Naive Bayes classifier

From Wikipedia, the free encyclopedia

In machine learning, **naive Bayes classifiers** are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features.

Naive Bayes has been studied extensively since the 1950's. It was introduced under a different name into the text retrieval community in the early 1960's,[1]:488 and remains a popular (baseline) method for text categorization, the problem of judging documents as belonging to one category or the other (such as spam or legitimate, sports or politics, etc.) with word frequencies as the features. With appropriate pre-processing, it is competitive in this domain with more advanced methods including support vector machines.[2] It also finds application in automatic medical diagnosis.[3]

Naive Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem. Maximum-likelihood training can be done by evaluating a closed-form expression,[1]:718 which takes linear time, rather than by expensive iterative approximation as used for many other types of classifiers.

In the statistics and computer science literature, Naive Bayes models are known under a variety of names, including **simple Bayes** and **independence Bayes**.[4] All these names reference the use of Bayes' theorem in the classifier's decision rule, but naive Bayes is not (necessarily) a Bayesian method.[1][4]

Contents

Introduction

Naive Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. It is not a single algorithm for training such classifiers, but a family of algorithms based on a common principle: all naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable. For example, a fruit may be considered to be an apple if it is red, round, and about 10 cm in diameter. A naive Bayes classifier considers each of these features to contribute independently to the probability that this fruit is an apple, regardless of any possible correlations between the color, roundness, and diameter features.

For some types of probability models, naive Bayes classifiers can be trained very efficiently in a supervised learning setting. In many practical applications, parameter estimation for naive Bayes models uses the method of maximum likelihood; in other words, one can work with the naive Bayes model without accepting Bayesian probability or using any Bayesian methods.

Despite their naive design and apparently oversimplified assumptions, naive Bayes classifiers have worked quite well in many complex real-world situations. In 2004, an analysis of the Bayesian classification problem showed that there are sound theoretical reasons for the apparently implausible efficacy of naive Bayes classifiers.[5] Still, a comprehensive comparison with other classification algorithms in 2006 showed that Bayes classification is outperformed by other approaches, such as boosted trees or random forests.[6]

An advantage of naive Bayes is that it only requires a small number of training data to estimate the parameters necessary for classification.

Probabilistic model

Abstractly, naive Bayes is a conditional probability model: given a problem instance to be classified, represented by a vector $\mathbf{x} = (x_1, \ldots, x_n)$ representing some n features (independent variables), it assigns to this instance probabilities

$$p(C_k|x_1, \ldots, x_n)$$

for each of K possible outcomes or *classes* C_k.[7]

The problem with the above formulation is that if the number of features n is large or if a feature can take on a large number of values, then basing such a model on probability tables is infeasible. We therefore reformulate the model to make it more tractable. Using Bayes' theorem, the conditional probability can be decomposed as

$$p(C_k|\mathbf{x}) = \frac{p(C_k)\ p(\mathbf{x}|C_k)}{p(\mathbf{x})}$$

In plain English, using Bayesian probability terminology, the above equation can be written as

$$\text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}}$$

In practice, there is interest only in the numerator of that fraction, because the denominator does not depend on C and the values of the features F_i are given, so that the denominator is effectively constant. The numerator is equivalent to the joint probability model

$$p(C_k, x_1, \ldots, x_n)$$

which can be rewritten as follows, using the chain rule for repeated applications of the definition of conditional probability:

$$\begin{aligned} p(C_k, x_1, \ldots, x_n) &= p(x_1, \ldots, x_n, C_k) \\ &= p(x_1 | x_2, \ldots, x_n, C_k) p(x_2, \ldots, x_n, C_k) \\ &= p(x_1 | x_2, \ldots, x_n, C_k) p(x_2 | x_3, \ldots, x_n, C_k) p(x_3, \ldots, x_n, C_k) \\ &= \ldots \\ &= p(x_1 | x_2, \ldots, x_n, C_k) p(x_2 | x_3, \ldots, x_n, C_k) \ldots p(x_{n-1} | x_n, C_k) p(x_n | C_k) p(C_k) \end{aligned}$$

Now the "naive" conditional independence assumptions come into play: assume that each feature F_i is conditionally independent of every other feature F_j for $j \neq i$, given the category C. This means that

$$p(x_i | x_{i+1}, \ldots, x_n, C_k) = p(x_i | C_k) \ .$$

Thus, the joint model can be expressed as

$$\begin{aligned} p(C_k | x_1, \ldots, x_n) &\propto p(C_k, x_1, \ldots, x_n) \\ &\propto p(C_k)\ p(x_1 | C_k)\ p(x_2 | C_k)\ p(x_3 | C_k)\ \cdots \\ &\propto p(C_k) \prod_{i=1}^{n} p(x_i | C_k)\ . \end{aligned}$$

This means that under the above independence assumptions, the conditional distribution over the class variable C is:

$$p(C_k | x_1, \ldots, x_n) = \frac{1}{Z} p(C_k) \prod_{i=1}^{n} p(x_i | C_k)$$

where the evidence $Z = p(\mathbf{x})$ is a scaling factor dependent only on $x_1, \ldots, x_n$, that is, a constant if the values of the feature variables are known.

Constructing a classifier from the probability model

The discussion so far has derived the independent feature model, that is, the naive Bayes probability model. The naive Bayes classifier combines this model with a decision rule. One common rule is to pick the hypothesis that is most probable; this is known as the *maximum a posteriori* or *MAP* decision rule. The corresponding classifier, a Bayes classifier, is the function that assigns a class label $\hat{y} = C_k$ for some k as follows:

$$\hat{y} = \underset{k \in \{1, \ldots, K\}}{\operatorname{argmax}}\ p(C_k) \prod_{i=1}^{n} p(x_i | C_k).$$

Parameter estimation and event models

A class's prior may be calculated by assuming equiprobable classes (i.e., priors = 1 / (number of classes)), or by calculating an estimate for the class probability from the training set (i.e., (prior for a given class) = (number of samples in the class) / (total number of samples)). To estimate the parameters for a feature's distribution, one must assume a distribution or generate nonparametric models for the features from the training set.[8]

The assumptions on distributions of features are called the *event model* of the Naive Bayes classifier. For discrete features like the ones encountered in document classification (include spam filtering), multinomial and Bernoulli distributions are popular. These assumptions lead to two distinct models, which are often confused.[9][10]

Gaussian naive Bayes

When dealing with continuous data, a typical assumption is that the continuous values associated with each class are distributed according to a Gaussian distribution. For example, suppose the training data contain a continuous attribute, x. We first segment the data by the class, and then compute the mean and variance of x in each class. Let μ_c be the mean of the values in x associated with class c, and let σ_c^2 be the variance of the values in x associated with class c. Suppose we have collected some observation value v. Then, the probability *distribution* of v given a class c, $p(x = v|c)$, can be computed by plugging v into the equation for a Normal distribution parameterized by μ_c and σ_c^2. That is,

$$p(x = v|c) = \frac{1}{\sqrt{2\pi\sigma_c^2}}\, e^{-\frac{(v-\mu_c)^2}{2\sigma_c^2}}$$

Another common technique for handling continuous values is to use binning to discretize the feature values, to obtain a new set of Bernoulli-distributed features; some literature in fact suggests that this is necessary to apply naive Bayes, but it is not, and the discretization may throw away discriminative information.[4]

Multinomial naive Bayes

With a multinomial event model, samples (feature vectors) represent the frequencies with which certain events have been generated by a multinomial $(p_1, \ldots, p_n)$ where p_i is the probability that event i occurs (or K such multinomials in the multiclass case). A feature vector $\mathbf{x} = (x_1, \ldots, x_n)$ is then a histogram, with x_i counting the number of times event i was observed in a particular instance. This is the event model typically used for document classification, with events representing the occurrence of a word in a single document (see bag of words assumption). The likelihood of observing a histogram $\mathbf{x}$ is given by

$$p(\mathbf{x}|C_k) = \frac{(\sum_i x_i)!}{\prod_i x_i!} \prod_i {p_{ki}}^{x_i}$$

The multinomial naive Bayes classifier becomes a linear classifier when expressed in log-space:[2]

$$\begin{aligned}\log p(C_k|\mathbf{x}) &\propto \log\left(p(C_k)\prod_{i=1}^{n} {p_{ki}}^{x_i}\right)\\ &= \log p(C_k) + \sum_{i=1}^{n} x_i \cdot \log p_{ki}\\ &= b + \mathbf{w}_k^\top \mathbf{x}\end{aligned}$$

where $b = \log p(C_k)$ and $w_{ki} = \log p_{ki}$.

If a given class and feature value never occur together in the training data, then the frequency-based probability estimate will be zero. This is problematic because it will wipe out all information in the other probabilities when they are multiplied. Therefore, it is often desirable to incorporate a small-sample correction, called pseudocount, in all probability estimates such that no probability is ever set to be exactly zero. This way of regularizing naive Bayes is called Laplace smoothing when the pseudocount is one, and Lidstone smoothing in the general case.

Rennie *et al.* discuss problems with the multinomial assumption in the context of document classification and possible ways to alleviate those problems, including the use of tf–idf weights instead of raw term frequencies and document length normalization, to produce a naive Bayes classifier that is competitive with support vector machines.[2]

Bernoulli naive Bayes

In the multivariate Bernoulli event model, features are independent booleans (binary variables) describing inputs. Like the multinomial model, this model is popular for document classification tasks,[9] where binary term occurrence features are used rather than term frequencies. If x_i is a boolean expressing the occurrence or absence of the i'th term from the vocabulary, then the likelihood of a document given a class C_k is given by[9]

$$p(\mathbf{x}|C_k) = \prod_{i=1}^{n} p_{ki}^{x_i}(1 - p_{ki})^{(1-x_i)}$$

where p_{ki} is the probability of class C_k generating the term w_i. This event model is especially popular for classifying short texts. It has the benefit of explicitly modelling the absence of terms. Note that a naive Bayes classifier with a Bernoulli event model is not the same as a multinomial NB classifier with frequency counts truncated to one.

Semi-supervised parameter estimation

Given a way to train a naive Bayes classifier from labeled data, it's possible to construct a semi-supervised training algorithm that can learn from a combination of labeled and unlabeled data by running the supervised learning algorithm in a loop:[11]

> Given a collection $D = L \uplus U$ of labeled samples L and unlabeled samples U, start by training a naive Bayes classifier on L.
> Until convergence, do:
>
> > Predict class probabilities $P(C|x)$ for all examples x in D.
> > Re-train the model based on the *probabilities* (not the labels) predicted in the previous step.

Convergence is determined based on improvement to the model likelihood $P(D|\theta)$, where θ denotes the parameters of the naive Bayes model.

This training algorithm is an instance of the more general expectation–maximization algorithm (EM): the prediction step inside the loop is the *E*-step of EM, while the re-training of naive Bayes is the *M*-step. The algorithm is formally justified by the assumption that the data are generated by a mixture model, and the components of this mixture model are exactly the classes of the classification problem.[11]

Discussion

Despite the fact that the far-reaching independence assumptions are often inaccurate, the naive Bayes classifier has several properties that make it surprisingly useful in practice. In particular, the decoupling of the class conditional feature distributions means that each distribution can be independently estimated as a one-dimensional distribution. This helps alleviate problems stemming from the curse of dimensionality, such as the need for data sets that scale exponentially with the number of features. While naive Bayes often fails to produce a good estimate for the correct class probabilities,[12] this may not be a requirement for many applications. For example, the naive Bayes classifier will make the correct MAP decision rule classification so long as the correct class is more probable than any other class. This is true regardless of whether the probability estimate is slightly, or even grossly inaccurate. In this manner, the overall classifier can be robust enough to ignore serious deficiencies in its underlying naive probability model.[3] Other reasons for the observed success of the naive Bayes classifier are discussed in the literature cited below.

Relation to logistic regression

In the case of discrete inputs (indicator or frequency features for discrete events), naive Bayes classifiers form a *generative-discriminative* pair with (multinomial) logistic regression classifiers: each naive Bayes classifier can be considered a way of fitting a probability model that optimizes the joint likelihood $p(C, \mathbf{x})$, while logistic regression fits the same probability model to optimize the conditional $p(C|\mathbf{x})$.[13]

The link between the two can be seen by observing that the decision function for naive Bayes (in the binary case) can be rewritten as "predict class C_1 if the odds of $p(C_1|\mathbf{x})$ exceed those of $p(C_2|\mathbf{x})$". Expressing this in log-space gives:

$$\log \frac{p(C_1|\mathbf{x})}{p(C_2|\mathbf{x})} = \log p(C_1|\mathbf{x}) - \log p(C_2|\mathbf{x}) > 0$$

The left-hand side of this equation is the log-odds, or *logit*, the quantity predicted by the linear model that underlies logistic regression. Since naive Bayes is also a linear model for the two "discrete" event models, it can be reparametrised as a linear function $b + \mathbf{w}^\top x > 0$. Obtaining the probabilities is then a matter of applying the logistic function to $b + \mathbf{w}^\top x$, or in the multiclass case, the softmax function.

Discriminative classifiers have lower asymptotic error than generative ones; however, research by Ng and Jordan has shown that in some practical cases naive Bayes can outperform logistic regression because it reaches its asymptotic error faster.[13]

Examples

Gender classification

Problem: classify whether a given person is a male or a female based on the measured features. The features include height, weight, and foot size.

Training

Example training set below.

Gender	height (feet)	weight (lbs)	foot size(inches)
male	6	180	12
male	5.92 (5'11")	190	11
male	5.58 (5'7")	170	12
male	5.92 (5'11")	165	10
female	5	100	6
female	5.5 (5'6")	150	8
female	5.42 (5'5")	130	7
female	5.75 (5'9")	150	9

The classifier created from the training set using a Gaussian distribution assumption would be (given variances are *unbiased* sample variances):

Gender	mean (height)	variance (height)	mean (weight)	variance (weight)	mean (foot size)	variance (foot size)
male	5.855	3.5033e-02	176.25	1.2292e+02	11.25	9.1667e-01
female	5.4175	9.7225e-02	132.5	5.5833e+02	7.5	1.6667e+00

Let's say we have equiprobable classes so P(male)= P(female) = 0.5. This prior probability distribution might be based on our knowledge of frequencies in the larger population, or on frequency in the training set.

Testing

Below is a sample to be classified as a male or female.

Gender	height (feet)	weight (lbs)	foot size(inches)
sample	6	130	8

We wish to determine which posterior is greater, male or female. For the classification as male the posterior is given by

$$\text{posterior (male)} = \frac{P(\text{male})\, p(\text{height}|\text{male})\, p(\text{weight}|\text{male})\, p(\text{foot size}|\text{male})}{evidence}$$

For the classification as female the posterior is given by

$$\text{posterior (female)} = \frac{P(\text{female})\, p(\text{height}|\text{female})\, p(\text{weight}|\text{female})\, p(\text{foot size}|\text{female})}{evidence}$$

The evidence (also termed normalizing constant) may be calculated:

$$\begin{aligned}\text{evidence} = {} & P(\text{male})\, p(\text{height}|\text{male})\, p(\text{weight}|\text{male})\, p(\text{foot size}|\text{male}) \\ & + P(\text{female})\, p(\text{height}|\text{female})\, p(\text{weight}|\text{female})\, p(\text{foot size}|\text{female})\end{aligned}$$

However, given the sample, the evidence is a constant and thus scales both posteriors equally. It therefore does not affect classification and can be ignored. We now determine the probability distribution for the sex of the sample.

$$\begin{aligned}&P(\text{male}) = 0.5 \\ &p(\text{height}|male) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(6-\mu)^2}{2\sigma^2}\right) \approx 1.5789\,,\end{aligned}$$

where $\mu = 5.855$ and $\sigma^2 = 3.5033 \cdot 10^{-2}$ are the parameters of normal distribution which have been previously determined from the training set. Note that a value greater than 1 is OK here – it is a probability density rather than a probability, because *height* is a continuous variable.

$$\begin{aligned}&p(\text{weight}|male) = 5.9881 \cdot 10^{-6} \\ &p(\text{foot size}|male) = 1.3112 \cdot 10^{-3} \\ &\text{posterior numerator (male)} = \text{their product} = 6.1984 \cdot 10^{-9}\end{aligned}$$

$$\begin{aligned}&P(\text{female}) = 0.5 \\ &p(\text{height}|\text{female}) = 2.2346 \cdot 10^{-1} \\ &p(\text{weight}|\text{female}) = 1.6789 \cdot 10^{-2} \\ &p(\text{foot size}|\text{female}) = 2.8669 \cdot 10^{-1} \\ &\text{posterior numerator (female)} = \text{their product} = 5.3778 \cdot 10^{-4}\end{aligned}$$

Since posterior numerator is greater in the female case, we predict the sample is female.

Document classification

Here is a worked example of naive Bayesian classification to the document classification problem. Consider the problem of classifying documents by their content, for example into spam and non-spam e-mails. Imagine that documents are drawn from a number of classes of documents which can be modelled as sets of words where the (independent) probability that the i-th word of a given document occurs in a document from class *C* can be written as

$$p(w_i|C)$$

(For this treatment, we simplify things further by assuming that words are randomly distributed in the document - that is, words are not dependent on the length of the document, position within the document with relation to other words, or other document-context.)

Then the probability that a given document *D* contains all of the words w_i, given a class *C*, is

$$p(D|C) = \prod_i p(w_i|C)$$

The question that we desire to answer is: "what is the probability that a given document *D* belongs to a given class *C*?" In other words, what is $p(C|D)$?

Now by definition

$$p(D|C) = \frac{p(D \cap C)}{p(C)}$$

and

$$p(C|D) = \frac{p(D \cap C)}{p(D)}$$

Bayes' theorem manipulates these into a statement of probability in terms of likelihood.

$$p(C|D) = \frac{p(C)}{p(D)} p(D|C)$$

Assume for the moment that there are only two mutually exclusive classes, *S* and ¬*S* (e.g. spam and not spam), such that every element (email) is in either one or the other;

$$p(D|S) = \prod_i p(w_i|S)$$

and

$$p(D|\neg S) = \prod_i p(w_i|\neg S)$$

Using the Bayesian result above, we can write:

$$p(S|D) = \frac{p(S)}{p(D)} \prod_i p(w_i|S)$$

$$p(\neg S|D) = \frac{p(\neg S)}{p(D)} \prod_i p(w_i|\neg S)$$

Dividing one by the other gives:

$$\frac{p(S|D)}{p(\neg S|D)} = \frac{p(S) \prod_i p(w_i|S)}{p(\neg S) \prod_i p(w_i|\neg S)}$$

Which can be re-factored as:

$$\frac{p(S|D)}{p(\neg S|D)} = \frac{p(S)}{p(\neg S)} \prod_i \frac{p(w_i|S)}{p(w_i|\neg S)}$$

Thus, the probability ratio p(S | D) / p($\neg S$ | D) can be expressed in terms of a series of likelihood ratios. The actual probability p(S | D) can be easily computed from log (p(S | D) / p($\neg S$ | D)) based on the observation that p(S | D) + p($\neg S$ | D) = 1.

Taking the logarithm of all these ratios, we have:

$$\ln \frac{p(S|D)}{p(\neg S|D)} = \ln \frac{p(S)}{p(\neg S)} + \sum_i \ln \frac{p(w_i|S)}{p(w_i|\neg S)}$$

(This technique of "log-likelihood ratios" is a common technique in statistics. In the case of two mutually exclusive alternatives (such as this example), the conversion of a log-likelihood ratio to a probability takes the form of a sigmoid curve: see logit for details.)

Finally, the document can be classified as follows. It is spam if $p(S|D) > p(\neg S|D)$ (i.e., $\ln \frac{p(S|D)}{p(\neg S|D)} > 0$), otherwise it is not spam.

See also

- AODE
- Bayesian spam filtering
- Bayesian network
- Random naive Bayes
- Linear classifier
- Logistic regression
- Perceptron
- Take-the-best heuristic

References

1. Russell, Stuart; Norvig, Peter (2003) [1995]. *Artificial Intelligence: A Modern Approach* (2nd ed.). Prentice Hall. ISBN 978-0137903955.
2. Rennie, J.; Shih, L.; Teevan, J.; Karger, D. (2003). *Tackling the poor assumptions of Naive Bayes classifiers* (PDF). ICML.
3. Rish, Irina (2001). *An empirical study of the naive Bayes classifier* (PDF). IJCAI Workshop on Empirical Methods in AI.
4. Hand, D. J.; Yu, K. (2001). "Idiot's Bayes — not so stupid after all?". *International Statistical Review*. **69** (3): 385–399. doi:10.2307/1403452. ISSN 0306-7734.
5. Zhang, Harry. *The Optimality of Naive Bayes* (PDF). FLAIRS2004 conference.
6. Caruana, R.; Niculescu-Mizil, A. (2006). *An empirical comparison of supervised learning algorithms*. Proc. 23rd International Conference on Machine Learning. CiteSeerX: 10.1.1.122.5901.
7. Narasimha Murty, M.; Susheela Devi, V. (2011). *Pattern Recognition: An Algorithmic Approach*. ISBN 0857294946.
8. John, George H.; Langley, Pat (1995). *Estimating Continuous Distributions in Bayesian Classifiers*. Proc. Eleventh Conf. on Uncertainty in Artificial Intelligence. Morgan Kaufmann. pp. 338–345.
9. McCallum, Andrew; Nigam, Kamal (1998). *A comparison of event models for Naive Bayes text classification* (PDF). AAAI-98 workshop on learning for text categorization.
10. Metsis, Vangelis; Androutsopoulos, Ion; Paliouras, Georgios (2006). *Spam filtering with Naive Bayes—which Naive Bayes?*. Third conference on email and anti-spam (CEAS).
11. Nigam, Kamal; McCallum, Andrew; Thrun, Sebastian; Mitchell, Tom (2000). "Learning to classify text from labeled and unlabeled documents using EM" (PDF). *Machine Learning*.

12. Niculescu-Mizil, Alexandru; Caruana, Rich (2005). *Predicting good probabilities with supervised learning* (PDF). ICML. doi:10.1145/1102351.1102430.
13. Ng, Andrew Y.; Jordan, Michael I. (2002). *On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes*. NIPS.

Further reading

- Domingos, Pedro; Pazzani, Michael (1997). "On the optimality of the simple Bayesian classifier under zero-one loss". *Machine Learning*. **29**: 103–137.
- Webb, G. I.; Boughton, J.; Wang, Z. (2005). "Not So Naive Bayes: Aggregating One-Dependence Estimators". *Machine Learning*. Springer. **58** (1): 5–24. doi:10.1007/s10994-005-4258-6.
- Mozina, M.; Demsar, J.; Kattan, M.; Zupan, B. (2004). *Nomograms for Visualization of Naive Bayesian Classifier* (PDF). Proc. PKDD-2004. pp. 337–348.
- Maron, M. E. (1961). "Automatic Indexing: An Experimental Inquiry". *JACM*. **8** (3): 404–417. doi:10.1145/321075.321084.
- Minsky, M. (1961). *Steps toward Artificial Intelligence*. Proc. IRE. pp. 8–30.

External links

- Book Chapter: Naive Bayes text classification, Introduction to Information Retrieval (http://nlp.stanford.edu/IR-book/html/htmledition/naive-bayes-text-classification-1.html)
- Naive Bayes for Text Classification with Unbalanced Classes (http://www.cs.waikato.ac.nz/~eibe/pubs/FrankAndBouckaertPKDD06new.pdf)
- Benchmark results of Naive Bayes implementations (http://tunedit.org/results?d=UCI/&a=bayes)
- Hierarchical Naive Bayes Classifiers for uncertain data (http://www.biomedcentral.com/1471-2105/7/514) (an extension of the Naive Bayes classifier).

Software

- Naive Bayes classifiers are available in many general-purpose machine learning and NLP packages, including Apache Mahout, Mallet (http://mallet.cs.umass.edu/), NLTK, Orange, scikit-learn and Weka.
- IMSL Numerical Libraries Collections of math and statistical algorithms available in C/C++, Fortran, Java and C#/.NET. Data mining routines in the IMSL Libraries include a Naive Bayes classifier.
- An interactive Microsoft Excel spreadsheet Naive Bayes implementation (http://downloads.sourceforge.net/naivebayesclass/NaiveBayesDemo.xls?use_mirror=osdn) using VBA (requires enabled macros) with viewable source code.
- jBNC - Bayesian Network Classifier Toolbox (http://jbnc.sourceforge.net/)
- Statistical Pattern Recognition Toolbox for Matlab (http://cmp.felk.cvut.cz/cmp/software/stprtool/).
- ifile (http://people.csail.mit.edu/jrennie/ifile/) - the first freely available (Naive) Bayesian mail/spam filter
- NClassifier (http://nclassifier.sourceforge.net/) - NClassifier is a .NET library that supports text classification and text summarization. It is a port of Classifier4J.
- Classifier4J (http://classifier4j.sourceforge.net/) - Classifier4J is a Java library designed to do text classification. It comes with an implementation of a Bayesian classifier.

Categories: Classification algorithms | Statistical classification