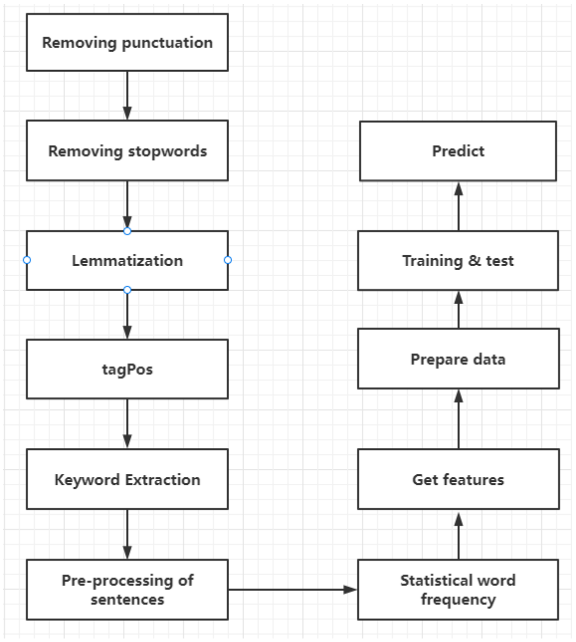
P(label) gives the probability that an input will receive each label, given no information about the input's features.

P(fname=fval|label) gives the probability that a given feature (fname) will receive a given value (fval), given that the label (label).

If the classifier encounters an input with a feature that has never been seen with any label, then rather than assigning a probability of 0 to all labels, it will ignore that feature.

The feature value 'None' is reserved for unseen feature values; you generally should not use 'None' as a feature value for one of your own features.

7. Implement process



Code idea
Specify a word, such as 'bank', and find all sentences containing 'bank'. Then 8:2 divide the training set and test set.

Remaining issues
The accuracy is not too high, accuracy is 0.0945945945945946.
I also used SVM for the process and got a similar result. The problem that comes to mind is that the dataset does not match.

Generally, the data in the dataset and the label correspond to each other
The normal WSD dataset is a sentence, for example, disambiguation of ‘bank’ (I go to bank to deposit money), in this statement it is considered that bank is a noun and there is a corresponding label match.

But the two datasets, corpus and label, are separated in this word's assignment.
A sentence has no label and can only be located out by NLTK, so there may be multiple results.
And Naïve bayes is ML without neural network nonlinear expression ability, just calculate the statistical law to get the result, so the result is not very satisfactory.

One solution that comes to mind is to restart the Jupyter Notebook , the random package is not seeded the same and the dataset is divided differently There may be different effects. Next is to replace the dataset.

Reference

[1] Ahlswede T., Lorand D.. Word sense disambiguation by human subjects: computational and psycholinguistic implications. In: Proceedings of the Workshop on Acquisitions of Lexical Knowledge from Text. Columbus,Ohio,1993. pp.1-9.

[2] Jadhav, S.D. and Channe, H.P., 2016. Comparative study of K-NN, naive Bayes and decision tree classification techniques. International Journal of Science and Research (IJSR), 5(1), pp.1842-1845.

[3] Lau, J.H., Cook, P., McCarthy, D., Newman, D. and Baldwin, T., 2012, April. Word sense induction for novel sense detection. In Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics (pp. 591-601).

[4] NLTK :: nltk.classify.naivebayes module