1 Naive Bayes

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

$$p(c\,|\,\mathbf{x}) = \frac{p(c)p(\mathbf{x}\,|\,c)}{p(\mathbf{x})} \tag{1}$$

where $\mathbf{x} \in \Re^m$ and $c \in \{c_1, c_2, \cdots, c_K\}$, i.e., there are K classes in total. In Bayesian inference framework, we will assign the example into the class c^* which maximizes the posterior, that is

$$c^* = \arg\max_c p(c\,|\,\mathbf{x}) \tag{2}$$

In (1), the evidence $p(\mathbf{x}) = \sum_{c=1}^{K} p(\mathbf{x}, c)$ is independent of the class c. Therefore, we can transform the optimization problem (2) into

$$c^* = \arg\max_c p(c\,|\,\mathbf{x}) = \arg\max_c p(c)p(\mathbf{x}\,|\,c). \tag{3}$$

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

$$p(\mathbf{x}\,|\,c) = \prod_{i=1}^{m} p(x_i\,|\,c) \tag{4}$$

where x_i is the value of the i^{th} feature of one example $\mathbf{x}$. Now the classification result c^* of a NBC $h_{NB}(\cdot)$ is

$$c^* = h_{NB}(\mathbf{x}) = \arg\max_c p(c)\prod_{i=1}^{m} p(x_i\,|\,c). \tag{5}$$

The remaining problem is to determine the class prior $p(c)$ and the conditional distribution of each feature $p(x_i\,|\,c)$.

1.1 Estimation of $p(c)$

In this problem, there are only two classes, $c = 1$ for spam and or $c = 0$ for non-spam. Therefore, it is reasonable to assume a Bernoulli distribution as the class prior, i.e., $\pi_c = p(c) = Ber(\alpha)$, where α is probability of being a spam $p(c = 1)$. Since there are enough examples in the training set, for simplicity, we just use a maximum likelihood method to estimate the parameter α, which turns out to be

$$\alpha^* = \frac{N_s}{N_s + N_{ns}} \tag{6}$$

where N_s and N_{ns} are the number of spam and non-spam mails in the training set respectively.

1.2 Estimation of $p(x_i|c)$

Since our data has been binarized, each feature can have a value of 0 or 1. Supposing the probability of j^{th} feature in class c being 1 is μ_{jc}, i.e., $p(x_j = 1|c) = \mu_{jc}$, we know that each feature in each class in Bernoulli distributed: $p(x_j|c) = Ber(x_j|\mu_{jc})$. The remaining work is to infer the parameter value μ_{jc} with the given training set $D=\{\mathbf{x}_i, y_i\}_{i=1,2,\cdots N}$.

As the parameter μ_{jc} is only concerned with the class c, we extract a subset D_c from D with the examples labeled in class c, that is

$$D_c=\{(\mathbf{x}_i, y_i) \mid (\mathbf{x}_i, y_i)\in D, y_i=c\}_{i=1,2,\cdots,N, c=c_1,\cdots,c_K}. \tag{7}$$

In this case, we will take the Bayesian methods to infer the unknown parameter μ_{jc} with the Maximum A Posteriori estimation. In this framework, the unknown parameter will also be considered as a random variable with a prior probability $p(\mu_{jc}=v)$, where $v\in[0,1]$ because μ_{jc} is the parameter of the above Bernoulli distribution. The posterior of μ_{jc} given the training subset D_c is

$$p(\mu_{jc}\mid D_c)=\frac{p(\mu_{jc})p(D_c\mid\mu_{jc})}{p(D_c)}. \tag{8}$$

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

$$p(\mu_{jc}\mid a_{jc},b_{jc})=Beta(\mu_{jc}\mid a_{jc},b_{jc})=\frac{1}{B(a_{jc},b_{jc})}\mu_{jc}^{a_{jc}-1}(1-\mu_{jc})^{b_{jc}-1} \tag{9}$$

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 a_{jc} and b_{jc} according to our prior knowledge. Then, for $p(D_c\mid\mu_{jc})$ in (8), if the examples in D_c are independently identically distributed, then $p(D_c\mid\mu_{jc})$ is a binomial distribution with respect to the binary value of the j^{th} feature. We count the number of examples in D_c according to the j^{th} feature value as follows:

$$N_{jc,0}=\#\{\mathbf{x}\in D_c\mid x_j=0\},\quad N_{jc,1}=\#\{\mathbf{x}\in D_c\mid x_j=1\}. \tag{10}$$

Then, the likelihood $p(D_c\mid\mu_{jc})$ in (8) is

$$p(D_c\mid\mu_{jc})=\binom{N_c}{N_{jc,1}}\mu_{jc}^{N_{jc,1}}(1-\mu_{jc})^{N_{jc,0}}. \tag{11}$$

The MAP estimation from (8) is

$$\mu_{jc}^{*}=\arg\max_{\mu_{jc}} p(\mu_{jc}\mid D_c)=\arg\max_{\mu_{jc}} p(\mu_{jc})p(D_c\mid\mu_{jc}). \tag{12}$$

Applying a logarithmic transformation on (12) yields

$$\mu_{jc}^{*}=\arg\max_{\mu_{jc}}\log\left(p(\mu_{jc})p(D_c\mid\mu_{jc})\right)=\arg\max_{\mu_{jc}}\left(\log p(\mu_{jc})+\log p(D_c\mid\mu_{jc})\right). \tag{13}$$

In (9) the prior $p(\mu_{jc})$ is parameterized by a_{jc} and b_{jc} and the likelihood is given in (11). Now we can insert (9) and (11) into the optimization objective (13), which leads to

$$\begin{aligned}\log p(\mu_{jc})&=(a_{jc}-1)\log\mu_{jc}+(b_{jc}-1)\log(1-\mu_{jc})-\log B(a_{jc},b_{jc}),\\ \log p(D_c\mid\mu_{jc})&=N_{jc,1}\log\mu_{jc}+N_{jc,0}\log(1-\mu_{jc})+\log\binom{N_c}{N_{jc,1}}.\end{aligned} \tag{14}$$

Note that the last term of the above two equations in (14) are both constants which are independent of μ_{jc}. Therefore, combining (13) and (14), the MAP estimation is now turned into

$$\begin{aligned}\mu_{jc}^* &= \arg\max_{\mu_{jc}} \log\left(p(\mu_{jc})p(D_c \mid \mu_{jc})\right) \\ &= \arg\max_{\mu_{jc}} \left((a_{jc}-1+N_{jc,1})\log\mu_{jc} + (b_{jc}-1+N_{jc,0})\log(1-\mu_{jc})\right).\end{aligned} \tag{15}$$

The derivative of (15) with respect to μ_{jc} is

$$\frac{d}{d\mu_{jc}}\log\left(p(\mu_{jc})p(D_c \mid \mu_{jc})\right) = \frac{a_{jc}-1+N_{jc,1}}{\mu_{jc}} - \frac{b_{jc}-1+N_{jc,0}}{1-\mu_{jc}}. \tag{16}$$

By making the derivative (16) zero, we get the maximizer to be

$$\mu_{jc}^* = \frac{N_{jc,1}+a_{jc}-1}{N_{jc,0}+N_{jc,1}+a_{jc}+b_{jc}-2} = \frac{N_{jc,1}+a_{jc}-1}{N_c+a_{jc}+b_{jc}-2} \tag{17}$$

where N_c is size of the subset D_c.

In fact, from (9) and (11) we can also see that

$$p(\mu_{jc})p(D_c \mid \mu_{jc}) \propto \mu_{jc}^{N_{jc,1}+a_{jc}-1}(1-\mu_{jc})^{N_{jc,0}+b_{jc}-1} \tag{18}$$

which indicates the posterior of μ_{jc} 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 (18), the solution is the unique mode of Beta distribution, that is

$$\text{mode} = \frac{N_{jc,1}+a_{jc}-1}{N_{jc,1}+a_{jc}+N_{jc,0}+b_{jc}-2} \tag{19}$$

which just leads to the same answer as (17).

1.3 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 (5), now that we have got $p(c)$ and $p(x_j \mid c)$, given a test example $(\mathbf{x}, y)$, its class label is predicted by the Naive Bayesian Classifier to be

$$c^* = \arg\max_c p(c)\prod_{j=1}^{m} p(x_j \mid c) \tag{20}$$

which maximizes the posterior probability $p(y|\mathbf{x}; D)$. The class prior and the feature likelihood in each class are

$$\begin{aligned}&p(c) = Ber(c \mid \alpha^*) = c^{\alpha^*}(1-c)^{(1-\alpha^*)}, \\ &p(x_j \mid c) = Ber(x_j \mid \mu_{jc}^*) = x_j^{\mu_{jc}^*}(1-x_j)^{(1-\mu_{jc}^*)}, \\ &c, x_j \in \{0,1\}.\end{aligned} \tag{21}$$

In practice, to avoid numeric issues such as underflow, we compute the logarithmic transform of the objective function in (20) instead, that is,

$$c^* = \arg\max_c \left(\log p(c) + \sum_{j=1}^{m} p(x_j \mid c) \right) \tag{22}$$

2 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.,

$$p(x_j \mid c) = N\left(\mu_{jc}, \sigma_{jc}^2\right) = \frac{1}{\sqrt{2\pi\sigma_{jc}^2}} \exp\left(-\frac{(x_j - \mu_{jc})^2}{2\sigma_{jc}^2} \right) \tag{23}$$

We use the maximum likelihood method to estimate both the class prior $p(c)$ and the conditional feature distribution $p(x_j|c)$. The estimation of $p(c)$ is still shown in (6). Next, we will derive the MLE for Gaussian distribution.

2.1 MLE of Gaussian distribution

For notational simplicity, consider N scalar observations $\{x_1, \quad x_2, \quad \cdots, \quad x_N\}$ stemming from a Gaussian distribution $N(\mu, \sigma^2)$. Assume these observations are mutually independent. Then the log-likelihood function for the N observations is

$$L(\mu, \sigma^2) = \log \prod_{i=1}^{N} p(x_i) = \sum_{i=1}^{N} \log p(x_i). \tag{24}$$

According to $x_i \sim N(\mu, \sigma^2)$, we obtain

$$\log p(x_i) = \log \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2} \right) = -\frac{1}{2}\log 2\pi - \frac{1}{2}\log \sigma^2 - \frac{(x_i-\mu)^2}{2\sigma^2}. \tag{25}$$

Inserting (25) into (24) leads to

$$L(\mu, \sigma^2) = -\frac{N}{2}\log \sigma^2 - \sum_{i=1}^{N} \frac{(x_i-\mu)^2}{2\sigma^2} - \frac{N}{2}\log 2\pi, \tag{26}$$

whose derivative with respect to μ is

$$\frac{\partial L(\mu, \sigma^2)}{\partial \mu} = \sum_{i=1}^{N} \frac{x_i - \mu}{\sigma^2}. \tag{27}$$

Equating (27) to zero gives

$$\mu_{ML} = \frac{1}{N}\sum_{i=1}^{N} x_i. \tag{28}$$

Next, to facilitate the computation, let $s = \sigma^2$ and take the derivative of (26) with respect to s. We get

$$\frac{\partial}{\partial s} L(\mu, s) = \frac{1}{2s^2}\sum_{i=1}^{N}(x_i - \mu)^2 - \frac{N}{2s}. \tag{29}$$

By making $\frac{\partial}{\partial s}L(\mu, s) = 0$, we can obtain the maximum likelihood estimation of the variance is

$$\sigma_{ML}^2 = s = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2. \tag{30}$$

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 (23). The maximum likelihood estimation for the mean and variance parameters of the class dependent feature Gaussian distribution is given by

$$\begin{aligned} \mu_{jc}^* &= \frac{1}{N_c}\sum_{\mathbf{x}_i \in D_c}^{N} x_{ij} \\ \sigma_{jc}^{2\,*} &= \frac{1}{N_c}\sum_{\mathbf{x}_i \in D_c}^{N} (x_{ij} - \mu_{jc}^*)^2, \end{aligned} \tag{31}$$

where D_c is a subset of examples belonging to class c with $N_c = \#D_c$ and x_{ij} is the value of the j^{th} feature of $\mathbf{x}_i$. Now with the estimated distribution parameters (31), we can continue the Naive Bayes predication as shown in subsection 1.3.

3 Logistic Regression

In Bayesian classification, the posterior probability for class assignment $p(c \mid \mathbf{x})$ 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.

3.1 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

$$p(y = 1 \mid \mathbf{x}, \mathbf{w}) = f(\mathbf{w}^T\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^T\mathbf{x}}}, \tag{32}$$

where $f(t) = \frac{1}{1 + e^{-t}}$ is the *standard logistic function*. A bias term has been involved in the parameter vector $\mathbf{w} \in \Re^{m+1}$ as the first element w_0 and the sample input has been prepended with 1 to allow the bias term, that is, $\mathbf{x} \in \Re^{m+1}$. Here, m is the number of features. It is easy to write the posterior probability of an example x being in class 0:

$$p(y = 0 \mid \mathbf{x}, \mathbf{w}) = 1 - p(y = 1 \mid \mathbf{x}, \mathbf{w}) = \frac{e^{-\mathbf{w}^T\mathbf{x}}}{1 + e^{-\mathbf{w}^T\mathbf{x}}} = \frac{1}{1 + e^{\mathbf{w}^T\mathbf{x}}}. \tag{33}$$

We can also see that

$$\ln\frac{p(y = 1 \mid \mathbf{x}, \mathbf{w})}{p(y = 0 \mid \mathbf{x}, \mathbf{w})} = \mathbf{w}^T\mathbf{x}, \tag{34}$$

which is known as the *logistic odds*. Supposing there are only two classes, y = 0 and y = 1, the posterior probability for class predication of (32) and (33) can be integrated together as

$$\begin{aligned} p(y \mid \mathbf{x}, \mathbf{w}) &= p(y=1 \mid \mathbf{x}, \mathbf{w})^{y} p(y=0 \mid \mathbf{x}, \mathbf{w})^{1-y} \\ &= f(\mathbf{w}^T \mathbf{x})^{y} + (1 - f(\mathbf{w}^T \mathbf{x}))^{1-y}. \end{aligned} \tag{35}$$

3.2 Parameter estimation

In logistic regression, given a training set $D = \{\mathbf{x}_i, y_i\}_{i=1,2,\cdots N}$, the parameter vector **w** is estimated via the maximum likelihood method. In the next, for notational conciseness, the logistic function $f(\mathbf{w}^T\mathbf{x})$ will be notated by a simple *f*. Then, the logistic regression model in (35) becomes

$$p(y \mid \mathbf{x}, \mathbf{w}) = f^{y}(1-f)^{1-y}. \tag{36}$$

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

$$P(D \mid \mathbf{w}) = \prod_{i=1}^{N} f_i^{y_i}(1-f_i)^{1-y_i}, \tag{37}$$

where $f_i \triangleq f(\mathbf{w}^T\mathbf{x}_i)$. To facilitate the subsequent MLE, we consider the negative log-likelihood function given by

$$NL(\mathbf{w}) = -\ln P(D \mid \mathbf{w}) = -\sum_{i=1}^{N}\left(y_i \ln f_i + (1-y_i)\ln(1-f_i)\right). \tag{38}$$

The derivative of the standard logistic function $f(\cdot)$ is

$$\frac{df(t)}{d(t)} = f(t)(1-f(t)), \tag{39}$$

which implies

$$\frac{df(\mathbf{w}^T\mathbf{x})}{d\mathbf{w}} = f(\mathbf{w}^T\mathbf{x})\left(1 - f(\mathbf{w}^T\mathbf{x})\right)\mathbf{x}^T \Rightarrow \nabla f_i = f_i(1-f_i)\mathbf{x}_i \tag{40}$$

according to the chain rule. Now it is readily to write the gradient of $NL(\mathbf{w})$ with respect to **w** as follows

$$\begin{aligned} \nabla NL(\mathbf{w}) &= -\sum_{i=}^{N} \frac{y_i}{f_i}\nabla f_i - \frac{1-y_i}{1-f_i}\nabla f_i \\ &= -\sum_{i=1}^{N}\left(y_i(1-f_i)\mathbf{x}_i - (1-y_i)f_i\mathbf{x}_i\right) \\ &= \sum_{i=1}^{N}(f_i - y_i)\mathbf{x}_i. \end{aligned} \tag{41}$$

The Hessian matrix of $NL(\mathbf{w})$ with respect to **w** is

$$\begin{aligned}\nabla^2 NL(\mathbf{w}) &= \frac{d}{d\mathbf{w}}\nabla NL(\mathbf{w}) = \frac{d}{d\mathbf{w}}\sum_{i=1}^{N} f_i\mathbf{x}_i \\ &= \sum_{i=1}^{N}\mathbf{x}_i\frac{df_i}{d\mathbf{w}} = \sum_{i=1}^{N}\mathbf{x}_i f_i(1-f_i)\mathbf{x}_i^T \\ &= \sum_{i=1}^{N} f_i(1-f_i)\mathbf{x}_i\mathbf{x}_i^T.\end{aligned} \tag{42}$$

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

$$\begin{aligned}\nabla NL(\mathbf{w}) &= X^T(\mathbf{f}-\mathbf{y}) \\ \nabla^2 NL(\mathbf{w}) &= \mathrm{X}^T SX\end{aligned} \tag{43}$$

where

$$\begin{aligned}&X^T = [\mathbf{x}_1,\mathbf{x}_2,\cdots,\mathbf{x}_N], \quad \mathbf{f} = [f_1, f_2,\cdots, f_N]^T, \quad \mathbf{y} = [y_1, y_2,\cdots, y_N]^T, \\ &S = diag\{f_i(1-f_i)\}_{i=1,2,\cdots,N}.\end{aligned} \tag{44}$$

Similarly, using the above matrix notation, the negative log-likelihood function in (38) is rewritten into

$$NL(\mathbf{w}) = -\left(\mathbf{y}^T\ln\mathbf{f} + (\mathbf{1}-\mathbf{y})^T\ln(\mathbf{1}-\mathbf{f})\right). \tag{45}$$

To estimate the parameter vector $\mathbf{w}$, we will adopt the maximum likelihood estimation, that is, to minimize the negative log-likelihood $NL(\mathbf{w})$. It should be noted that since for any finite input t, the standard logistic function $f(x) \in (0,1)$, we can conclude from (43) that $NL(\mathbf{w})$ is positive definite. The proof is straightforward and shown in below.

$$f_i \in (-1,1) \Rightarrow S > 0 \Rightarrow \nabla^2 NL(\mathbf{w}) = \mathrm{X}^T SX > 0. \tag{46}$$

Now that $NL(\mathbf{w})$ is convex, we can safely the descent optimization methods such as the gradient descent or Newton' method to find the unique global minimum.

3.3 Optimization: gradient descent and Newton's method

A general unconstrained optimization problem can be stated as

$$\text{minimize } f(\boldsymbol{\theta}) \tag{47}$$

where $f: R^n \to R$ is a convex and twice continuously differentiable. A general descent method produces a minimizing sequence as following

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} + t^{(k)}\Delta\boldsymbol{\theta}^{(k)} \tag{48}$$

where $t^{(k)} \geq 0$ is the *step size* and $\Delta\boldsymbol{\theta}^{(k)}$ is the *step* or *search direction*.

By the Taylor expansion theorem, we know that

$$f(\boldsymbol{\theta}^{(k+1)}) \approx f(\boldsymbol{\theta}^{(k)}) + t^{(k)}\nabla f(\boldsymbol{\theta}^{(k)})^T\Delta\boldsymbol{\theta}^{(k)} \tag{49}$$

where we neglect the 2nd-order and other higher-order terms and $\nabla f(\boldsymbol{\theta}^{(k)})$ is the gradient. In all decent methods, we expect

$$f(\boldsymbol{\theta}^{(k+1)}) < f(\boldsymbol{\theta}^{(k)}) \tag{50}$$

except when $\boldsymbol{\theta}^{(k)}$ is already the minimizer in order to minimize the target function in an iterative manner. Combining (49) and (50), the search direction is required to satisfy

$$\nabla f(\boldsymbol{\theta}^{(k)})^T \Delta\boldsymbol{\theta}^{(k)} < 0 \tag{51}$$

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.

3.3.1 Gradient descent method

According to (51), a natural choice of the search direction is negative gradient, that is

$$\Delta\boldsymbol{\theta}^{(k)} = -\nabla f(\boldsymbol{\theta}^{(k)}). \tag{52}$$

The stopping criterion in the iteration of gradient descent approach is usually set to be $\left\|\nabla f(\boldsymbol{\theta}^{(k)})\right\|_2 < \eta$, where η is a small positive number since we know that the gradient will be zero exactly at the minimizer.

3.3.2 Newton's method

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

$$f(\boldsymbol{\theta}+\mathbf{v}) \approx \hat{f}(\boldsymbol{\theta}+\mathbf{v}) = f(\boldsymbol{\theta}) + \nabla f(\boldsymbol{\theta})^T \mathbf{v} + \frac{1}{2}\mathbf{v}^T \nabla^2 f(\boldsymbol{\theta})\mathbf{v} \tag{53}$$

If f is convex, i.e., $\nabla^2 f(\boldsymbol{\theta}) > 0$, then equation (53) is indeed a convex quadratic function of the step $\mathbf{v}$. To minimize $\hat{f}(\boldsymbol{\theta}+\mathbf{v})$, we can equal its gradient with respect **v** to be zero and we get

$$\mathbf{v}^* = -\nabla^2 f(\boldsymbol{\theta})^{-1}\nabla f(\boldsymbol{\theta}). \tag{54}$$

This is called a *Newton step* $\Delta\boldsymbol{\theta}_{nt} = \mathbf{v}^* = -\nabla^2 f(\boldsymbol{\theta})^{-1}\nabla f(\boldsymbol{\theta})$. Now consider the requirement of (51) for descent directions. The positive definiteness of $\nabla^2 f(\boldsymbol{\theta})$ implies that

$$\nabla f(\boldsymbol{\theta}^{(k)})^T \Delta\boldsymbol{\theta}^{(k)} = -\nabla f(\boldsymbol{\theta}^{(k)})^T \nabla^2 f(\boldsymbol{\theta}^{(k)})^{-1}\nabla f(\boldsymbol{\theta}^{(k)}) < 0 \tag{55}$$

unless $\nabla f(\boldsymbol{\theta}^{(k)})$ is zero. Therefore, the Newton step is a valid descent direction.

After all, Taylor approximation of $f(\boldsymbol{\theta}+\mathbf{v})$ by $\hat{f}(\boldsymbol{\theta}+\mathbf{v})$ is valid only in a local neighborhood. Therefore, we can see that if $\boldsymbol{\theta}$ is near the optimizer $\boldsymbol{\theta}^*$, then the Newton step $\boldsymbol{\theta}+\Delta\boldsymbol{\theta}_{nt}$ is a very good estimate of $\boldsymbol{\theta}^*$.

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

$$\lambda^2 = \nabla f(\boldsymbol{\theta})^T \Delta\boldsymbol{\theta} = \nabla f(\boldsymbol{\theta})^T \nabla^2 f(\boldsymbol{\theta})\nabla f(\boldsymbol{\theta}), \tag{56}$$

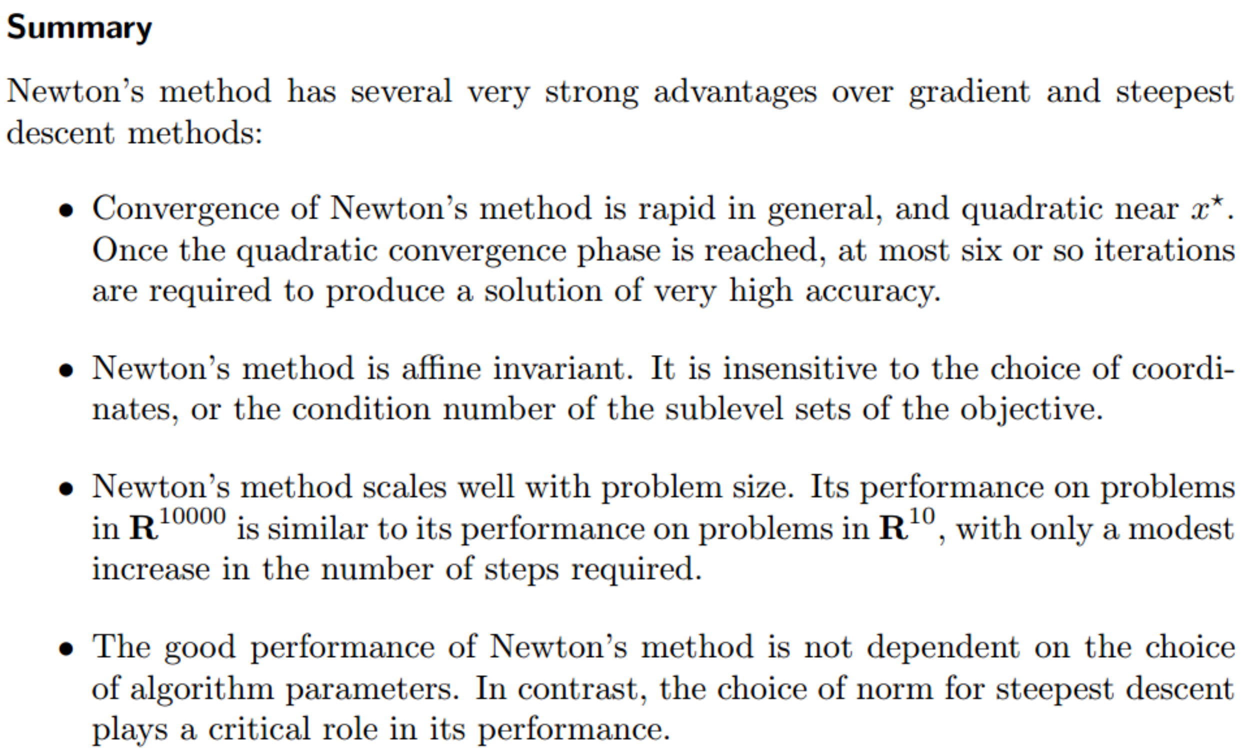
and

$$\lambda^2 < \varepsilon, \tag{57}$$

where λ^2 represents the decrement of target function value during iterations and $\epsilon > 0$ 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].

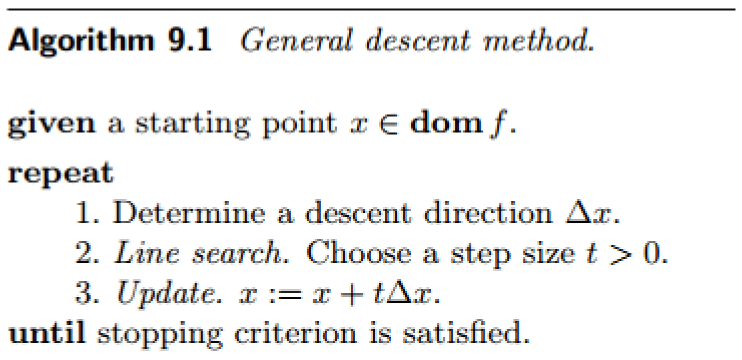


Summary

Newton's method has several very strong advantages over gradient and steepest descent methods:

- Convergence of Newton's method is rapid in general, and quadratic near $x^\star$. Once the quadratic convergence phase is reached, at most six or so iterations are required to produce a solution of very high accuracy.
- Newton's method is affine invariant. It is insensitive to the choice of coordinates, or the condition number of the sublevel sets of the objective.
- Newton's method scales well with problem size. Its performance on problems in $\mathbf{R}^{10000}$ is similar to its performance on problems in $\mathbf{R}^{10}$, with only a modest increase in the number of steps required.
- The good performance of Newton's method is not dependent on the choice of algorithm parameters. In contrast, the choice of norm for steepest descent plays a critical role in its performance.

3.3.3 Determine the step size: line search



Algorithm 9.1 *General descent method.*

```
given a starting point $x \in \mathbf{dom}\, f$.
repeat
    1. Determine a descent direction $\Delta x$.
    2. Line search. Choose a step size $t > 0$.
    3. Update. $x := x + t\Delta x$.
until stopping criterion is satisfied.
```

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 t for each iteration? This is usually down through *line search*.

- Exact line search

In this method, after we have fixed the step Δx, the step size t is chosen to minimize f along the ray $\{x + t\Delta x\}$:

$$t = \arg\min_{s\geq 0} f(x + s\Delta x). \qquad (58)$$

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

[1] Boyd, Stephen, and Lieven Vandenberghe. Convex optimization. Cambridge university press, 2004.

minimize f along the ray.

Algorithm 9.2 *Backtracking line search.*

```
given a descent direction $\Delta x$ for $f$ at $x \in \mathbf{dom}\, f$, $\alpha \in (0, 0.5)$, $\beta \in (0, 1)$.
$t := 1$.
while $f(x + t\Delta x) > f(x) + \alpha t \nabla f(x)^T \Delta x$,    $t := \beta t$.
```

Algorithm 9.5 *Newton's method.*

```
given a starting point $x \in \mathbf{dom}\, f$, tolerance $\epsilon > 0$.
repeat
    1. Compute the Newton step and decrement.
        $\Delta x_{\mathrm{nt}} := -\nabla^2 f(x)^{-1} \nabla f(x)$;    $\lambda^2 := \nabla f(x)^T \nabla^2 f(x)^{-1} \nabla f(x)$.
    2. Stopping criterion. quit if $\lambda^2/2 \le \epsilon$.
    3. Line search. Choose step size $t$ by backtracking line search.
    4. Update. $x := x + t\Delta x_{\mathrm{nt}}$.
```

3.4 L2 regularization

When the data in the training set are actually linearly separable, then it turns out the above ML estimation will lead to $||\mathrm{w}|| \rightarrow \infty$. To prevent w from exploding and overfitting, a l_2 regularization may be added. In this scenario, the negative log-likelihood in (38) is turned into

$$NL_2(\mathbf{w}) = NL(\mathbf{w}) + \frac{1}{2}\lambda \mathbf{w}^T \mathbf{w}, \tag{59}$$

where the regularization parameter $\lambda \ge 0$ is a tuning knob and $NL(\mathbf{w})$ is defined in (45). In practice, there is no need to penalize the bias term, i.e., the first component w_0 of $\mathbf{w}$. Then, it is more convenient to write the regularization into a matrix form like

$$NL_2(\mathbf{w}) = NL(\mathbf{w}) + \frac{1}{2}\mathbf{w}^T \Lambda \mathbf{w}, \tag{60}$$

where $\Lambda = diag\{0, \quad \lambda, \quad \cdots, \quad \lambda\} \in \Re^{m+1}$, where m is the number of features. From equation (60), the gradient and Hessian in l_2 regularized version is

$$\begin{aligned} \nabla NL_2(\mathbf{w}) &= \nabla NL(\mathbf{w}) + \Lambda \mathbf{w}, \\ \nabla^2 NL_2(\mathbf{w}) &= \nabla^2 NL(\mathbf{w}) + \Lambda, \end{aligned} \tag{61}$$

where the gradient $\nabla NL(\mathbf{w})$ and Hessian $\nabla^2 NL(\mathbf{w})$ without regularization are given in (43). Now that we have (60) and (61), it is straightforward to minimize $NL_2(\mathbf{w})$ using the aforementioned descent algorithms.