CAMBRIDGE
Cambridge University Press & Assessment
978-1-316-51950-9 — Introduction to Machine Learning
Ruye Wang
Excerpt
More Information

Part I

Mathematical Foundations

Many machine learning (ML) problems can be formulated as an optimization problem to build a model of a set of observed data, called *training dataset* or simply *training set*, so that the model output matches the data in an optimal manner in a certain sense. Typically this is done by either minimizing or maximizing an *objective function* of a set of model parameters, which measures the goodness of the model. For example, in either regression or classification, the goal of a certain type of learning algorithms is to minimize the error of the model, the difference between the model prediction and the given observed data, measured in a certain way, such as the squared error. In other types of algorithms, the goal is to maximize the *likelihood* of the model parameters, defined as the probability of observing the given data conditioned on these parameters.

In the typical *supervised learning* paradigm, the modeling process can be formulated so as to construct a function $y = f(\mathbf{x},\boldsymbol{\theta})$ of some d variables as the components of a vector $\mathbf{x} = [x_1,\cdots,x_d]^T$, parameterized by a set of M parameters as the components of a vector $\boldsymbol{\theta} = [\theta_1,\cdots,\theta_M]^T$. The goal is for this function to best fit a set of N data points in the training set, denoted by $\mathcal{D} = \{(\mathbf{x}_n, y_n), \quad n = 1,\cdots,N\}$. While the form of the model function is usually assumed to be known based on some prior knowledge, such as a polynomial (e.g., linear or quadratic) function, or certain *probability density distribution (pdf)* of the data, the values of the M parameters in $\boldsymbol{\theta}$ are unknown and to be estimated based on the training set $\mathcal{D}$. The estimation can be carried out by various methods such as the following two that are widely used in ML.

- *Least squares estimation (LSE)*
  The sum of squared error (SSE) between the model output and the desired output according to the training set is to be minimized

$$J_{sse}(\boldsymbol{\theta}) = \sum_{n=1}^{N}[y_n - f((\mathbf{x}_n,\boldsymbol{\theta}))]^2 \tag{0.1}$$

- *Maximum likelihood estimate (MLE)*
  The likelihood of the parameter $\boldsymbol{\theta}$, the conditional probability of observing the data in $\mathcal{D}$ given a specific parameter $\boldsymbol{\theta}$, is to be maximized

$$J_{mle}(\boldsymbol{\theta}) = L(\boldsymbol{\theta}|\mathcal{D}) \propto p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{n=1}^{N} p(y_n|\mathbf{x}_n,\boldsymbol{\theta}) \tag{0.2}$$

© in this web service Cambridge University Press & Assessment www.cambridge.org

CAMBRIDGE
Cambridge University Press & Assessment
978-1-316-51950-9 — Introduction to Machine Learning
Ruye Wang
Excerpt
More Information

Solving either the minimization or maximization problem we obtain the optimal model parameter $\boldsymbol{\theta}^*$ for the model to best fit the observed data.

To prepare for the discussions of various ML algorithms in future parts of this book, such as how to find specifically the optimal $\boldsymbol{\theta}^*$ that minimizes/maximizes function $J(\boldsymbol{\theta})$ just discussed, in this part we will consider a set of typical methods for the optimization problem in the most generic form, and find the optimal solution $\mathbf{x}^* = [x_1^2, \cdots, x_N^*]^T$ that either maximizes or minimizes a real valued multivariable function $y = f(\mathbf{x}) = f(x_1, \cdots, x_N)$. As maximizing $f(\mathbf{x})$ is equivalent to minimizing $-f(\mathbf{x})$, we typically only need to consider the minimization problem to find the optimal solution $\mathbf{x}^*$ that minimizes $f(\mathbf{x})$, denoted by

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} f(\mathbf{x}), \qquad \text{i.e.,} \qquad f(\mathbf{x}^*) = \min_{\mathbf{x}} f(\mathbf{x}) \le f(\mathbf{x}) \tag{0.3}$$

Here the function $f(\mathbf{x})$ can be either linear or nonlinear, and an optimization problem is either *unconstrained* if the independent variable $\mathbf{x}$ of the function is allowed to take any value in the function domain $\mathbb{R}^N$, or *constrained* if $\mathbf{x}$ must be inside a subset of $\mathbb{R}^N$, called the *feasible region*. In ML, both linear and nonlinear methods, either constrained or not, are widely used. For example, the linear least squares method (LLS) for regression (to minimize the squared error such as that given in Eq. (0.1), Section 5.1) is solved as a linear unconstrained optimization problem, whereas the method of support vector machine (SVM) for classification (to maximize the margin between two classes, Section 14.1) is solved as a nonlinear constrained optimization problem.

In the following, we will review some necessary mathematical topics as the foundation for many of the learning methods to be discussed in future parts, including the numerical methods for solving equations in Chapter 1, which are not only important in their own right, but also most closely related to both unconstrained and constrained optimization problems in Chapters 2 and 3, which in turn will be needed for many important ML algorithms to be discussed in the upcoming chapters.

© in this web service Cambridge University Press & Assessment www.cambridge.org

CAMBRIDGE
Cambridge University Press & Assessment
978-1-316-51950-9 — Introduction to Machine Learning
Ruye Wang
Excerpt
More Information

1 Solving Equations

In this chapter, we consider some basic methods for solving an equation system in the most general form of M simultaneous equations of N variables

$$\begin{cases} f_1(x_1,\cdots,x_N) = f_1(\mathbf{x}) = 0 \\ \cdots\cdots\cdots \\ f_M(x_1,\cdots,x_N) = f_M(\mathbf{x}) = 0 \end{cases} \tag{1.1}$$

Here $\mathbf{x} = [x_1,\cdots,x_N]^T \in \mathbb{R}^N$ is an N–D column vector representing a point in the N–D vector space $\mathbb{R}^N$ spanned by $\{x_1,\cdots,x_N\}$, and $yf_m(\mathbf{x})$ $(m = 1,\cdots,M)$ is a scalar valued multivariate function that maps any point in its domain $\mathbf{x} \in \mathbb{R}^N$ to a scalar value y. This equation system can be represented more concisely in vector form

$$\mathbf{f}(\mathbf{x}) = \mathbf{0} \tag{1.2}$$

where $\mathbf{f} = [f_1,\cdots,f_M]^T$ is a column vector containing the M functions as components and $\mathbf{0} = [0,\cdots,0]^T$ is a vector containing M zeros. Solving this equation system means to finding a solution $\mathbf{x}^* \in \mathbb{R}^N$ that satisfies all M equations. If such a solution does not exist, we would still like to find an approximate solution by which certain error (e.g., the sum of squared error (SSE) $||\mathbf{f}(\mathbf{x})||^2$), is minimized.

Geometrically, a function $y = f(\mathbf{x})$ defined over domain $\mathbb{R}^N$ is a *hypersurface* in an $N+1$ dimensional space, of which the $N+1$st dimension represents the function value $y = f(\mathbf{x})$ corresponding to a point $\mathbf{x} \in \mathbb{R}^N$. Specially the hypersurface is a curve if $N = 1$ or a surface if $N = 2$. The *roots* or *zeros* of function $f(\mathbf{x})$, that is, the solutions of equation $f(\mathbf{x}) = 0$, are composed of all points on the intersection, if it exists, of the hypersurface $y = f(\mathbf{x})$ and the hyperplane $y = 0$.

For example, in the special case of $M = N = 2$, functions $f_1(x_1,x_2)$ and $f_2(x_1,x_2)$ are two surfaces defined over the 2-D space spanned by x_1 and x_2, and the roots of each of the two functions are on a curve as the intersection of the corresponding surface and the 2-D space. The solutions of the system composed of the two equations $f_1 = 0$ and $f_2 = 0$ are the intersection of these two curves in the 2-D space if they do intersect, otherwise no solution exists.

© in this web service Cambridge University Press & Assessment www.cambridge.org

CAMBRIDGE
Cambridge University Press & Assessment
978-1-316-51950-9 — Introduction to Machine Learning
Ruye Wang
Excerpt
More Information

Example 1.1

Consider a simultaneous equation system with $M = N = 2$

$$\begin{cases} f_1(x_1, x_2) = x_1^2 + x_2^2 - 2 = 0 \\ f_2(x_1, x_2) = x_1 + x_2 - C = 0 \end{cases}$$

The first function $f_1(x_1, x_2)$ is a parabolic cone in the $N + 1 = 3$ dimensional space centrally symmetric to the vertical axis, and its roots form a circle $x_1^2 + x_2^2 = 2$ on the 2-D plane spanned by x_1 and x_2 centered at the origin (0, 0) with radius 1; the second function $f_2(x_1, x_2)$ is a plane in the 3-D space through the origin, and its roots form a straight line $x_2 = C - x_1$ on the 2-D plane. The solutions of the equation system are where the two curves intersect:

- if $C = 0$, there are two solutions (1, −1) and (−1, 1);
- if $C = 2$, there is only one solution (1, 1);
- if $C = 3$, the two curves do not intersect, that is, no solution exists.

1.1 Linear Equation Systems

Consider a linear equation system of M equations and N variables

$$\begin{cases} f_1(\mathbf{x}) & = f_1(x_1, \cdots, x_N) & = \sum_{j=1}^{N} a_{1j} x_j - b_1 = 0 \\ \cdots & \cdots & \cdots \\ f_M(\mathbf{x}) & = f_1(x_1, \cdots, x_N) & = \sum_{j=1}^{N} a_{Mj} x_j - b_M = 0 \end{cases} \tag{1.3}$$

which can be expressed in vector form

$$\mathbf{f}(\mathbf{x}) = \mathbf{A}\mathbf{x} - \mathbf{b} = \mathbf{0} \tag{1.4}$$

where

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} f_1(\mathbf{x}) \\ \vdots \\ f_M(\mathbf{x}) \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} a_{11} & \cdots & a_{1N} \\ \vdots & \ddots & \vdots \\ a_{M1} & \cdots & a_{MN} \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_1 \\ \vdots \\ b_M \end{bmatrix} \tag{1.5}$$

In general, the existence and uniqueness of the solution can be determined based on the *fundamental theorem of linear algebra* (A.349 in Section A.5), depending on the rank R of the coefficient matrix $\mathbf{A}$, and whether the system is underdetermined (underconstrained) if $M < N$, or overdetermined (overconstrained) if $M > N$.

If $\mathbf{A}$ is a full rank square matrix with $R = M = N$, then its inverse $\mathbf{A}^{-1}$ exists and the system has a unique solution $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$. But if $M > N = R$ and $\mathbf{b}$ is not in the column space of $\mathbf{A}$, the system is overconstrained with no solutions. In this case we can still find an optimal approximate solution that minimizes the SSE, called *objective function* in general optimization problems, defined as

© in this web service Cambridge University Press & Assessment www.cambridge.org

CAMBRIDGE
Cambridge University Press & Assessment
978-1-316-51950-9 — Introduction to Machine Learning
Ruye Wang
Excerpt
More Information

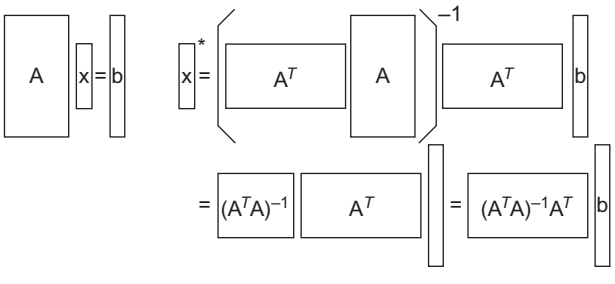

Fig. 1.1 The pseudo inverse (Section 1.1).

$$\varepsilon(\mathbf{x}) = \frac{1}{2}||\mathbf{r}||^2 = \frac{1}{2}\mathbf{r}^T\mathbf{r} = \frac{1}{2}(\mathbf{b}-\mathbf{A}\mathbf{x})^T(\mathbf{b}-\mathbf{A}\mathbf{x})$$
$$= \frac{1}{2}\left(\mathbf{b}^T\mathbf{b} - \mathbf{x}^T\mathbf{A}^T\mathbf{b} - \mathbf{b}^T\mathbf{A}\mathbf{x} + \mathbf{x}^T\mathbf{A}^T\mathbf{A}\mathbf{x}\right) \tag{1.6}$$

where $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}$ is the *residual* of the system, and $||\mathbf{r}||^2 = \sum_{m=1}^{M} r_m^2$ (thereby the term sum of squares error). The optimal solution $\mathbf{x}^*$ that minimizes $\varepsilon(\mathbf{x})$ can be found by setting its derivative with respect to $\mathbf{x}$ to zero (see Section A.4.3 for differentiation with respect to a vector variable)

$$\frac{d}{d\mathbf{x}}\varepsilon(\mathbf{x}) = \frac{1}{2}\frac{d}{d\mathbf{x}}||\mathbf{r}||^2 = \frac{1}{2}\frac{d}{d\mathbf{x}}\left(\mathbf{b}^T\mathbf{b} - \mathbf{x}^T\mathbf{A}^T\mathbf{b} - \mathbf{b}^T\mathbf{A}\mathbf{x} + \mathbf{x}^T\mathbf{A}^T\mathbf{A}\mathbf{x}\right)$$
$$= -\mathbf{A}^T\mathbf{b} + \mathbf{A}^T\mathbf{A}\mathbf{x} = \mathbf{0} \tag{1.7}$$

Solving this matrix equation we get the optimal approximate solution, as illustrated in Fig. 1.1:

$$\mathbf{x}^* = \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{b} = \mathbf{A}^-\mathbf{b} \tag{1.8}$$

where $\mathbf{A}^- = \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T$ is the *pseudo-inverse* (see Section A.6.1) of the non-square matrix $\mathbf{A}$. Here we assume the M equations are independent, that is, the $N \times N$ square matrix $\mathbf{A}^T\mathbf{A}$ has a full rank $R = N$ (i.e., it is nonsingular and invertible with all eigenvalues being non-zero).

If the M equations are barely independent, that is, some eigenvalues of $\mathbf{A}^T\mathbf{A}$ are very close to zero, then it is near-singular but still invertible, and some eigenvalues of its inverse $(\mathbf{A}^T\mathbf{A})^{-1}$ may take huge values. Consequently the result in Eq. (1.8) may vary greatly due to some small random fluctuation in either $\mathbf{A}$ or $\mathbf{b}$. In this case, the system is said to be *ill-conditioned* or *ill-posed*, as it is prone to noise, that is, its solution is highly sensitive to noise and therefore unreliable and unstable.

Such an ill-posed problem can be addressed by *regularization*, so that the solution $\mathbf{x}$ is forced to avoid taking huge values and is therefore less sensitive to noise. This can be done by including in the objective function an additional penalty term for large values of $\mathbf{x}$ scaled by a parameter λ

$$J(\mathbf{x}) = \frac{1}{2}||\mathbf{r}||^2 + \frac{\lambda}{2}||\mathbf{x}||^2 \tag{1.9}$$

By minimizing $J(\mathbf{x})$, we can obtain a solution $\mathbf{x}$ of small norm $||\mathbf{x}||$ as well as a small error $\varepsilon(\mathbf{x})$. Just as before, the solution $\mathbf{x}$ can be obtained by setting the derivative of $J(\mathbf{x})$ to zero

© in this web service Cambridge University Press & Assessment www.cambridge.org

CAMBRIDGE
Cambridge University Press & Assessment
978-1-316-51950-9 — Introduction to Machine Learning
Ruye Wang
Excerpt
More Information

$$\frac{d}{d\mathbf{x}}J(\mathbf{x}) = -\mathbf{A}^T\mathbf{b} + \mathbf{A}^T\mathbf{A}\mathbf{x} + \lambda\mathbf{x} = \mathbf{0} \tag{1.10}$$

and solving the resulting equation to obtain

$$\mathbf{x}^* = (\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})^{-1}\mathbf{A}^T\mathbf{b} \tag{1.11}$$

We see that even if $\mathbf{A}^T\mathbf{A}$ is near singular, matrix $\mathbf{A}^T\mathbf{A}+\lambda\mathbf{I}$ is not due to the additional term $\lambda\mathbf{I}$.

Here λ is called the *hyperparameter*. By adjusting it we can make a proper tradeoff between accuracy and stability.

- Small λ: the solution is more accurate but less stable as it is more prone to noise, that is, the variance error may be large. This is called *overfitting*.
- Large λ: the solution is less accurate but more stable as it is less affected by noise. This is called *underfitting*.

The issue of overfitting versus underfitting is of essential importance in machine learning (ML) in general; it will appear repeatedly and be specifically addressed while discussing various regression and classification algorithms in the later chapters.

1.2 The Bisection and Secant Methods

Most equation systems of interest are nonlinear. Unlike the linear equation systems, which can be solved under the guidance of the fundamental theorem of linear algebra, for nonlinear equation systems, there exists neither a theory regarding the existence and uniqueness of their solutions, nor closed-form solutions in general. We therefore have to rely on numerical methods to find one or more solutions, if they do exist, in an iterative manner from certain initial guess of the solution, if the iteration converges, that is, the difference between two consecutive results eventually approach to zero.

We first consider some methods that find a solution of a single-variable nonlinear equation $f(x) = 0$, by searching iteratively through a neighborhood of the domain, in which a solution is known to be located.

1.2.1 The Bisection Search

This method requires two initial guesses $x_0 < x_1$ satisfying $f(x_0)f(x_1) < 0$. As $f(x_0)$ and $f(x_1)$ are on opposite sides of the x-axis $y = 0$, the solution x^* at which $f(x^*) = 0$ must reside somewhere in between these two guesses, that is, $x_0 < x^* < x_1$.

Given such two end points x_0 and x_1, we could find any point x_2 in between and use it to replace one of the end points at which the function value $f(x)$ has the same sign as that of $f(x_2)$. The new search interval is $a = x_2 - x_0$ if $f(x_2)f(x_1) > 0$, or $b = x_1 - x_2$ if $f(x_0)f(x_2) > 0$, either of which is smaller than the previous interval

© in this web service Cambridge University Press & Assessment www.cambridge.org

CAMBRIDGE
Cambridge University Press & Assessment
978-1-316-51950-9 — Introduction to Machine Learning
Ruye Wang
Excerpt
More Information

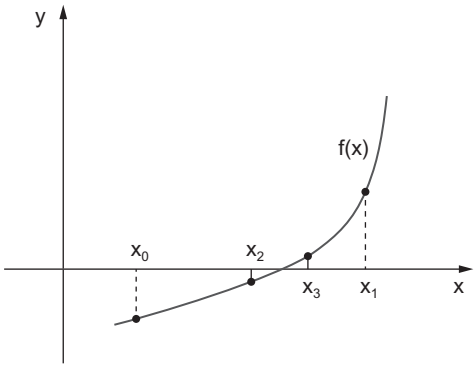

Fig. 1.2 Bisection search.

$a + b = x_1 - x_0$. This process is then carried out iteratively until the solution is eventually approached, with a guaranteed convergence.

To avoid the worst possible case in which the solution always happens to be in the larger of the two sections a and b, we typically choose x_2 to be the middle point $x_2 = (x_0 + x_1)/2$ between the two end points, so that $a = b = (x_1 - x_0)/2$, that is, the new search interval is always halved in each step of the iteration

$$x_{n+1} = \frac{x_n + x_{n-1}}{2} \tag{1.12}$$

as illustrated in Fig. 1.2. Here, without loss of generality, we have assumed $f(x_{n-1})f(x_{n+1}) > 0$ and x_{n-1} is replaced by x_{n+1}.

For example, if we assume that the root is located at $x^* = 2.2$ between the two initial values $x_0 = 0$ and $x_1 = 8$, then we get

Table 1.1

n	0	1	2	3	4	5	6	7	8
x_n	0	8	4	2	3	2.5	2.25	2.125	⋯
$\|e_n\|$	2.2	5.8	1.8	0.2	0.8	0.3	0.05	0.075	⋯

Note that the error $|e_n| = |x_n - x^*|$ does not necessarily always reduce monotonically. However, as in each iteration the search interval is always halved, the worst possible error is also halved

$$|e_{n+1}| = |x_{n+1} - x^*| \le \frac{|x_n - x_{n-1}|}{2} = \frac{|x_0 - x_1|}{2^n} \approx \frac{|e_n|}{2} \tag{1.13}$$

In general, to measure how quickly or slowly an iterative method converges, we consider the following limit of the ratio of errors of two consecutive iterations

$$\lim_{n\to\infty} \frac{|e_{n+1}|}{|e_n|^q} = \lim_{n\to\infty} \frac{|e_{n+1}|}{|e_n|} \le \mu \tag{1.14}$$

where q is the *order of convergence* and μ is the *rate of convergence*. The higher the order q, the more rapidly the iteration converges. For the bisection method we have

$$\lim_{n\to\infty} \frac{|e_{n+1}|}{|e_n|^q} = \lim_{n\to\infty} \frac{|e_{n+1}|}{|e_n|} \le \mu = \frac{1}{2} \tag{1.15}$$

© in this web service Cambridge University Press & Assessment www.cambridge.org

CAMBRIDGE
Cambridge University Press & Assessment
978-1-316-51950-9 — Introduction to Machine Learning
Ruye Wang
Excerpt
More Information

We see that its order of convergence is $q = 1$ (it converges linearly) with the rate of convergence $\mu = 1/2$. Compared to other methods to be considered later, the linear convergence of the bisection method is rather slow, but it has the advantage that no derivative $f'(x)$ of the given function is needed and the given function does not need to be differentiable.

1.2.2 The Secant Method

As in the bisection method, we assume there are two initial values x_0 and x_1 available, but they no longer have to satisfy $f(x_0)f(x_1) < 0$. Here the iteration is based on the zero-crossing of the secant line passing through the two points $f(x_0)$ and $f(x_1)$, instead of the middle point between them. The equation for the secant is

$$y = f(x_0) + \frac{x - x_0}{x_1 - x_0}(f(x_1) - f(x_0)) = f(x_0) + \hat{f}'(x_0)(x - x_0) \tag{1.16}$$

where $\hat{f}'(x_0)$ is the finite difference that approximates the derivative $f'(x_0)$

$$\hat{f}'(x_0) = \frac{f(x_1) - f(x_0)}{x_1 - x_0} = \frac{\Delta f(x_0)}{\Delta x_0} \overset{\Delta x_0 \to 0}{\Longrightarrow} f'(x_0) \tag{1.17}$$

Setting $y = 0$ in Eq. (1.17), we obtain the zero crossing point

$$x = x_0 - \frac{x_1 - x_0}{f(x_1) - f(x_0)} f(x_0) = x_0 - \frac{f(x_0)}{\hat{f}'(x_0)} \tag{1.18}$$

As shown in Fig. 1.3, the new value x just obtained is a better estimate of the root than either x_0 or x_1, and is used to replace one of them.

- If $f(x_0)f(x_1) < 0$, then x_i $(i = 0, 1)$ satisfying $f(x_i)f(x) > 0$ is replaced by $x_2 = x$ (same as the bisection method).
- If $f(x_0)f(x_1) > 0$, then x_i $(i = 0, 1)$ with greater $|f(x_i)|$ is replaced by $x_2 = x$.

This process is then carried out iteratively to approach the root

$$x_{n+1} = x_n - \frac{f(x_n)}{\hat{f}'(x_n)} \tag{1.19}$$

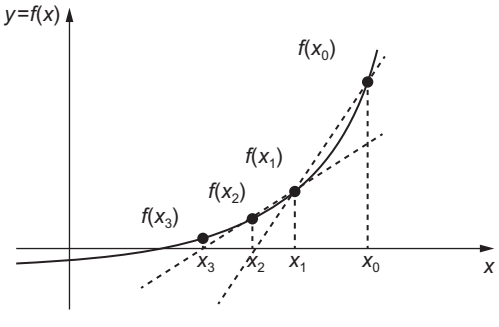

Fig. 1.3 The secant method.

© in this web service Cambridge University Press & Assessment www.cambridge.org

CAMBRIDGE
Cambridge University Press & Assessment
978-1-316-51950-9 — Introduction to Machine Learning
Ruye Wang
Excerpt
More Information

If the derivative $f'(x)$ is available, it can be used to replace its approximation $\hat{f}'(x_n)$ and the iteration becomes the Newton–Raphson method (Section 1.4)

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{1.20}$$

In case $f(x_n) = f(x_{n-1})$, the approximated derivative is zero $\hat{f}'(x_n) = 0$, and x_{n+1} cannot be found. In such a case, a root may exist between x_n and x_{n-1}. Therefore the way to resolve this problem is to combine the bisection search with the secant method so that when $\hat{f}' = 0$, the algorithm will switch to a bisection search. This is Dekker's method

$$x_{n+1} = \begin{cases} x_n - f(x_n)/\hat{f}'(x_n) & \text{if } f(x_n) \neq f(x_{n-1}) \\ (x_n + x_{n-1})/2 & \text{otherwise} \end{cases} \tag{1.21}$$

The order of convergence of the secant method is $q = 1.618$ as shown in Appendix 1 of this chapter, that is, the secant method converges more quickly than the bisection search method with $q = 1$ (which does not take advantage of the information of the specific function $f(x)$).

1.3 Fixed-Point Iteration

Fixed-point iteration is a method of finding the fixed point of a given function. It can be used to solve the generic equation system $\mathbf{f}(\mathbf{x}) = \mathbf{0}$, as seen in this section. In particular, it will be used to solve the Bellman equation in dynamic programming, as we will see in Chapter 20.

We first consider the special case of solving a single-variable nonlinear equation $f(x) = 0$ ($M = N = 1$). To find a solution x^* that satisfies the equation, we could first convert it into an equivalent equation $g(x) = x$, so that an x satisfying one of the equations will also satisfy the other. We then carry out an iteration $x_{n+1} = g(x_n)$ from some initial value x_0. If the iteration converges at a point x^*, that is, $g(x^*) = x^*$, then we also have $f(x^*) = 0$ (i.e., x^* is a solution of the equation $f(x) = 0$). Consider the following two examples.

Example 1.2

$$f(x) = \log(x) - 0.5 = 0$$

To solve the equation, we construct another function

$$g_1(x) = x - f(x) = x - \log(x) + 0.5$$

so that the equation $g(x) = x$ is equivalent to the given equation $f(x) = 0$ (left Fig. 1.4). We then carry out the iteration $x_{n+1} = g(x_n)$ from an initial value, such as $x_0 = 0.5$, and obtain

© in this web service Cambridge University Press & Assessment www.cambridge.org

CAMBRIDGE
Cambridge University Press & Assessment
978-1-316-51950-9 — Introduction to Machine Learning
Ruye Wang
Excerpt
More Information

Table 1.2

n	x_n	$g(x_n)$	$f(x_n)$
0	0.50000	1.69315	−1.19315
1	1.69315	1.66656	0.02659
2	1.66656	1.65580	0.01076
3	1.65580	1.65152	0.00428
4	1.65152	1.64982	0.00169
5	1.64982	1.64915	0.00067
6	1.64916	1.64889	0.00026
7	1.64889	1.64879	0.00010
8	1.64879	1.64875	0.00004
9	1.64875	1.64873	0.00002
10	1.64873	1.64873	0.00001
11	1.64873	1.64872	0.00000
12	1.64872	1.64872	0.00000

We see that the iteration converges to $x^* = \lim_{n\to\infty} g(x_n) = 1.64872$ satisfying $g(x^*) = x^* - f(x^*) = x^*$, and, equivalently, x^* is also the solution of the given equation $f(x) = 0.5 - \log(x) = 0$

$$0.5 - \log(1.64872) = 0, \qquad \text{i.e.,} \qquad e^{0.5} = \sqrt{e} = 1.64872$$

Alternatively, we could construct a different function

$$g'(x) = x + f(x) = x + \log(x) - 0.5$$

also equivalent to $f(x) = 0$ (right Fig. 1.4). But this iteration no longer converges!

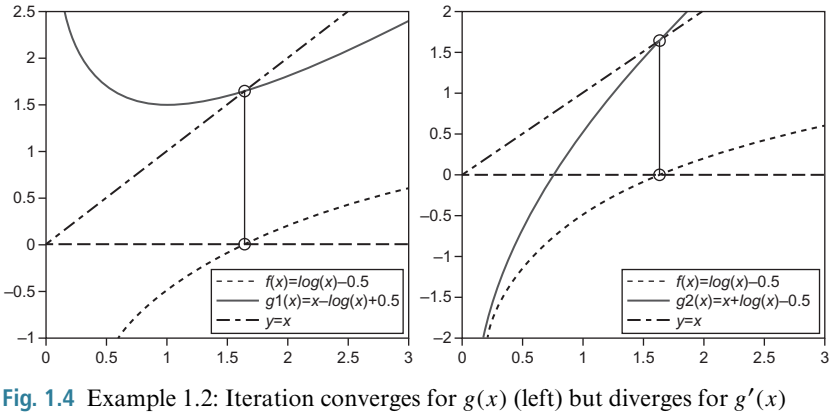

Fig. 1.4 Example 1.2: Iteration converges for $g(x)$ (left) but diverges for $g'(x)$ (right).

Example 1.3

$$f(x) = e^x - 1/x = 0$$

Taking a logarithm of the equation, we obtain an equivalent form (left in Fig. 1.5)

© in this web service Cambridge University Press & Assessment www.cambridge.org