# Naive Bayes

Given an example input **x**, the probability of it belonging to class c is given to be



where  and , i.e., there are *K* classes in total. In Bayesian inference framework, we will assign the example into the class which maximizes the posterior, that is



In , the evidence  is independent of the class . Therefore, we can transform the optimization problem into



In a Naive Bayes Classifier (NBC), the basic assumption is conditional independency. That it, in each class , we assume the *m* features of the input are independent from each other, which implies



where is the value of the feature of one example . Now the classification result of a NBC is



The remaining problem is to determine the class prior  and the conditional distribution of each feature .

## Estimation of

In this problem, there are only two classes, for spam and or for non-spam. Therefore, it is reasonable to assume a Bernoulli distribution as the class prior, i.e., , where is probability of being a spam . Since there are enough examples in the training set, for simplicity, we just use a maximum likelihood method to estimate the parameter , which turns out to be



where and are the number of spam and non-spam mails in the training set respectively.

## Estimation of : MAP or MLE?

Since our data has been binarized, each feature can have a value of 0 or 1. Supposing the probability of feature in class *c* being 1 is , i.e., , we know that each feature in each class in Bernoulli distributed: . The remaining work is to infer the parameter value with the given training set .

As the parameter is only concerned with the class , we extract a subset from with the examples from class , that is



In this case, we will take the Bayesian methods to infer the unknown parameter with the Maximum A Posteriori estimation. In this framework, the unknown parameter will also be considered as a random variable with a prior probability , where because is the parameter of the above Bernoulli distribution. The posterior of given the training subset is



We assume the prior of is a Beta distribution, the conjugate prior of the Bernoulli distribution. Then we have



where beta function, *B*, is a normalization constant to ensure that the total probability integrates to 1. In practice, we should choose the two shape parameters and according to our prior knowledge. Then, for  in , if the examples in are independently identically distributed, then  is a binomial distribution with respect to the binary value of the feature. We count the number of examples in according to the feature value as follows:



Then, the likelihood  in is



The MAP estimation from is



Applying a logarithmic transformation on yields



In the prior  is parameterized by and and the likelihood is given in . Now we can insert and into the optimization objective , which leads to



Note that the last term of the above two equations in are both constants which are independent of . Therefore, combining and , the MAP estimation is now turned into



The derivative of with respect to is



By making the derivative zero, we get the maximizer to be



where is size of the subset .

In fact, from and we can also see that



which indicates the posterior of is still Beta distributed with two updated shape parameters compared with its prior. This stems from the fact that the prior we have chosen, Beta distribution, is exactly the conjugate distribution of Bernoulli and binomial distribution. To maximize the posterior in , the solution is the unique mode of Beta distribution, that is



which just leads to the same answer as .

## Bayesian testing

In the above sections 1.2 and 1.3, we have estimated the prior and the likelihood from the training set *D* based on the conditional independency assumption of a Naive Bayesian Classifier. As we have shown in equation , now that we have got  and , given a test example , its class label is predicted by the Naive Bayesian Classifier to be



which maximizes the posterior probability . The class prior and the feature likelihood in each class are



In practice, to avoid numeric issues such as underflow, we compute the logarithmic transform of the objective function in instead, that is,



# Gaussian Naive Bayes

In this question, the data are not binarized. Therefore, to build a Naive Bayes Classifier, we assume the continuous feature data are Gaussian distributed conditioned on each class, i.e.,



We use the maximum likelihood method to estimate both the class prior and the conditional feature distribution ). The estimation of is still shown in . Next, we will derive the MLE for Gaussian distribution.

## MLE of Gaussian distribution

For notational simplicity, consider *N* scalar observations  stemming from a Gaussian distribution . Assume these observations are mutually independent. Then the log-likelihood function for the *N* observations is



According to , we obtain



Inserting into leads to



whose derivative with respect to is



Equating to zero gives



Next, to facilitate the computation, let  and take the derivative of with respect to *s*. We get



By making , we can obtain the maximum likelihood estimation of the variance is



In conclusion, we can see that maximum likelihood estimation of the Gaussian distribution parameters are exactly the corresponding sample mean and sample variance.

Let’s come back to our Naive Bayes Classifier problem in . The maximum likelihood estimation for the mean and variance parameters of the class dependent feature Gaussian distribution is given by



where is a subset of examples belonging to class *c* with and is the value of the feature of . Now with the estimated distribution parameters , we can continue the Naive Bayes predication as shown in subsection 1.3.

# Logistic Regression

In Bayesian classification, the posterior probability for class assignment  is estimated via the respective conditional pdfs, which is not, in general, an easy task. On the contrary, the logistic regression method can model such posterior probabilities directly, where the distribution of data is of no interest. Therefore, logistic regression is a discriminative approach.

## Definition

For simplicity, we consider a binary classification problem. There are only two possible classes and the output (the class label) can be 0 or 1. The mode assumes that



where  is the *standard logistic function*. A bias term has been involved in the parameter vector  as the first element and the sample input has been prepended with 1 to allow the bias term, that is, . Here, *m* is the number of features. It is easy to write the posterior probability of an example **x** being in class 0:



We can also see that



which is known as the *logistic odds*. Supposing there are only two classes, and , the posterior probability for class predication of and can be integrated together as



## Parameter estimation

In logistic regression, given a training set , the parameter vector **w** is estimated via the maximum likelihood method. In the next, for notational conciseness, the logistic function  will be notated by a simple *f*. Then, the logistic regression model in becomes



The likelihood function for the training set *D* is written as



where . To facilitate the subsequent MLE, we consider the negative log-likelihood function given by



The derivative of the standard logistic function is



which implies



according to the chain rule. Now it is readily to write the gradient of  with respect to **w** as follows



The Hessian matrix of with respect to **w** is



We can express the gradient and Hessian using matrix notation as follows



where



Similarly, using the above matrix notation, the negative log-likelihood function in is rewritten into



To estimate the parameter vector , we will adopt the maximum likelihood estimation, that is, to minimize the negative log-likelihood . It should be noted that since for any finite input t, the standard logistic function , we can conclude from that is positive definite. The proof is straightforward and shown in below.



Now that is convex, we can safely the descent optimization methods such as the gradient descent or Newton’ method to find the unique global minimum.

## Optimization: gradient descent and Newton’s method

A general unconstrained optimization problem can be stated as



where is a convex and twice continuously differentiable. A general descent method produces a minimizing sequence as following



where  is the *step size* and  is the *step* or *search direction*.

By the Taylor expansion theorem, we know that



where we neglect the 2nd-order and other higher-order terms and  is the gradient. In all decent methods, we expect



except when  is already the minimizer in order to minimize the target function in an iterative manner. Combining and , the search direction is required to satisfy



i.e., it must make an acute angle with the negative gradient. We call such a direction a *descent direction*. There are a few candidates for the choice of descent directions and next we focus on the gradient descent method and the Newton’s method.

### Gradient descent method

According to , a natural choice of the search direction is negative gradient, that is



The stopping criterion in the iteration of gradient descent approach is usually set to be , where is a small positive number since we know that the gradient will be zero exactly at the minimizer.

### Newton’s method

Now let’s consider the 2nd-order Taylor approximation of the target function , which is



If is convex, i.e., , then equation is indeed a convex quadratic function of the step . To minimize , we can equal its gradient with respect **v** to be zero and we get



This is called a *Newton step*  . Now consider the requirement of for descent directions. The positive definiteness of  implies that



unless  is zero. Therefore, the Newton step is a valid descent direction.

After all, Taylor approximation of  by  is valid only in a local neighborhood. Therefore, we can see that if is near the optimizer , then the Newton step  is a very good estimate of .

The **stopping criterion** for Newton’s method is set as follows

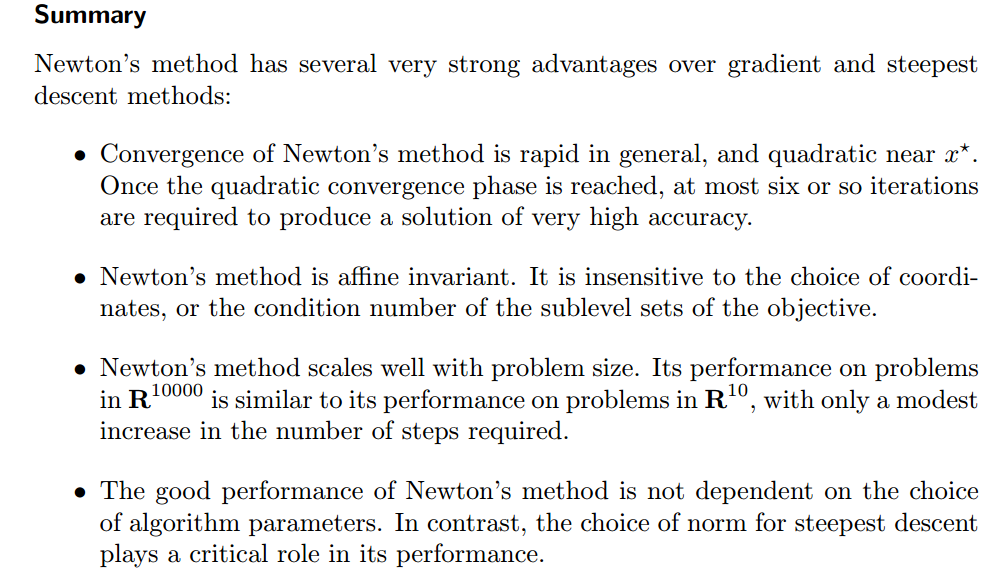


and

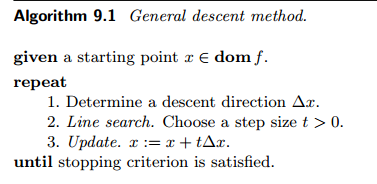


where represents the decrement of target function value during iterations and is a small positive number. Therefore, when the iteration can only lead to a very small decrement of the function value, we stop the searching.

Usually, Newton’s method is much faster than the gradient descent. The pros and cons of Newton’s method is summarized as follows[[1]](#footnote-1).



### Determine the step size: line search



In the above sections, we have chosen the step to be the gradient or the Newton step. However, how shall we choose the step size for each iteration? This is usually down through *line search*.

* Exact line search

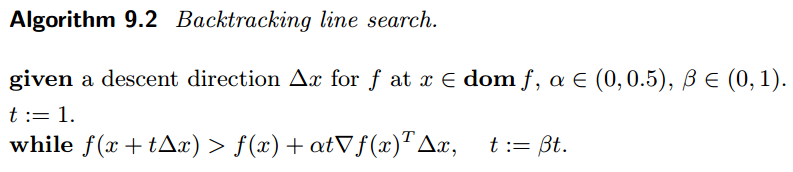
In this method, after we have fixed the step , the step size t is chosen to minimize f along the ray :

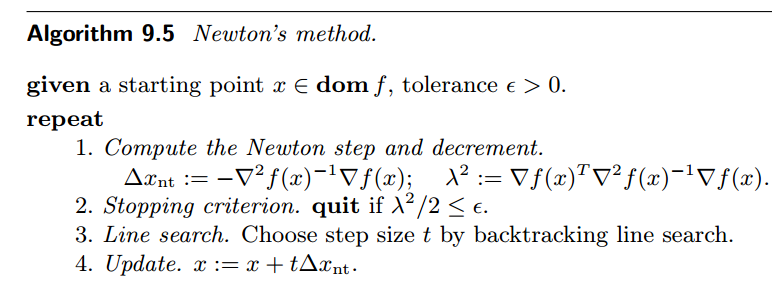


In practice, it is usually expensive to solve the above minimization problem. Therefore, the following *backtracking line search* is more widely used.

* Backtracking line search

Most line searches used in practice are inexact: the step length is chosen to approximately minimize *f* along the ray.





## L2 regularization

When the data in the training set are actually linearly separable, then it turns out the above ML estimation will lead to . To prevent w from exploding and overfitting, a regularization may be added. In this scenario, the negative log-likelihood in is turned into



where the regularization parameter is a tuning knob and  is defined in . In practice, there is no need to penalize the bias term, i.e., the first component of . Then, it is more convenient to write the regularization into a matrix form like



where , where *m* is the number of features. From equation , the gradient and Hessian in regularized version is



where the gradient  and Hessian  without regularization are given in . Now that we have and , it is straightforward to minimize  using the aforementioned descent algorithms.

## Loss function

Now we can consider a more general case. In the above logistic regression, we actually obtain a linear classifier by setting and use the logistic function to get the probability of . Then, the parameters are fitted by MLE.

More generally, we can of course use other kind of such as a nonlinear one. If we assign the two class labels as , then we have



Thus, in both cases, there exists



which is the most general case of logistic regression for binary classification. Correspondingly, the *log odds* is



From equation , the likelihood for *N s*amples in the training set is



Because we want to maximize the likelihood, equivalent to minimizing the negative log-likelihood, which is



Therefore, a natural choice of the loss function for logistic regression is



where is single input sample. Equation is called *log loss*.

Furthermore, here can be any form we like, and our aim is to minimize the negative log-likelihood (or call it the *cost function*) by fitting the parameters. Besides, the boosting methods like gradient boosting trees can also be adopted, where is a simple decision tree. Since in the logistic regression case, the log-odds is just . When , we can get , which suggests class +1 for this sample. Therefore, the sign of the link function is the predicted class label:



Note that if is increasingly positive in the loss function , the loss is further reduced. Therefore, this loss function considers the uncertainty in the predication: and both predicts a label +1, while the latter one is more confident about its predication with a smaller loss, assuming the true label is indeed +1.

### Binary classification with

If the two labels are 0 and 1, then we can form another loss function. Assume



Then for the two labels , the probability of the input vector belonging to class is



We want to maximize , which is equivalent to minimize its logarithm transformation



where is defined in . Therefore, the cost function is



which corresponds to the negative log-likelihood.

The multi-class version is in <https://www.kaggle.com/wiki/LogLoss>.

Therefore, for different labeling scheme, the log loss function has different forms. However, the loss function and are actually telling the same story: minimize the negative log-likelihood, i.e., maximize the likelihood.

Another point of view: since the logistic function always lies inside , we can also take it as the predication of the binary label , and our loss functions just measures how close the predicated   is to the true . On the other hand, we can also interpret the predication as the probability of being 1.

### Bayesian view

The *expected loss* of choosing estimation



because the true value of is unknown and is a random variable.

1. Boyd, Stephen, and Lieven Vandenberghe. Convex optimization. Cambridge university press, 2004. [↑](#footnote-ref-1)