Introduction To Optimization

Dr. Ranjit Kumar

E-Mail: du.ranjit@gmail.com

Tel: 91-9650289875

Formerly, Assistant Professor of Physics at Dyal Singh College,
University of Delhi, Delhi, India
Working as Quantitative Analyst in Investment Bank, Mumbai, India.

Optimization (constrained as well as unconstrained) method aims to find the maxima or minima of a function $f(\mathbf{x})$. In constrained optimization case, there is some constraint on $\mathbf{x}$. We will cover some of the famous unconstrained optimization techniques such as gradient descent method, stochastic gradient descent method, Newton's method, steepest descent method, random search method and simulated annealing method. The random search method and simulated annealing method are based on Monte-Carlo simulation method. We will also discuss the constrained optimization problem such as Lagrange method of undermined multiplier and Karush-Kuhn-Tucker (KKT) condition.

Calculus of One and Two Variables

I. INTRODUCTION

In optimization (constrained as well as unconstrained) problem our aim is to find the maxima or minima of a function $f(\mathbf{x})$. In constrained optimization case, there is some constraint on $\mathbf{x}$. In mathematical form, the constrained optimization problem can be written as

$$\min_{\mathbf{x}\in\mathcal{C}} f(\mathbf{x})$$

where $\mathcal{C}$ is some constraint set. If there is no constraint for $\mathbf{x}$, then it is called an unconstrained optimization problem.

- We will discuss the constrained as well as unconstrained optimization problem in this module.

- We will cover some of the famous unconstrained optimization techniques such as gradient descent method, stochastic gradient descent method, Newton's method, steepest descent method, random search method and simulated annealing method. The random search method and simulated annealing method are based on Monte-Carlo simulation method.

- We will also discuss the constrained optimization problem such as Lagrange method of undermined multiplier and Karush-Kuhn-Tucker (KKT) condition.

A. Elementary Calculus

Let us consider a function one variable $f(x)$. Then the derivative of the function is defined as

$$\frac{df}{dx}$$

where $\frac{df}{dx}$ represents the rate of change of f with respect to x. For example, if $f(x) = x^2$, then

$$\frac{df}{dx} = f'(x) = 2x$$

Let us now consider a function of two variables $f(x,y)$. The total derivative of the function is defined as

$$df = \left(\frac{\partial f}{\partial x}\right)\,dx + \left(\frac{\partial f}{\partial y}\right)\,dy$$

where $\frac{\partial f}{\partial x}$ on the right hand side represents the partial derivative of f with respect to x, keeping y constant and the second term $\frac{\partial f}{\partial y}$ represents the partial derivative of f with respect to y, keeping x constant.

Ex: Given that

$$f(x,y) = 4 - x^2 - 2y^2$$

Find $\frac{\partial f}{\partial x}(1,1)$ and $\frac{\partial f}{\partial y}(1,1)$.

Sol: We have

$$\frac{\partial f}{\partial x} = -2x \qquad \text{then} \qquad \frac{\partial f}{\partial x}(1,1) = -2$$

Similarly,

$$\frac{\partial f}{\partial y} = -4y \qquad \text{then} \qquad \frac{\partial f}{\partial y}(1,1) = -4$$

Higher Derivative: Suppose f is function of two variables, then its partial derivatives $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$ are also functions of two variables. Then the derivative of the derivative (also called as second partial derivative) are given by

$$\begin{aligned}
\frac{\partial^2 f}{\partial x^2} &= \frac{\partial}{\partial x}\left(\frac{\partial f}{\partial x}\right)\\
\frac{\partial^2 f}{\partial y \partial x} &= \frac{\partial}{\partial y}\left(\frac{\partial f}{\partial x}\right)\\
\frac{\partial^2 f}{\partial x \partial y} &= \frac{\partial}{\partial x}\left(\frac{\partial f}{\partial y}\right)\\
\frac{\partial^2 f}{\partial y^2} &= \frac{\partial}{\partial y}\left(\frac{\partial f}{\partial y}\right)
\end{aligned}$$

Ex: Find the second partial derivative of

$$f(x,y) = x^3 + x^2y^3 + 2y^2x$$

Sol: The first partial derivative of the function is

$$\frac{\partial f}{\partial x} = 3x^2 + 2xy^3 + 2y^2, \qquad \frac{\partial f}{\partial y} = 3x^2y^2 + 4xy$$

Then the second partial derivative becomes

$$\begin{aligned}
\frac{\partial^2 f}{\partial x^2} &= \frac{\partial}{\partial x}(3x^2+2xy^3+2y^2) = 6x+2y^3 \\
\frac{\partial^2 f}{\partial y\partial x} &= \frac{\partial}{\partial y}(3x^2+2xy^3+2y^2) = 6xy^2+4y \\
\frac{\partial^2 f}{\partial x\partial y} &= \frac{\partial}{\partial x}(3x^2y^2+4xy) = 6xy^2+4y \\
\frac{\partial^2 f}{\partial y^2} &= \frac{\partial}{\partial y}(3x^2y^2+4xy) = 6x^2y+4x
\end{aligned}$$

Note that $\frac{\partial^2 f}{\partial x\partial y} = \frac{\partial^2 f}{\partial y\partial x}$, which is not true in general. There are many examples where $\frac{\partial^2 f}{\partial x\partial y} \neq \frac{\partial^2 f}{\partial y\partial x}$. In the rest part of the notes, we will denote the partial derivatives by the following notation:

$$f_x = \frac{\partial f}{\partial x}, \quad f_y = \frac{\partial f}{\partial y}, \quad f_{x,y} = \frac{\partial^2 f}{\partial x\partial y}, \quad f_{x,x} = \frac{\partial^2 f}{\partial x^2}, \cdots$$

B. Taylor's Series

Assume a function of single variable $f : R \rightarrow R$ which is n times differentiable $f \in \mathcal{C}^m$ on an interval $[a, x]$. Denote $h = x - a$, then

$$f(x) = f(a) + \frac{h}{1!}\left.\frac{df}{dx}\right|_{x=a} + \frac{h^2}{2!}\left.\frac{d^2 f}{dx^2}\right|_{x=a} + \cdots + \frac{h^{n-1}}{(n-1)!}\left.\frac{d^{n-1} f}{dx^{n-1}}\right|_{x=a} + R_n$$

is the Taylor series expansion of $f(x)$ near the point $x = a$ and R_n is called the remainder term. The Taylor's formula for function of two variable $f(x, y)$ near the point (x_0, y_0) is given by

$$\begin{aligned}
f(x,y) &= f(x_0,y_0) + (x-x_0)\left.\frac{\partial f}{\partial x}\right|_{x=x_0,y=y_0} + (y-y_0)\left.\frac{\partial f}{\partial y}\right|_{x=x_0,y=y_0} + \frac{1}{2}(x-x_0)^2\left.\frac{\partial^2 f}{\partial x^2}\right|_{x=x_0,y=y_0} + \\
&\qquad \frac{1}{2}\left.\frac{\partial^2 f}{\partial y^2}\right|_{x=x_0,y=y_0} + (x-x_0)(y-y_0)\left.\frac{\partial^2 f}{\partial x\partial y}\right|_{x=x_0,y=y_0} + \cdots
\end{aligned}$$

where we assumed that

$$\frac{\partial^2 f}{\partial x\partial y} = \frac{\partial^2 f}{\partial y\partial x}$$

Ex: Find the Taylor's series expansion of

$$f(x) = e^{-x}$$

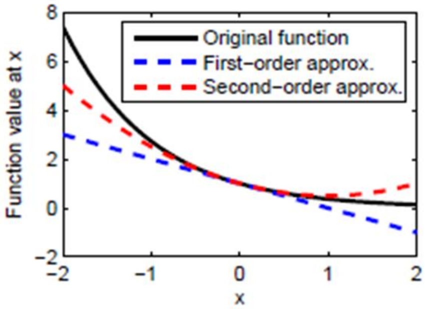


FIG. 1. Plot of e^{-x} and its first and second order Taylor approximation near $x = 0$.

near $x = 0$.

Sol: We have

$$f(x) = e^{-x}, \quad f'(x) = -e^{-x}, \quad f'(0) = -1, \quad f''(0) = 1$$

Therefore

$$\begin{aligned} f(x) = e^{-x} &= f(0) + (x-0)f'(0) + \frac{(x-0)^2}{2!}f''(0) + \cdots + \cdots \\ &= 1 - x + \frac{x^2}{2} + \cdots \end{aligned}$$

See Fig. 1.

C. Maxima and Minima of a Function of One Variable

Let us consider a function of one variable, say $f(x)$. As an example let us consider

$$f(x) = x^2$$

Now we know from elementary calculus that at the point of maxima or minima, the slope of the function is zero, i.e.,

$$\frac{df}{dx} = f'(x) = 2x = 0$$

From here, we obtain $x = 0$. Thus the point $x = 0$ (also called as stationary point) is a point of maxima or minima. Let us obtain the condition to check the nature of the stationary point.

D. Quadratic Approximation in One Variable

We know that the Taylor series expansion of the function $f(x)$ about the point x_0 is given by

$$f(x) = f(x_0) + \left.\frac{df}{dx}\right|_{x=x_0} + \frac{1}{2}(x-x_0)^2 \left.\frac{d^2f}{dx^2}(x_0)\right|_{x=x_0} + \cdots +$$

At a stationary point x_0 we have $f'(x_0) = 0$ and also assuming that the point x_0 is very close to the point x, so that the higher order term in the Taylor series can be ignored.

$$f(x) \approx f(x_0) + \frac{1}{2}(x-x_0)^2 \left. f''(x)\right|_{x=x_0}$$

This is the quadratic approximation of function $f(x)$ near x_0.

1. If $f''(x_0) > 0$, then graph of $f(x)$ near x_0 is parabola opening upward, and so f has a local minimum at $x = x_0$.

2. If $f''(x_0) < 0$, then graph of f near x_0 is parabola opening downward, and so f has a local maximum at $x = x_0$.

3. If $f''(x_0) = 0$, then f is a constant function near x_0 and so the quadratic approximation says nothing about the type of stationary point at x_0.

In the case of $f(x) = x^2$, we know that the function $f(x)$ is stationary at the point at $x = 0$ and hence

$$\frac{d^2f}{dx^2} = 2 > 0$$

Therefore, the stationary point $x = 0$ is the point of minima of the function $f(x)$.

Q1: Find the maxima or/and minima of the following functions:

$$f(x) = x^2 + 2x, \qquad f(x) = x^4 + x^3 + 3x^2$$

II. MAXIMA AND MINIMA OF FUNCTION OF TWO VARIABLES

For functions of one variable, we sometimes use the second derivative to distinguish among stationary points of different types.The situation is similar but a little more complicated for functions of two variables. Suppose that a function $f(x, y)$ has a local maxima or a local minimum at (x_0, y_0). If both partial derivative exist, then

$$f_x(x_0, y_0) = 0, \qquad f_y(x_0, y_0) = 0$$

There are three main types of stationary point for a function $f(x, y)$.

1. **Local Minimum Point:** A stationary point (x_0, y_0) is a local minimum point for f if $f(x_0, y_0) \leq f(x, y)$ for all (x, y) near (x_0, y_0).

2. **Local Maximum Point:** A stationary point (x_0, y_0) is a local maximum point for f if $f(x_0, y_0) \geq f(x, y)$ for all (x, y) near $(x_0, y_)$.

3. **Saddle Point:** A stationary point (x_0, y_0) is a saddle point for f if f assumes neither a local maximum nor a local minimum at (x_0, y_0).

A. Quadratic Approximation for Function of Two Variables

Let us now consider the Taylor series of this function $f(x, y)$ near the point (x_0, y_0).

$$\begin{aligned} f(x,y) &= f(x_0,y_0) + (x-x_0)\left.\frac{\partial f}{\partial x}\right|_{x=x_0,y=y_0} + (y-y_0)\left.\frac{\partial f}{\partial y}\right|_{x=x_0,y=y_0} + \frac{1}{2}(x-x_0)^2\left.\frac{\partial^2 f}{\partial x^2}\right|_{x=x_0,y=y_0} + \\ &\quad \frac{1}{2}\left.\frac{\partial^2 f}{\partial y^2}\right|_{x=x_0,y=y_0} + (x-x_0)(y-y_0)\left.\frac{\partial^2 f}{\partial x \partial y}\right|_{x=x_0,y=y_0} + \cdots \end{aligned}$$

Now we will assume that the point (x_0, y_0) is close to the point (x, y), so that the higher order term in $(x - x_0)$ and $(y - y_0)$ can be ignored on the right hand side.

$$\begin{aligned} f(x,y) &\approx f(x_0,y_0) + (x-x_0)f_x(x_0,y_0) + (y-y_0)f_y(x_0,y_0)y + \frac{(x-x_0)^2}{2}f_{xx}(x_0,y_0) + \\ &\quad (x-x_0)(y-y_0)f_{xy}(x_0,y_0) + \frac{(y-y_0)^2}{2}f_{yy}(x_0,y_0) \end{aligned}$$

Since (x_0, y_0) is a stationary point, then the first-order terms disappear. Here is what remains:

$$\begin{aligned} f(x,y) &\approx f(x_0,y_0) + \frac{(x-x_0)^2}{2}f_{xx}(x_0,y_0) + \\ &\quad (x-x_0)(y-y_0)f_{xy}(x_0,y_0) + \frac{(y-y_0)^2}{2}f_{yy}(x_0,y_0) \end{aligned} \tag{1}$$

The first term here is constant, and so what matters here are the last three terms, which have the form $Ax^2 + Bxy + Cy^2$, where $A = \frac{f_{xx}(x_0,y_0)}{2}$, $C = \frac{f_{yy}(x_0,y_0)}{2}$ and $B = f_{xy}(x_0, y_0)$. Then

$$f(x, y) - f(x_0, y_0) \approx Ax^2 + Bxy + Cy^2$$

We can write

$$\begin{aligned} f(x, y) - f(x_0, y_0) &\approx A\left(x^2 + \frac{B}{A}xy + \frac{C}{A}y^2\right) \\ &\approx A\left[\left(x + \frac{B}{2A}y\right)^2 + \left(\frac{C}{A} - \frac{B^2}{4A^2}\right)y^2\right] \end{aligned}$$

This shows that type of stationary point depends on the sign of coefficient of y^2. Let

$$\frac{C}{A} - \frac{B^2}{4A^2} \geq 0 \quad \text{or} \quad 4AC - B^2 \geq 0$$

1. If $4AC - B^2 > 0$, then f has either a local maxima or local minimum at (x_0, y_0), depends on whether $A < 0$ or $A > 0$.

2. If $4AC - B^2 < 0$, then f has saddle point at (x_0, y_0).

3. If $4AC - B^2 = 0$, then various things can happen and we normally look for more information.

We have $4AC - B^2 = f_{xx}f_{yy} - f_{xy}^2$. In other words $4AC - B^2$ is determinant of Hessian matrix

$$f''(x_0, y_0) = \begin{pmatrix} f_{xx}(x_0, y_0) & f_{xy}(x_0, y_0) \\ f_{xy}(x_0, y_0) & f_{yy}(x_0, y_0) \end{pmatrix}$$

We have $4AC - B^2 = f_{xx}f_{yy} - f_{xy}^2$. In other words $4AC - B^2$ is determinant of Hessian matrix

$$f''(x_0, y_0) = \begin{pmatrix} f_{xx}(x_0, y_0) & f_{xy}(x_0, y_0) \\ f_{xy}(x_0, y_0) & f_{yy}(x_0, y_0) \end{pmatrix}$$

Theorem: Let (x_0, y_0) be a stationary point of a function f and $f''(x_0, y_0)$ be the Hessian matrix of f and let

$$D = f_{xx}(x_0, y_0)f_{yy}(x_0, y_0) - f_{xy}^2(x_0, y_0)$$

be the determinant of $f''(x_0, y_0)$. Then

1. If $D > 0$ and $f_{xx}(x_0, y_0) > 0$, then f has local minimum at (x_0, y_0).

2. If $D > 0$ and $f_{xx}(x_0, y_0) < 0$, then f has local maxima at (x_0, y_0).

3. If $D < 0$, then f has saddle point at (x_0, y_0).

4. If $D = 0$, then more information is needed.

B. Matrix Representation

Note that the Eq. (1) in matrix form can be written as

$$f(\mathbf{x}) \approx f(\mathbf{x}_0) + \frac{1}{2}\mathbf{X}^T\mathbf{H}\mathbf{X} \tag{2}$$

where

$$X = \begin{pmatrix} x - x_0 \\ y - y_0 \end{pmatrix}, \qquad \mathbf{H} = \begin{pmatrix} f_{xx} & f_{xy} \\ f_{xy} & f_{yy} \end{pmatrix}$$

Note that the Eq. (2) is general case of one variable optimization problem. From this equation we can see that if

- $\mathbf{X}^T\mathbf{H}\mathbf{X} > 0$, then $f(x, y)$ has local minimum at (x_0, y_0).
- $\mathbf{X}^T\mathbf{H}\mathbf{X} < 0$, then $f(x, y)$ has local maxima at (x_0, y_0).
- $\mathbf{X}^T\mathbf{H}\mathbf{X} = 0$, then more information is needed.

Definition: A matrix $\mathbf{H}$ is called positive semi-definite (p.s.d.) if for all $\mathbf{X}$,

$$\mathbf{X}^T\mathbf{H}\mathbf{X} \geq 0$$

and positive definite (p.d.) if

$$\mathbf{X}^T\mathbf{H}\mathbf{X} > 0$$

Similarly, a matrix $\mathbf{H}$ is called negative definite (n.d.) if for all $\mathbf{X}$,

$$\mathbf{X}^T\mathbf{H}\mathbf{X} < 0$$

It can be verified that a p.d. matrix has positive eigenvalues and a n.d. matrix has negative eigenvalues.

Q1: Determine the Hessian and hence maxima/minima of the following function

$$f(x, y) = 2 - x^2 - xy - y^2, \qquad f(x, y) = x^2 + y^2$$

Q2: Suppose we are given the Hessian of some functions at three different points are as follows:

$$\mathbf{H}(\mathbf{X}_1) = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}, \quad \mathbf{H}(\mathbf{X}_2) = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}, \quad \mathbf{H}(\mathbf{X}_3) = \begin{bmatrix} 4 & 2 \\ 2 & 1 \end{bmatrix}$$

Which among the above points will be a local minimizer.

III. FUNCTION OF n VARIABLES

For a function of n real variables, $f(x_1, x_2, \ldots, x_n)$, we require that, at all stationary points, $\partial f/\partial x_i = 0$ for all x_i. In order to determine the nature of a stationary point, we must expand the function as a Taylor series about the point. The Taylor expansion for a function of n variables is given by

$$f(\mathbf{x}) = f(\mathbf{X}_0) + \sum_i \frac{\partial f}{\partial x_i}\mathbf{X}_i + \frac{1}{2!}\sum_i\sum_j \frac{\partial^2 f}{\partial x_i \partial x_j}\mathbf{X}_i\mathbf{X}_j + \cdots$$

where

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, \quad \mathbf{X}_i = x_i - x_{i0}$$

The partial derivatives are evaluated at $(x_{10}, x_{20}, \ldots, x_{n0})$. Now at all stationary points, $\frac{\partial f}{\partial x_i} = 0$ for all x_i. Hence finally, we have

$$\triangle f = f(\mathbf{x}) - f(\mathbf{x}_0) \approx \frac{1}{2}\sum_i\sum_j \frac{\partial^2 f}{\partial x_i \partial x_j}\mathbf{X}_i\mathbf{X}_j \tag{3}$$

This equation in matrix form can be written as

$$\triangle f = \mathbf{X}^T\mathbf{H}\mathbf{X}$$

where

$$\mathbf{X} = \begin{pmatrix} x_1 - x_{10} \\ x_2 - x_{20} \\ \vdots \\ x_n - x_{n0} \end{pmatrix}, \quad \mathbf{H} = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix}$$

Hence from above equation, we can see that if

- $\mathbf{X}^T\mathbf{H}\mathbf{X} > 0$, then $f(x_1, x_2, \cdots, x_n)$ has local minimum at $(x_{10}, x_{20}, \cdots, x_{n0})$.
- $\mathbf{X}^T\mathbf{H}\mathbf{X} < 0$, then $f(x_1, x_2, \cdots, x_n)$ has local maxima at $(x_{10}, x_{20}, \cdots, x_{n0})$.
- $\mathbf{X}^T\mathbf{H}\mathbf{X} = 0$, then more information is needed.

Since $\mathbf{H}$ is real and symmetric it has n real eigenvalues λ_r and n orthogonal eigenvectors e_r, which after suitable normalization satisfy

$$\mathbf{H}e_r = \lambda_r e_r, \qquad e_r^T e_s = \delta_{rs}$$

These eigenvectors form a basis set for the n-dimensional space and we can therefore expand $\mathbf{X}$ in terms of them, obtaining

$$\mathbf{X} = \sum_r a_r e_r$$

where the a_r are coefficients dependent upon $\mathbf{X}$. Substituting this into $\triangle f$, we find

$$\triangle f = \frac{1}{2}\mathbf{X}^T\mathbf{H}\mathbf{X} = \frac{1}{2}\sum_r \lambda_r a_r^2$$

Now, for the stationary point to be a minimum, we require

$$\triangle f = \frac{1}{2}\sum_r \lambda_r a_r^2 > 0$$

for all sets of values of the a_r, and therefore all the eigenvalues of $\mathbf{H}$ to be greater than zero. Conversely, for a maximum we require

$$\triangle f = \frac{1}{2}\sum_r \lambda_r a_r^2 < 0$$

and therefore all the eigenvalues of $\mathbf{H}$ to be less than zero. If the eigenvalues have mixed signs, then we have a saddle point. Note that the test may fail if some or all of the eigenvalues are equal to zero and all the non-zero ones have the same sign.

Ex: Derive the conditions for maxima, minima and saddle points for a function of two real variables, using the above analysis.

Sol: For a two-variable function the matrix $\mathbf{H}$ is given by

$$\mathbf{H} = \begin{pmatrix} f_{xx} & f_{xy} \\ f_{xy} & f_{yy} \end{pmatrix}$$

Therefore its eigenvalues satisfy the equation

$$\begin{vmatrix} f_{xx}-\lambda & f_{xy} \\ f_{xy} & f_{yy}-\lambda \end{vmatrix} = 0$$

Hence

$$\begin{aligned} (f_{xx}-\lambda)(f_{yy}-\lambda) - f_{xy}^2 &= 0 \\ \text{or} \quad \lambda^2 - (f_{xx}+f_{yy})\lambda + f_{xx}f_{yy} - f_{xy}^2 &= 0 \end{aligned}$$

Thus

$$2\lambda = (f_{xx}+f_{yy}) \pm \sqrt{(f_{xx}+f_{yy})^2 - 4(f_{xx}f_{yy} - f_{xy}^2)}$$

Now, that **H** is real and symmetric implies that its eigenvalues are real, and so for both eigenvalues to be positive (corresponding to a minimum), we require f_{xx} and f_{yy} positive and also

$$\begin{aligned} f_{xx}+f_{yy} &> \sqrt{(f_{xx}+f_{yy})^2 - 4(f_{xx}f_{yy} - f_{xy}^2)} \\ \text{or} \qquad f_{xx}f_{yy} - f_{xy}^2 &> 0 \end{aligned}$$

A similar procedure will find the criteria for maxima and saddle points.

IV. SUMMARY

Necessary and Sufficient Condition for Optimality: Our goal is to find the local (ideally global) minimizer of f.

- A point $\mathbf{x}^* \in R^n$ is said to be local minimizer of $f : R^n \to R$ if there exists $\epsilon > 0$ such that

$$f(\mathbf{x}^*) < f(\mathbf{x}) \quad \forall \quad \mathbf{x} \in R^n, \quad \text{such that} \quad |\mathbf{x} - \mathbf{x}^*| \leq \epsilon$$

Proposition 1: (Necessary and sufficient conditions for local optimality) – Let $f : R^n \to R$ be a function in $\mathcal{C}^2$ (here $\mathcal{C}^2$ refers to double differentiable function). If $\mathbf{x}^* \in R^n$ is a local minimizer of f, then

1. $\nabla f(\mathbf{x}^*) = 0$
2. $\nabla^2 f(\mathbf{x}^*)$ is positive semi-definite (p.s.d.)

If $\nabla^2 f(\mathbf{x}^*)$ is positive definite (p.d.), then x^* is a local minimizer of f. Hence in order to minimize a function f over R^n,

- We will have to find a point with zero gradient
- The Hessian at this point can then be used to determine if this point is indeed a (local) minimizer of f.

[1] Harikrishna Narasimhan, `http://drona.csa.iisc.ernet.in/~e0270/Jan-2015/Tutorials/lecture-notes-1.pdf`.

[2] Edwin K. P. Chong and Stanislaw H. Żak, *An Introduction to Optimization*__4th Ed., Wiley 2014.

Iterative Methods

V. ITERATIVE METHODS

In this lecture we will discuss some of the famous iterative methods to find the local minima of the function.

A. Gradient Descent Method

Let us consider a function

$$f(x) = (x-2)^2$$

In iterative process first we have to make some initial guess. Let us assume that our initial guess is x_1 which is lying to the left of the minimizer of this function (see Fig. 2). At x_1, the slope/gradient of this function is negative. Now one can decrease the value of the function by moving to the right of x_1. Similarly, if our initial guess is x_2 to the right of the minimizer, the slope/gradient of the function is positive, and therefore that one can decrease the function value by moving to the left of x_2.

In both cases, moving in a direction opposite to the sign of the function slope at a point produces a decrease in its value. In general, for any function $f : R^n \rightarrow R$ one can decrease the value of f at a point, by moving in a direction opposite to that of the gradient of f at that point.

Let $\mathbf{x}_0 \in R^n$ and $\mathbf{x}_1 \in R^n$ such that

$$\mathbf{x}_1 = \mathbf{x}_0 - \eta_0 \nabla f(\mathbf{x}_0), \qquad \eta_0 > 0$$

where

$$\mathbf{x}_0 = \begin{pmatrix} x_{10} \\ x_{20} \\ \vdots \\ x_{n0} \end{pmatrix}, \qquad \mathbf{x}_1 = \begin{pmatrix} x_{11} \\ x_{21} \\ \vdots \\ x_{n1} \end{pmatrix}$$

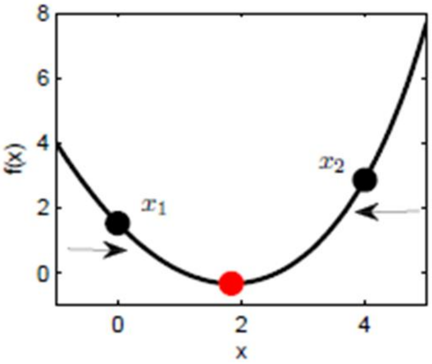


FIG. 2. Plot of one variable function $f(x) = (x-2)^2$.

Then from the first-order Taylor expansion of f at $\mathbf{x}_0$, we have

$$\begin{aligned} f(\mathbf{x}_1) &\approx f(\mathbf{x}_0) + (\mathbf{x}_1 - \mathbf{x}_0)\nabla f(\mathbf{x}_0) \\ &\approx f(\mathbf{x}_0) - \eta_0 \left|\nabla f(\mathbf{x}_0)\right|^2 \end{aligned}$$

from here we can see that

$$f(\mathbf{x}_1) < f(\mathbf{x}_0)$$

In the above equation

$$\nabla f = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix}$$

Now we move to another point x_2 and again follow the same procedure, i.e., at the point x_2, we have

$$\mathbf{x}_2 = \mathbf{x}_1 - \eta_1 \nabla f(\mathbf{x}_1), \qquad \eta_1 > 0$$

and

$$\begin{aligned} f(\mathbf{x}_2) &\approx f(\mathbf{x}_1) + (\mathbf{x}_2 - \mathbf{x}_1)\nabla f(\mathbf{x}_1) \\ &\approx f(\mathbf{x}_1) - \eta_1 \left|\nabla f(\mathbf{x}_1)\right|^2 \end{aligned}$$

In the $i+1$th iteration, the update rule is

$$\mathbf{x}_i = \mathbf{x}_{i-1} - \eta_{i-1} \nabla f(\mathbf{x}_{i-1}), \qquad \eta > 0$$

B. Stopping Criterion

In the above algorithm if

$$\nabla f(\mathbf{x}^k) = 0 \tag{4}$$

Then

$$\mathbf{x}_i = \mathbf{x}_{i-1}$$

We can use the above as the basis for a stopping criterion for the algorithm. However, condition (4) is not directly suitable as a practical stopping criterion, because the numerical computation of the gradient will rarely be identically equal to zero. We may compute

$$|f(\mathbf{x}_i) - f(\mathbf{x}_{i-1})|$$

and if the difference is less than some threshold, then we stop. Another alternative is to compute the absolute value of the difference (also called as norm)

$$|\mathbf{x}_i - \mathbf{x}_{i-1}|$$

and we stop if the norm is less than a pre-specified threshold (also called as tolerance). The algorithm for basic gradient descent method is described below in the table:

```
Gradient Descent Method
Input: $f : R^n \rightarrow R$
Initialize: $\mathbf{x}_0 \in R^n$
tol = 0.00001
i = 0
while($|\mathbf{x}_i - \mathbf{x}_{i-1}| > $ tol)
Select step-size $\eta_i > 0$
$x_i = x_{i-1} - \eta_{i-1} \nabla f(\mathbf{x}_i)$
Output: $x_i$
```

Ex: Use the gradient descent method to find the minimum of the function

$$f(x) = (x-2)^2$$

The Python code for the given function using the gradient descent method is provided in the table below:

```python
import numpy as np
x_0 = 0.0
x_1 = 4.0 # Our initial guess is x=4
precision = 0.00001

x_list = [x_1]
y_list = [f(x_1)]
i=0
eta_i = 0.01*np.exp(-i) # Variable step size

# f = lambda x: (x-2)**2
# returns the value of the derivative of our function

def f_prime(x):
    return 2*(x-2)

while abs(x_1 - x_0) > precision:
    x_0 = x_1
    grad_i = -f_prime(x_0)
    x_1 = x_0 + eta_i * grad_i
    x_list.append(x_1)
    y_list.append(f(x_1))
print("Local minimum occurs at:", x_1)
print("Number of steps:", len(x_list))
```

```
Local minimum occurs at: 2.00048548205
Number of steps: 413
```

In the above code the step size (also called as learning rate) is taken as

$$\eta_i = 0.01 * \exp(-i)$$

However it is not clear how one should choose η_i in each iteration. While a small value of η_i will result in slow convergence, with a large value, we will not be able to guarantee a decrease in f after every iteration. We will discuss the methods to choose appropriate step size.

> The gradient descent can be susceptible to local minima in general. If the function is a convex function, then the local minima is equal to the global minima. Thus gradient descent always converges if the function is convex.

C. Linear Regression

Let us suppose we have a data set of some population. The variables in the data are height and weight of some population. Our goal is to obtain a relation between height and weight of the people. Let us assume that our independent variable is weight (represented by x) and dependent variable is height (represented by y). Let us also assume that the relation between height and weight is linear, i.e.,

$$\hat{y}_i = \alpha_0 + \alpha_1 x_{i1} = \sum_{j=0}^{1} \alpha_j x_{ij}, \qquad x_{i0} = 1$$

For historical reasons, the function y is called a hypothesis. When the target variable that we're trying to predict is continuous, we call the learning problem a **regression problem**. When y can take on only a small number of discrete values we call it a **classification problem**. The parameters $\alpha_i,\ i = 1, 2$ are also called as weights. If the number of independent variables are more than one then the above equation gets modified to

$$\hat{y}_i = \alpha_0 + \alpha_1 x_{i1} + \alpha_2 x_{i2} + \cdots + \alpha_m x_{im} = \sum_{j=0}^{m} \alpha_j x_{ij}, \qquad x_{i0} = 1$$

In matrix notation, this equation can be written as

$$y_i = \boldsymbol{\alpha}^T \mathbf{X}$$

where

$$\boldsymbol{\alpha} = \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_m \end{pmatrix}, \qquad \mathbf{X} = \begin{pmatrix} 1 \\ x_{i1} \\ \vdots \\ x_{im} \end{pmatrix}$$

Now for $\hat{y}$ to be a good predictor of x, the difference between y and $\hat{y}$ should be minimum. In general, we minimize the cost function

$$J(\boldsymbol{\alpha}) = \sum_{i=n}[y_i - \hat{y}_i]^2 = \sum_{i=1}^{n}\left[y_i - \sum_{j=0}^{m}\alpha_j x_{ij}\right]^2$$

where n is the total number of data points. This is the familiar least-squares cost function that gives rise to the ordinary **least squares regression model**. Now choose $\boldsymbol{\alpha}$ that minimizes the cost function. In order to minimize the cost function, we first find the gradient of the cost function

$$\frac{\partial J(\boldsymbol{\alpha})}{\partial \alpha_j} = -2\sum_{i=1}^{n}\left[y_i - \sum_{k=0}^{m}\alpha_k x_{ik}\right]x_{ij}$$

In case of single variable regression problem, we have $m = 0,\ 1$. Therefore

$$\begin{aligned}
\frac{\partial J(\boldsymbol{\alpha})}{\partial \alpha_0} &= -2\sum_{i=1}^{n}\left[y_i - \alpha_0 - \alpha_1 x_{i1}\right] = -2\sum_{i=1}^{n}[y_i - \hat{y}_i] \\
\frac{\partial J(\boldsymbol{\alpha})}{\partial \alpha_1} &= -2\sum_{i=1}^{n}\left[y_i - \alpha_0 - \alpha_1 x_{i1}\right]x_{i1} = -2\sum_{i=1}^{n}[y_i - \hat{y}_i]x_{i1}
\end{aligned}$$

Now we use the gradient descent algorithm to find the optimal value of the parameters $\alpha_0,\ \alpha_1$. Here we start with some initial $\boldsymbol{\alpha}$ and update the parameters according to the rule

$$\alpha^i = \alpha^{i-1} - \eta^{i-1}\frac{\partial J(\boldsymbol{\alpha})}{\partial \alpha_j}$$

Substituting the value of the derivative, we obtain

$$\alpha_j^i = \alpha_j^{i-1} + 2\eta^{i-1}\sum_{i=1}^{n}\left[y_i - \sum_{k=0}^{m}\alpha_k x_{ik}\right]x_{ij} \tag{5}$$

The update rule for the parameters in case of single variable regression can therefore be written as

$$\begin{aligned}
\alpha_0^i &= \alpha_0^{i-1} + 2\eta^{i-1}\sum_{i=1}^{n}\left[y_i - \alpha_0\right] \\
\alpha_1^i &= \alpha_1^{i-1} + 2\eta^{i-1}\sum_{i=1}^{n}\left[y_i - \alpha_1 x_{i1}\right]x_{i1}
\end{aligned}$$

The rule is called the LMS (least mean squares) update rule, and is also known as the **Widrow-Hoff** learning rule. Note that the cost function J in our example is a convex quadratic function. We can prove that if the function is a convex function, then the local minima is equal to the global minima.

- For a single training example, this gives the update rule:

$$\alpha_j^i = \alpha_j^{i-1} + 2\eta^{i-1}\left[y_i - \sum_{k=0}^{m} \alpha_k x_{ik}\right] x_{ij} \tag{6}$$

D. Stochastic Gradient Descent Method

The LSM rule for a single training example is given by Eq. (6). Let us modify this method for training set of more than one example. There are two ways to do this:

1. The first is replace it with the following algorithm while ($|\alpha_j^i - \alpha_j^{i-1}| >$ tol)

   $$\alpha_j^i \;=\; \alpha_j^{i-1} + 2\eta^{i-1}\sum_{i=1}^{n}\left[y_i - \sum_{k=1}^{m} \alpha_{kj} x_{kj}\right] x_{ij}$$

   This method looks at every example in the entire training set on every step, and is called **batch gradient descent**. This can be quite slow if the training set is sufficiently large.

2. The second one is to use the **stochastic gradient descent method**. According to this algorithm, we repeatedly run through the training set, and each time we encounter a training example, we update the parameters according to the gradient of the error with respect to that single training example only. Thus, in stochastic gradient descent, we update our values after looking at each item in the training set, so that we can start making progress right away. The algorithm of stochastic gradient descent method are as follows:

```
Randomly shuffle the data $p$ times
for k=1 to $p$ do
for i=1 to n do
        $\alpha_j^i \;=\; \alpha_j^{i-1} + 2\eta^{i-1}\sum_{i=1}^{n}\left[y_i - \sum_{k=1}^{m} \alpha_{kj} x_{kj}\right] x_{ij}$
end for
end for
```

3. With batch gradient descent, we must go through the entire data set before we make any progress. With stochastic gradient algorithm though, we can make progress right away and continue to make progress as we go through the data set.

4. When the training set is large, stochastic gradient descent is often preferred over batch gradient descent.

VI. NEWTON'S METHOD

We will now discuss the Newton's method for unconstrained optimization. In gradient descent method we start with initial point $\mathbf{x}_0 \in R^n$ and at each iteration i obtains a new point by moving in the direction of the negative gradient of f at the current point:

$$\mathbf{x}_i = \mathbf{x}_{i-1} - \eta_{i-1}\nabla f(\mathbf{x}_{i-1}), \qquad \eta_{i-1} > 0$$

For appropriate choices of η_i's, this procedure always produces a decrease in the value of f after each iteration. This is evident from the first-order Taylor expansion of f, which tells us that for a small η_i,

$$f(\mathbf{x}_i) \approx f(\mathbf{x}_{i-1}) - \eta_{i-1}\nabla f(\mathbf{x}_{i-1})^T\nabla f(\mathbf{x}_{i-1}) < f(\mathbf{x}_{i-1})$$

We can use a more general update rule:

$$\mathbf{x}_i = \mathbf{x}_{i-1} + \eta_{i-1}\mathbf{d}_{i-1}, \qquad \eta_i > 0$$

with the direction $\mathbf{d}_i \in R^n$ chosen such that

$$\nabla f(\mathbf{x}_{i-1})^T\mathbf{d}_{i-1} < 0$$

We will call any direction that satisfies the mentioned condition as a descent direction at $\mathbf{x}_i$. From here we can also see that in the case of gradient descent method,

$$\mathbf{d}_{i-1} = -\nabla f(\mathbf{x}_{i-1})$$

Slow convergence is one of the problem of gradient descent method. Newton's method is one optimization procedure that uses a different descent direction at each iteration, which has better convergence properties than gradient descent in many settings.

- Remember that our aim is to find the point at which the gradient of the function is zero. The Newton's method is one such method which can be used to find the point at which the gradient of the function is zero. Before that let us first discuss the Newton-Raphson method of root finding.

1. Newton-Raphson Method

We seek one or more points at which the value of the function f is zero.

2. Interpretations of Newton's Method

In Newton's method, it is assumed at once that the function f is differentiable. This implies that the graph of f has a definite slope at each point and hence a unique tangent line. Now let us pursue the following simple idea. At a certain point $(x_0, f(x_0))$ on the graph of f, there is a tangent, which is a rather good approximation to the curve in the vicinity of that point. Analytically, it means that the linear function

$$l(x) = f'(x_0)(x - x_0) + f(x_0)$$

is close to the given function f near x_0. At x_0, the two functions l and f agree. We take the zero of l as an approximation to the zero of f. The zero of l is easily found:

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

Thus, starting with point x_0 (which we may interpret as an approximation to the root sought), we pass to a new point x_1 obtained from the preceding formula. Naturally, the process can be repeated (iterated) to produce a sequence of points:

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}, \qquad x_3 = x_2 - \frac{f(x_2)}{f'(x_2)}$$

etc. Under favorable conditions, the sequence of points will approach a zero of f. The geometry of Newton's method is shown in Fig. 3. The line $y = l(x)$ is tangent to the curve $y = f(x)$. It intersects the x-axis at a point x_1. The slope of $l(x)$ is $f'(x_0)$.

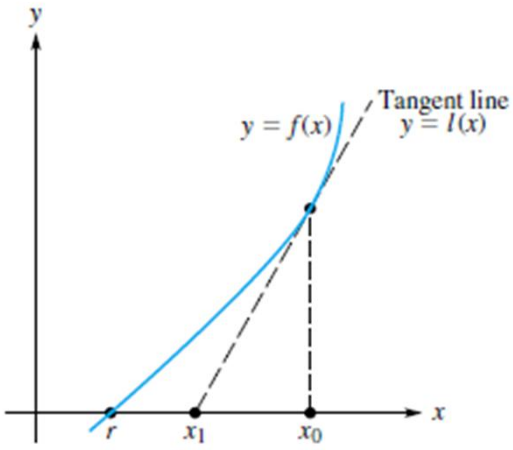


FIG. 3. Geometry of Newton-Raphson Method

3. *Another Interpretation of Newton-Raphson Method*

There are other ways of interpreting Newton's method. Suppose again that x_0 is an initial approximation to a root of f. We ask: What correction h should be added to x_0 to obtain the root precisely? Obviously, we want

$$f(x_0 + h) = 0$$

If f is a sufficiently well-behaved function, it will have a Taylor series at x_0. Thus, we could write

$$f(x_0) + hf'(x_0) + \frac{h^2}{2}f''(x_0) + \cdots = 0$$

Determining h from this equation is, of course, not easy. Therefore, we give up the expectation of arriving at the true root in one step and seek only an approximation to h. This can be obtained by ignoring all but the first two terms in the series:

$$f(x_0) + hf'(x_0) = 0$$

The h that solves this is not the h that solves $f(x_0 + h) = 0$, but it is the easily computed number

$$h = -\frac{f(x_0)}{f'(x_0)}$$

Our new approximation is then

$$x_1 = x_0 + h = x_0 - \frac{f(x_0)}{f'(x_0)}$$

and the process can be repeated. In retrospect, we see that the Taylor series was not needed after all because we used only the first two terms. In the analysis to be given later, it is assumed that f'' is continuous in a neighborhood of the root. This assumption enables us to estimate the errors in the process. If Newton's method is described in terms of a sequence $x_0, x_1, \ldots$, then the following recursive or inductive definition applies:

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

Naturally, the interesting question is whether

$$\lim_{n \to \infty} x_n = r$$

where r is the desired root.

- In the use of Newton's method, consideration must be given to the proper choice of a starting point.
- Usually, one must have some insight into the shape of the graph of the function.

Ex: Now we illustrate Newton's method by locating a root of

$$x^3 + x = 2x^2 + 3$$

We apply the method to the function

$$f(x) = x^3 - 2x^2 + x - 3$$

starting with $x_0 = 3$. Of course,

$$f'(x) = 3x^2 - 4x + 1$$

and these two functions should be arranged in nested form for efficiency: Figure 4 shows a computer plot of three iterations of Newton's method for this sample problem.

A. Optimization Using Newton's Method

Note that if we replace the function $f(x)$ by its derivative

$$g(x) = f'(x)$$

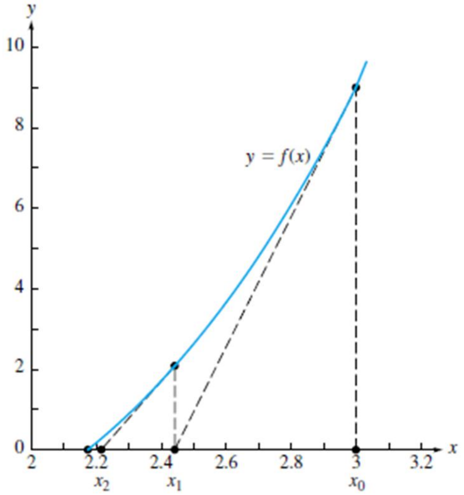


FIG. 4. Three steps of Newton's method for $f(x) = x^3 - 2x^2 + x - 3$.

TABLE I. Newton-Raphson method for the equation $x^3 - 2x^2 + x - 3$.

n	x_n	$f(x_n)$
0	3.0	9.0
1	2.4375	2.04
2	2.2130327224731445	0.256
3	2.1755549386143684	6.46×10^{-3}
4	2.1745601006550714	4.48×10^{-6}
5	2.1745594102932841	1.97×10^{-12}

then the Newton's method will give the root of the function $g(x)$, i.e., the root of the derivative of the function $f(x)$. Hence the update rule according to the Newton's method becomes

$$x_{i+1} = x_i - \frac{g(x_i)}{g'(x_i)} = 1 - \frac{f'(x_i)}{f''(x_i)}$$

For multivariate function, the above equation gets modified to

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \frac{g(\mathbf{x}_i)}{\nabla g(\mathbf{x}_i)} = 1 - \frac{\nabla f(\mathbf{x}_i)}{\nabla^2 f(\mathbf{x}_i)}$$

where $\nabla^2 f(\mathbf{x}_i)$ is the Hessian at the point $\mathbf{x}_i$. Also, it can be shown using the positive definiteness of the Hessian $\nabla^2 f(\mathbf{x}_i)$ at the initial point that the Hessian $\nabla^2 f(\mathbf{x}_i)$ at each subsequent point is also p.d. and hence invertible. The method is outlined below:

```
Newton's Method

Input: $f : R^n \to R$
Initialize: $\mathbf{x}_0 \in R^n$
tol = 0.00001
i = 0
while($|\mathbf{x}_i - \mathbf{x}_{i-1}| > \text{tol}$)
Select step-size $\eta_i > 0$
$x_i = x_{i-1} - \eta_{i-1} \left[\nabla^2 f(\mathbf{x}_i)\right]^{-1} \nabla f(\mathbf{x}_i)$
Output: $x_i$
```

According to the Newton's method, the update can be seen as moving in a direction

$$\mathbf{d}_{i-1} = -\left[\nabla^2 f(\mathbf{x}_{i-1})\right]^{-1} \nabla f(\mathbf{x}_{i-1})$$

Further, by positive definiteness of $\nabla^2 f(\mathbf{x}_{i-1})$, we have that

$$\nabla f(\mathbf{x}_{i-1})^T \mathbf{d}_{i-1} = -\nabla f(\mathbf{x}_{i-1})^T \left[\nabla^2 f(\mathbf{x}_{i-1})\right]^{-1} \nabla f(\mathbf{x}_{i-1}) < 0$$

and hence $\mathbf{d}_{i-1}$ is indeed a descent direction.

Ex: Apply Newton-Raphson method to find the root of the function

$$f(x) = x^3 - 2x^2 + x - 3$$

with starting point $x_0 = 3$.

```
import numpy as np
x_0 = 0.0
x_1 = 3.0 # Our initial guess is x=3
precision = 0.00001

x_list = [x_1]
y_list = [f(x_1)]
i=0
# f = lambda x: x**3-2*x**2+x-3.0
# returns the value of the function

def f(x):
    return x**3-2*x**2+x-3.0

# returns the value of the derivative of our function

def f_prime(x):
    return 3*x**2-4*x+1.0

while abs(x_1 - x_0) > precision:
    x_0 = x_1
    grad_i = -f_prime(x_0)
    x_1 = x_0 + f(x_0)/grad_i
    x_list.append(x_1)
    y_list.append(f(x_1))
print(”The root of the function is:”, x_1)
print(”Number of steps:”, len(x_list))
```

The root of the function is: 2.1745594102933126

Number of steps: 6

Ex: Use Newton's method to find the minima of the function

$$f(x) = \frac{x^2}{2} - \sin(x), \qquad x_0 = 0.5$$

```
import numpy as np
x_0 = 0.0
x_1 = 0.5 # Our initial guess is x=3
precision = 0.00001

x_list = [x_1]
y_list = [f(x_1)]
z_list = [f_doubleprime(x_1)]
i=0

# f = lambda x: (x**2/2.0)-np.sin(x)
# returns the value of the derivative of our function

def f_prime(x):
    return x-np.cos(x)

# returns the second derivative of our function

def f_doubleprime(x):
    return 1+np.sin(x)
```

```
while abs(x_1 - x_0) > precision:
    x_0 = x_1
    grad_i = -f_prime(x_0)
    hes_i = f_doubleprime(x_0)
    x_1 = x_0 + grad_i/hes_i
    x_list.append(x_1)
    y_list.append(f(x_1))
print(”Local minimum occurs at:”, x_1)
print(”Number of steps:”, len(x_list))
```

```
Local minimum occurs at: 0.739085133215
Number of steps: 5
```

B. Convex Function

Convex Set: A set $S \subseteq R^n$ is said to be convex if for any $\mathbf{x}_1,\ \mathbf{x}_2 \in S$ and $\alpha \in (0,1)$, the point

$$\alpha\mathbf{x}_1 + (1-\alpha)\mathbf{x}_2$$

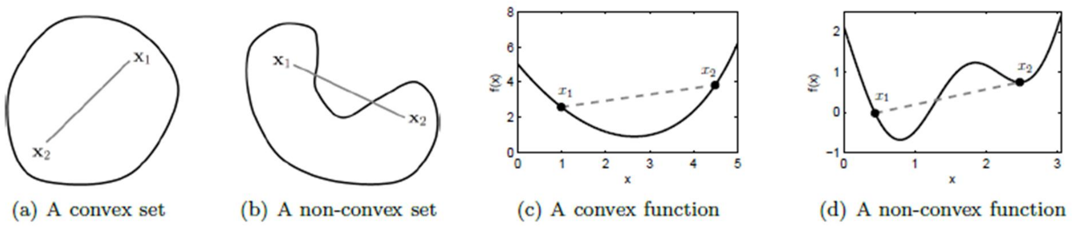
is also in S.

Convex Function: A function $f : R^n \ \rightarrow \ R$ is said to be convex if for any $\mathbf{x}_1,\ \mathbf{x}_2 \in R^n$ and $\alpha \in (0,1)$

$$f(\alpha\mathbf{x}_1 + (1-\alpha)\mathbf{x}_2) \leq \alpha f(\mathbf{x}_1) + (1-\alpha)f(\mathbf{x}_2)$$

From the above definition, it can be seen that the plot of a convex function between any two points will always lie (weakly) below the line joining the two points (see Fig. 5).

We can show that a function is convex if and only if its Hessian is positive semi-definite at all points. A key property that follows from this is that all local minimizers of a convex function are also its global minimizers.



(a) A convex set (b) A non-convex set (c) A convex function (d) A non-convex function

FIG. 5. Examples of convex and non-convex sets and functions.

Remark: Newton's method work best for the case of convex functions. In other words, Newton's method works well if $f''(x) > 0$ everywhere.

C. Secant Method

Newton's method for minimizing f uses second derivatives of f

$$x_i = x_{i-1} - \frac{f'(x_{i-1})}{f''(x_{i-1})}$$

If the second derivative is not available, we may attempt to approximate it using first derivative information. We may approximate f'' with

$$f''(x_{i-1}) = \frac{f'(x_{i-1}) - f'(x_{i-2})}{x_{i-1} - x_{i-2}}$$

Using the foregoing approximation of the second derivative, we obtain the algorithm

$$\begin{aligned} x_i &= x_{i-1} - f'(x_{i-1}) \cdot \frac{x_{i-1} - x_{i-2}}{f'(x_{i-1}) - f'(x_{i-2})} \\ &= \frac{x_{i-2} f'(x_{i-1}) - x_{i-1} f'(x_{i-2})}{f'(x_{i-1}) - f'(x_{i-2})} \end{aligned}$$

This method is called the secant method. Like Newton's method, the secant method does not directly involve values of $f(x_i)$. Instead, it tries to drive the derivative f' to zero.

VII. LINE SEARCH METHOD

Note that in the method discussed above, it is not clear how one should choose η in each iteration. While a small value of η_i will result in slow convergence, with a large value, we will not be able to guarantee a decrease in f after every iteration. Below, we discuss the schemes for selecting η_i, which come with convergence guarantees.

A. Exact Line Search

A natural approach for step-size selection would be to choose a value that produces the maximum decrease in function value at the given iteration:

$$\eta_{i-1} \in \arg\min_{\eta>0} \; f\left(x_{i-1} - \eta \nabla f(x_{i-1})\right.$$

Note that the above sub-problem is itself an optimization problem (involving a single variable). In some cases, this sub-problem can be solved analytically, resulting in a simple closed-form solution for η_{i-1}.

Iterative algorithms for finding a minimizer of $f : R^n \rightarrow R$ are of the form

$$\mathbf{x}_i = \mathbf{x}_{i-1} + \eta_{i-1}\mathbf{d}_{i-1} \qquad \eta_{i-1} > 0$$

In other words, iterative algorithms for solving optimization problems involve a line search at every iteration. Now η_{i-1} is chosen to minimize

$$\phi_{i-1}(\eta) = f(\mathbf{x}_{i-1} + \eta\mathbf{d}_{i-1})$$

The vector $\mathbf{d}_{i-1}$ is called the search direction. The choice of η_{i-1} itself involves a one-dimensional minimization. This choice ensures that under appropriate conditions

$$f(\mathbf{x}_i) < f(\mathbf{x}_{i-1})$$

B. The Method of Steepest Descent

Steepest descent is a gradient algorithm where the step size η_i is chosen to achieve the maximum amount of decrease of the objective function at each individual step.

$$\eta_{i-1} \in \arg\min_{\eta>0} \; f\left(\mathbf{x}_{i-1} - \eta \nabla f(\mathbf{x}_{i-1})\right)$$

At each step, starting from the point $\mathbf{x}_i$, we conduct a line search in the direction $-\nabla f(\mathbf{x}_{i-1})$ until a minimizer, $\mathbf{x}_{i-1}$, is found (see Fig. 6).

Proposition: If $\{\mathbf{x}_k\}_{k=0}^{\infty}$ is a steepest descent sequence for a given function $f : R^n \rightarrow R$ then for each k the vector $\mathbf{x}_{i+1} - \mathbf{x}_i$ is orthogonal to the vector $\mathbf{x}_{i+2} - \mathbf{x}_{i+1}$.

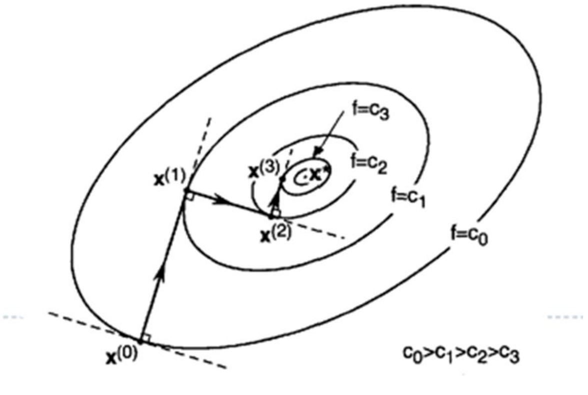


FIG. 6. Method of steepest descent.

Proof: From the iterative formula of the method of steepest descent it follows that

$$\langle \mathbf{x}_{i+1} - \mathbf{x}_i,\ \mathbf{x}_{i+2} - \mathbf{x}_{i+1} \rangle = \eta_i \eta_{i+1} \langle \nabla f(\mathbf{x}_i),\ \nabla f(\mathbf{x}_{i+1}) \rangle$$

To complete the proof it is enough to show

$$\langle \nabla f(\mathbf{x}_i),\ \nabla f(\mathbf{x}_{i+1}) \rangle = 0$$

Observe that η_i is a nonnegative scalar that minimizes

$$\phi_i(\eta) = f\left(\mathbf{x}_i - \eta \nabla f(\mathbf{x}_i)\right)$$

Hence using the condition of minimization, and chain rule gives us

$$\phi_i'(\eta_i) = \frac{d\phi_i}{d\eta}(\eta_i) = \nabla f\left(\mathbf{x}_i - \eta_i \nabla f(\mathbf{x}_i)\right)^T \left(-\nabla f(\mathbf{x}_i)\right) = -\langle \nabla f(\mathbf{x}_{i+1}),\ \nabla f(\mathbf{x}_i) \rangle = 0$$

Proposition: If $\{\mathbf{x}_k\}_{k=0}^{\infty}$ is a steepest descent sequence for a given function $f : R^n \rightarrow R$ and if $\nabla f(\mathbf{x}_i) \neq 0$, then

$$f(\mathbf{x}_{i+1}) < f(\mathbf{x}_i)$$

Proof: recall that

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \eta_i \nabla f(\mathbf{x}_i)$$

where $\eta_i \geq 0$ is the minimizer of

$$\phi_i(\eta) = f\left(\mathbf{x}_i - \eta \nabla f(\mathbf{x}_i)\right)$$

over all $\eta \geq 0$. Thus for $\eta \geq 0$, we have

$$\phi_i(\eta_i) \leq \phi_i(\eta)$$

By the chain rule

$$\phi_i'(0) = \frac{d\phi_i}{d\eta}(0) = -\nabla f\left(\mathbf{x}_i - 0 \cdot \nabla f(\mathbf{x}_i)\right)^T \left(\nabla f(\mathbf{x}_i)\right) = -\left|\nabla f(\mathbf{x}_i)\right|^2 < 0$$

because $\nabla f\left(\mathbf{x}_i\right) \neq 0$ by assumption. Thus

$$\phi_i'(0) < 0$$

This implies that there is an $\bar{\eta} > 0$ such that

$$\phi_i(0) > \phi_i(\eta), \qquad \forall \quad \eta \in (0, \bar{\eta}]$$

Hence

$$f(\mathbf{x}_{i+1}) = \phi_i(\eta_i) \leq \phi_i(\bar{\eta}) < \phi_i(0) = f(\mathbf{x}_i)$$

Ex: Use the steepest descent method to find the minimizer of

$$f(x_1, x_2, x_3) = (x_1 - 4)^4 + (x_2 - 3)^2 + 4(x_3 + 5)^4$$

The initial point is

$$\mathbf{x}_0 = \begin{pmatrix} 4 \\ 2 \\ -1 \end{pmatrix}$$

Sol: We find

$$\nabla f(\mathbf{x}) = \begin{pmatrix} 4(x_1 - 4)^3 \\ 2(x_2 - 3) \\ 16(x_3 + 5)^3 \end{pmatrix}$$

Hence

$$\nabla f(\mathbf{x}_0) = \begin{pmatrix} 0 \\ -2 \\ 1024 \end{pmatrix}$$

Now

$$\begin{aligned} \mathbf{x_0} - \eta \nabla f(\mathbf{x}_0) &= \begin{pmatrix} 4 \\ 2 \\ -1 \end{pmatrix} - \eta \begin{pmatrix} 0 \\ -2 \\ 1024 \end{pmatrix} \\ &= \begin{pmatrix} 4 \\ 2 + 2\eta \\ -1 - 1024\eta \end{pmatrix} \end{aligned}$$

Hence

$$f\left(\mathbf{x_0} - \eta \nabla f(\mathbf{x}_0)\right) = \begin{pmatrix} 0 \\ (2 + 2\eta - 3)^2 \\ (-1 - 1024\eta + 5)^4 \end{pmatrix}$$

To compute $\mathbf{x}_1$, we need

$$\begin{aligned} \eta_0 &= \arg\min_{\eta \geq 0} \; f\left(\mathbf{x_0} - \eta \nabla f(\mathbf{x}_0)\right) \\ &= \arg\min_{\eta \geq 0} \; f\left(0 + (2 + 2\eta - 3)^2 + 4(-1 - 1024\eta + 5)^4\right) \\ &= \arg\min_{\eta \geq 0} \phi_0(\eta) \end{aligned}$$

Using the secant method, we obtain

$$\eta_0 = 3.967 \times 10^{-3}$$

Then we compute

$$\mathbf{x}_1 = \mathbf{x}_0 - \eta_0 \nabla f(\mathbf{x}_0) = \begin{pmatrix} 4.000 \\ 2.008 \\ -5.062 \end{pmatrix}$$

To find $\mathbf{x}_2$, we first determine

$$\nabla f(\mathbf{x}_1) = \begin{pmatrix} 0.000 \\ -1.994 \\ -0.003875 \end{pmatrix}$$

Next we find η_1

$$\begin{aligned} \eta_1 &= \arg\min_{\eta \geq 0} \; f\left(\mathbf{x_1} - \eta \nabla f(\mathbf{x}_1)\right) \\ &= \arg\min_{\eta \geq 0} \; f\left(0 + (2.008 + 1.984\eta - 3)^2 + 4(-5.062 + 0.003875\eta + 5)^4\right) \\ &= \arg\min_{\eta \geq 0} \phi_1(\eta) \end{aligned}$$

Using the secant method again, we obtain $\eta_1 = 0.5000$. Then we compute

$$\mathbf{x}_2 = \mathbf{x}_1 - \eta_1 \nabla f(\mathbf{x}_1) = \begin{pmatrix} 4.000 \\ 3.000 \\ -5.060 \end{pmatrix}$$

To find $\mathbf{x}_3$, we first determine

$$\nabla f(\mathbf{x}_2) = \begin{pmatrix} 0.000 \\ 0.000 \\ -0.003525 \end{pmatrix}$$

Next we find η_2

$$\begin{aligned} \eta_2 &= \arg\min_{\eta \geq 0} \; f\left(\mathbf{x_2} - \eta \nabla f(\mathbf{x}_2)\right) \\ &= \arg\min_{\eta \geq 0} \; f\left(0.000 + 0.000 + 4(-5.060 + 0.003525\eta + 5)^4\right) \\ &= \arg\min_{\eta \geq 0} \phi_2(\eta) = 16.29 \end{aligned}$$

Then

$$\mathbf{x}_3 = \begin{pmatrix} 4.000 \\ 3.000 \\ -5.002 \end{pmatrix}$$

Note that the minimizer of f is $[4, 3, -5]^T$.

VIII. PYTHON CODE FOR LINEAR REGRESSION MODEL

Let us consider the following data set

x	1	2	3	4	5
y	3	4	5	6	8

Let us use the simple linear regression model to fit the data, i.e.,

$$\hat{y}_i = \alpha_0 + \alpha_1 x_i$$

We use the least square regression method to find the value of the coefficients. This means we minimize the expression (also called as cost function)

$$J = \sum_{i=1}^{N} (y_i - \widehat{y}_i)^2$$

For the simple linear regression analysis, sum of the squared error is given by

$$J = \sum_{i=1}^{N} (y_i - \alpha_0 - \alpha_1 x_i)^2$$

To determine the values of α_0 and α_1 that minimizes the SSE, we use the condition

$$\frac{\partial J}{\partial \alpha_0} = 0, \qquad \frac{\partial J}{\partial \alpha_1} = 0$$

from which we get the equations

$$\begin{aligned} \sum_{i=1}^{n} y_i &= \alpha_0 \, n + \alpha_1 \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i y_i &= \alpha_0 \sum_{i=1}^{n} x_i + \alpha_1 \sum_{i=1}^{n} x_i^2 \end{aligned}$$

This is a system of linear equations with α_0 and α_1 as the unknowns. In matrix form, these equations can be written as

$$\begin{bmatrix} \sum\limits_{i=1}^{n} x_i & n \\ \sum\limits_{i=1}^{n} x_i^2 & \sum\limits_{i=1}^{n} x_i \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_0 \end{bmatrix} = \begin{bmatrix} \sum\limits_{i=1}^{n} y_i \\ \sum\limits_{i=1}^{n} x_i y_i \end{bmatrix} \tag{7}$$

We can show that the value of the coefficients are given by

$$\begin{aligned} \alpha_0 &= \bar{y} - \alpha_1 \bar{x} \\ \alpha_1 &= \frac{n \sum x_i y_i - \sum x_i \sum y_i}{n \sum x_i^2 - \left(\sum x_i\right)^2} \end{aligned}$$

For the given x and y, the value of the parameters are obtained as

$$\alpha_0 = 1.6, \qquad \alpha_1 = 1.2$$

A. Gradient Descent Method

Now we will use the gradient descent method to find the value of these parameters (see the python code below):

```
————————————————————————
Batch Gradient Descent
————————————————————————


import numpy as np

# Input the data

x = [1, 2, 3, 4, 5]
y = [3, 4, 5, 6, 8]

n = len(x)

# Define the model y_hat = alpha_0+alpha_1*x

y_hat = lambda alpha_0,alpha_1,x: alpha_0 + alpha_1*x

# Define the square sum: sum_i[y[i]-y_hat[i]]^2

def J(x,y,n,alpha_0,alpha_1):
    returnValue = 0
    for i in range(n):
        returnValue += (y[i]-y_hat(alpha_0,alpha_1,x[i]))**2
    returnValue = returnValue
    return returnValue
```

```
# Find the gradient of the cost function J for all i.

def grad_J(x,y,n,alpha_0,alpha_1):
    grad = np.array([0.,0.])
    for i in range(n):
        grad[0] += -2*(y[i]-y_hat(alpha_0,alpha_1,x[i]))
        grad[1] += -2*(y[i]-y_hat(alpha_0,alpha_1,x[i]))*x[i]
    grad = grad
    return grad

# Use the gradient descent algorithm

alpha_old = np.array([0.,0.])
alpha_new = np.array([1.,1.]) # The algorithm starts at [1,1]
eta_k = 0.001 # step size
precision = 0.001
num_steps = 0
s_k = float(”inf”)

while np.linalg.norm(s_k) > precision:
    num_steps += 1
    alpha_old = alpha_new
    s_k = -grad_J(x,y,n,alpha_old[0],alpha_old[1])
    alpha_new = alpha_old + eta_k * s_k

print(”Local minimum occurs where:”)
print(”alpha_0 =”, alpha_new[0])
print(”alpha_1 =”, alpha_new[1])
print(”This took”,num_steps,”steps to converge”)
```

Local minimum occurs where:

alpha_0 = 1.59943175061

alpha_1 = 1.20015739597

This took 4014 steps to converge

Compare this result with the analytical result.

B. Stochastic Gradient Descent Method

Now we will use the stochastic gradient descent method to find the optimal value of the parameter.

```
————————————————————————
Stochastic Gradient Descent
————————————————————————
import numpy as np

# Input the data
x = [1, 2, 3, 4, 5]
y = [3, 4, 5, 6, 8]
n = len(x)

# Define the model y_hat = alpha_0+alpha_1*x
y_hat = lambda alpha_0,alpha_1,x: alpha_0 + alpha_1*x
```

C. Newton's Method

In this last section we use the Newton's method to find to find the optimal value of the regression model parameters. Note that in the Newton's method we also need Hessian of the cost function. Since the gradient of the cost function is given by

$$\frac{\partial J(\boldsymbol{\alpha})}{\partial \alpha_j} = -2\sum_{i=1}^{n}\left[y_i - \sum_{k=0}^{m}\alpha_k x_{ik}\right]x_{ij}$$

In case of single variable regression problem, we have $m = 0,\ 1$. Therefore

$$\begin{aligned}
\frac{\partial J(\boldsymbol{\alpha})}{\partial \alpha_0} &= -2\sum_{i=1}^{n}\left[y_i - \alpha_0 - \alpha_1 x_{i1}\right] = -2\sum_{i=1}^{n}[y_i - \hat{y}_i] \\
\frac{\partial J(\boldsymbol{\alpha})}{\partial \alpha_1} &= -2\sum_{i=1}^{n}\left[y_i - \alpha_0 - \alpha_1 x_{i1}\right] x_{i1} = -2\sum_{i=1}^{n}[y_i - \hat{y}_i] x_{i1}
\end{aligned}$$

The Hessian of the cost function is obtained by the differentiation of this function

$$\begin{aligned}
\frac{\partial^2 J(\boldsymbol{\alpha})}{\partial \alpha_0^2} &= 2\sum_{i=1}^{n} 1 = 2n \\
\frac{\partial^2 J(\boldsymbol{\alpha})}{\partial \alpha_1^2} &= 2\sum_{i=1}^{n}\left[x_{i1}\right] x_{i1}
\end{aligned}$$

These two expression can be written in a compact form as

$$\frac{\partial^2 J(\boldsymbol{\alpha})}{\partial \alpha_j^2} = 2\sum_{i=1}^{n}\sum_{k=0}^{m}[\delta_{jk} x_{ik}]\ x_{ij}, \qquad x_{i0} = 1$$

where

$$\delta_{jk} = \begin{cases} 1 & j = k \\ 0 & j \neq k \end{cases}$$

```
————————————————————————————————
Python Code For Newton's Method
————————————————————————————————
import numpy as np


# Input the data


x = [1, 2, 3, 4, 5]
y = [3, 4, 5, 6, 8]


n = len(x)


# Define the model y_hat = alpha_0+alpha_1*x


y_hat = lambda alpha_0,alpha_1,x: alpha_0 + alpha_1*x
```

```
# Define the square sum: sum_i[y[i]-y_hat[i]]^2

def J(x,y,n,alpha_0,alpha_1):
    returnValue = 0
    for i in range(n):
        returnValue += (y[i]-y_hat(alpha_0,alpha_1,x[i]))**2
    returnValue = returnValue
    return returnValue

# Find the gradient of the cost function J for all i.

def grad_J(x,y,n,alpha_0,alpha_1):
    grad = np.array([0.,0.])
    for i in range(n):
        grad[0] += -2*(y[i]-y_hat(alpha_0,alpha_1,x[i]))
        grad[1] += -2*(y[i]-y_hat(alpha_0,alpha_1,x[i]))*x[i]
    grad = grad
    return grad

# Find the Hessian of the cost function J for all i.

def hes_J(x,n):
    hes = np.array([0.,0.])
    for i in range(n):
        hes[0] +=  2.0
        hes[1] += 2*(x[i])*x[i]
    hes = hes
    return hes
```

```
# Use the Newton's method algorithm

alpha_old = np.array([0.,0.])
alpha_new = np.array([1.,1.]) # The algorithm starts at [1,1]
precision = 0.001
num_steps = 0
s_k = float(”inf”)

while np.linalg.norm(s_k) > precision:
    num_steps += 1
    alpha_old = alpha_new
    s_k = -grad_J(x,y,n,alpha_old[0],alpha_old[1])
    h_k = hes_J(x,n)
    alpha_new = alpha_old + (s_k/h_k)

print(”Local minimum occurs where:”)
print(”alpha_0 =”, alpha_new[0])
print(”alpha_1 =”, alpha_new[1])
print(”This took”,num_steps,”steps to converge”)
```

```
Local minimum occurs where:
alpha_0 = 1.5999881956585973
alpha_1 = 1.1999960652195325
This took 108 steps to converge
```

This result is very close to the analytical result obtained above.

D. Probabilistic Interpretation of Ordinary Least Square

Earlier we solved the least square regression problem using the least square cost function J. In this section, we will give a set of probabilistic assumptions, under which least squares regression is derived as a very natural algorithm. We know that the input and output are related via the equation

$$y_i = \sum_{j=0}^{m} \alpha_j x_{ij} + \epsilon_i$$

where ϵ_i is random noise, i.e, the part of the output that is not explained by the input. We also assume that ϵ_i are independently and identically distributed (IID) according to a Gaussian distribution $N(0, \sigma^2)$. In other words,

$$\epsilon_i \;\sim\; N(0, \sigma^2)$$

Then the density of ϵ_i can be written as

$$p(\epsilon_i) = \frac{1}{\sqrt{2\pi}\ \sigma} e^{-\frac{\epsilon_i^2}{2\sigma^2}}$$

This implies that

$$p(y_i|x_i; \boldsymbol{\alpha}) = \frac{1}{\sqrt{2\pi}\ \sigma} e^{-\frac{\left(y_i - \sum\limits_{j=0}^{m} \alpha_j x_{ij}\right)^2}{2\sigma^2}}$$

We can also write the distribution of y_i as

$$y_i|x_i; \boldsymbol{\alpha} \;\sim\; N\left(\sum_{j=0}^{m} \alpha_j x_{ij}, \sigma^2\right)$$

Now given $\mathbf{x}$ and $\boldsymbol{\alpha}$, what is the distribution of y_i. The probability of the data is given by $p(\mathbf{y}|\mathbf{x}, \boldsymbol{\alpha})$. This quantity is typically viewed a function of $\mathbf{y}$ and perhaps $\mathbf{x}$, for a fixed value of $\boldsymbol{\alpha}$. When we wish to explicitly view this as a function of $\boldsymbol{\alpha}$, we will instead call it the likelihood function:

$$L(\boldsymbol{\alpha}) = L(\boldsymbol{\alpha}; \mathbf{x}, \mathbf{y}) = p(\mathbf{y}|\mathbf{x}; \boldsymbol{\alpha})$$

Note that by the independence assumption on the ϵ_i's and hence also the y_i's given the x_i's, this can also be written

$$L(\boldsymbol{\alpha}) = \prod_{i=1}^{n} p(y_i|x_i; \boldsymbol{\alpha}) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\ \sigma} e^{-\frac{\left(y_i - \sum\limits_{j=0}^{m} \alpha_j x_{ij}\right)^2}{2\sigma^2}}$$

The principal of maximum likelihood says that we should choose $\boldsymbol{\alpha}$ so as to make the data as high probability as possible. i.e., we should choose $\boldsymbol{\alpha}$ to maximize $L(\boldsymbol{\alpha})$. Now we will maximize log of the likelihood of the function.

$$\begin{aligned}
l(\boldsymbol{\alpha}) \;&=\; \ln L(\boldsymbol{\alpha}) = \ln \left[\prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\ \sigma} e^{-\frac{\left(y_i - \sum\limits_{j=0}^{m} \alpha_j x_{ij}\right)^2}{2\sigma^2}} \right] \\
&=\; \sum_{i=1}^{n} \ln \left[\frac{1}{\sqrt{2\pi}\ \sigma} e^{-\frac{\left(y_i - \sum\limits_{j=0}^{m} \alpha_j x_{ij}\right)^2}{2\sigma^2}} \right] \\
&=\; n \ln \left(\frac{1}{\sqrt{2\pi}\ \sigma} \right) - \frac{1}{2\sigma^2} \sum_{i=1}^{n} \left(y_i - \sum_{j=0}^{m} \alpha_j x_{ij} \right)^2
\end{aligned}$$

Hence, maximizing $l(\boldsymbol{\alpha})$ gives the same answer as minimizing

$$\frac{1}{2} \sum_{i=1}^{n} \left(y_i - \sum_{j=0}^{m} \alpha_j x_{ij} \right)^2$$

which we recognize to be $J(\boldsymbol{\alpha})$, our original least squares cost function.

IX. REGULARIZATION

- Ridge Regression is a technique for analyzing multiple regression data that suffer from multicollinearity.

- When multicollinearity occurs, least squares estimates are unbiased, but their variances are large so they may be far from the true value.

- By adding a degree of bias to the regression estimates, ridge regression reduces the standard errors.

A. Multivariate Linear Regression

In multiple linear regression, we have

$$y_i = \alpha_0 + \alpha_1 x_{i1} + \alpha_2 x_{i2} + \cdots + \alpha_d x_{id} + \text{Error} \qquad i = 1, 2, \ldots, N$$

In matrix form this equation can be written as

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix} = \begin{pmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1d} \\ 1 & x_{21} & x_{22} & \cdots & x_{2d} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N1} & x_{N2} & \cdots & x_{Nd} \end{pmatrix} \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_d \end{pmatrix} + \text{Error}$$

Here

$$\underbrace{\mathbf{y}}_{N\times 1} = \underbrace{\mathbf{x}}_{N\times d+1} \underbrace{\boldsymbol{\alpha}}_{d+1\times 1}$$

where $\mathbf{y}$ is $N \times 1$ column vector, $\mathbf{x}$ is $N \times d + 1$ column vector, $\boldsymbol{\alpha}$ is $d + 1 \times 1$ column vector. For multivariate case, the hypothesis set can be written as

$$h(\mathbf{x}) = \boldsymbol{\alpha}^T \mathbf{x}$$

Then the sum of squared error can be written as

$$E_{in}(\boldsymbol{\alpha}) \ = \ \frac{1}{N} \sum_{n=1}^{N} [\boldsymbol{\alpha}^T \mathbf{x}_n - y_n]^2 = (\mathbf{x}\boldsymbol{\alpha} - \mathbf{y})^T (\mathbf{x}\boldsymbol{\alpha} - \mathbf{y}) = \frac{1}{N} ||\mathbf{x}\boldsymbol{\alpha} - \mathbf{y}||^2$$

where

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_N^T \end{bmatrix}, \qquad \mathbf{y} = \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \vdots \\ \mathbf{y}_N \end{bmatrix}$$

Now we will minimize the error term

$$\nabla E_{in}(\boldsymbol{\alpha}) = \frac{2}{N} \mathbf{x}^T (\mathbf{x}\boldsymbol{\alpha} - \mathbf{y}) = 0$$

Or,

$$\mathbf{x}^T \mathbf{x} \boldsymbol{\alpha} = \mathbf{x}^T \mathbf{y}$$

Finally we get

$$\boldsymbol{\alpha} = (\mathbf{x}^T \mathbf{x})^{-1} \mathbf{x}^T \mathbf{y} = \mathbf{x}^\dagger \mathbf{y}$$

where

$$\mathbf{x}^\dagger = (\mathbf{x}^T \mathbf{x})^{-1} \mathbf{x}^T$$

is called pseudo inverse of $\mathbf{x}$.

B. Effects of Multicollinearity

- Multicollinearity can create inaccurate estimates of the regression coefficients.
- Inflate the standard errors of the regression coefficients, deflate the partial t-tests for the regression coefficients, give false, nonsignificant, pvalues, and degrade the predictability of the model.

C. Sources of Multicollinearity

- **Data Collection:** In this case, the data have been collected from a narrow subspace of the independent variables.
- Physical constraints of the linear model or population.
- **Over Defined Model:** Here, there are more variables than observations ($p >> n$). If ($n >> p$) – that is, if n, the number of observations, is much larger than p, the number of variables – then the least squares estimates tend to also have low variance, and hence will perform well on test observations.

In the regression setting, the standard linear model

$$y_i = \alpha_0 + \alpha_1 x_{i1} + \alpha_2 x_{i2} + \cdots + \alpha_d x_{id} + e = \sum_{j=0}^{p} \alpha_j x_{ij}, \qquad x_{i0} = 1$$

In the case of more than one dependent variable, we have

$$\mathbf{y} = \mathbf{x}\boldsymbol{\alpha} + \mathbf{e}$$

where $\mathbf{X}$ is $N \times d + 1$ matrix, $\boldsymbol{\alpha}$ is $(d+1) \times 1$ matrix and $\mathbf{y}$ is $N \times 1$ column vector. In ordinary least squares, the regression coefficients are estimated using the formula

$$\boldsymbol{\alpha} = (\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\mathbf{y} = \mathbf{C}\mathbf{x}^T\mathbf{y}$$

where $\mathbf{C} = \mathbf{x}^T\mathbf{x}$ is the correlation matrix of independent variables. These estimates are unbiased so that the expected value of the estimates are the population values. That is,

$$E[\boldsymbol{\alpha}] = \boldsymbol{\alpha}$$

The variance-covariance matrix of the estimates is

$$\text{Var-Cov}[\boldsymbol{\alpha}] = \sigma^2(\mathbf{x}^T\mathbf{x})^{-1}$$

Assume that y's are standardized, so that $\sigma^2 = 1$. Then we can show that

$$\text{Var}[\alpha_j] = r^{ij} = \frac{1}{1 - R_j^2}$$

- R_j^2 is the R-squared value obtained from regression x_j on the other independent variables.

- In this case, this variance is the VIF.

- As the R-squared in the denominator gets closer and closer to one, the variance (and thus VIF) will get larger and larger.

- The rule of thumb cut-off value for VIF is 10. Solving backwards, this translates into an R-squared value of 0.90.

- Hence, whenever the R-squared value between one independent variable and the rest is greater than or equal to 0.90, you will have to face multicollinearity.

D. Summary

- When $p > n$, then the OLS method cannot be used. The variance in this case becomes infinite.

- The variance can be reduced by shrinking (constraining) the estimated coefficient at the cost of negligible increase in bias.

- The size of the coefficients increases exponentially with increase in model complexity. This is the problem of overfitting.

- Since there are large number of explanatory variables, it is not necessary that all of them are associated with the response. Also, it would be difficult to interpret the model with large number of variables. Hence we need some variable selection method, i.e., the method which exclude the irrelevant variables from a multiple regression model.

- Subset selection (Best subset selection, forward stepwise selection, backward stepwise selection), shrinkage (Ridge, LASSO, LARS) dimension reduction etc., are some of the variable reduction method.

E. Shrinkage Method

The shrinkage method shrinks some of the coefficient estimates toward zero. Least squares fitting procedure estimates $\alpha_0, \alpha_1, \ldots, \alpha_p$ using the values that minimize

$$\text{RSS} = \sum_{i=1}^{N} \left[y_i - \sum_{j=0}^{d} \alpha_j x_{ij} \right]^2, \qquad x_{i0} = 1$$

In ridge regression the coefficients are estimated by minimizing a slightly different quantity. In particular, the ridge regression coefficient estimate $\widehat{\boldsymbol{\alpha}}^R$ are the values that minimizes

$$\sum_{i=1}^{N} \left[y_i - \sum_{j=0}^{d} \alpha_j x_{ij} \right]^2 + \lambda \sum_{j=1}^{d} \alpha_j^2 = \text{RSS} + \lambda \sum_{j=1}^{d} \alpha_j^2 \tag{8}$$

The second term is called as shrinkage penalty (also called as l_2 penalty). The term is small when the coefficients $\alpha_0, \alpha_1, \ldots, \alpha_p$ are close to zero.

- When $\lambda = 0$, the ridge regression will produce the OLS estimates.
- As $\lambda \to \infty$, the impact of the shrinkage penalty grows, and the ridge regression coefficient estimates will approach zero.

Ridge regression proceeds by adding a small value, λ, to the diagonal elements of the correlation matrix. This is where ridge regression gets its name since the diagonal of ones in the correlation matrix may be thought of as a ridge. That is,

$$\boldsymbol{\alpha}^R = (\mathbf{C} + \lambda \mathbf{I})^{-1} \mathbf{x}^T \mathbf{y}$$

λ is a positive quantity less than one (usually less than 0.3).

Proof: Here we put constraint on the weights, i.e.,

$$\sum_{j=1}^{d} \alpha_j^2 \leq C$$

The we minimize

$$\frac{1}{N}(\mathbf{x}\boldsymbol{\alpha} - \mathbf{y})^T(\mathbf{x}\boldsymbol{\alpha} - \mathbf{y}) \qquad \text{subject to} \qquad \boldsymbol{\alpha}^T\boldsymbol{\alpha} \leq C$$

This is a constrained (inequality) optimization problem. Use the Lagrange method of undetermined multiplier to get the solution of this problem. Minimize

$$E_{in}(\boldsymbol{\alpha}) + \frac{\lambda}{N}\boldsymbol{\alpha}^T\boldsymbol{\alpha}$$

In other words, we are minimizing the augmented error

$$\begin{aligned} E_{aug}(\boldsymbol{\alpha}) &= E_{in}(\boldsymbol{\alpha}) + \frac{\lambda}{N}\boldsymbol{\alpha}^T\boldsymbol{\alpha} \\ &= \frac{1}{N}(\mathbf{x}\boldsymbol{\alpha} - \mathbf{y})^T(\mathbf{x}\boldsymbol{\alpha} - \mathbf{y}) + \frac{\lambda}{N}\boldsymbol{\alpha}^T\boldsymbol{\alpha} \\ &= \frac{1}{N}\left[(\mathbf{x}\boldsymbol{\alpha} - \mathbf{y})^T(\mathbf{x}\boldsymbol{\alpha} - \mathbf{y}) + \lambda\boldsymbol{\alpha}^T\boldsymbol{\alpha}\right] \end{aligned}$$

This is unconstrained problem and we can use quadratic programming to get the solution. Now the minima of the augmented error is obtained as

$$\nabla E_{aug}(\boldsymbol{\alpha}) = \mathbf{x}^T(\mathbf{x}\boldsymbol{\alpha} - \mathbf{y}) + \lambda\boldsymbol{\alpha} = 0$$

From here we obtain

$$\boldsymbol{\alpha}^R = (\mathbf{x}^T\mathbf{x} + \lambda\mathbf{I})^{-1}\mathbf{x}^T\mathbf{y}$$

The amount of bias in this estimator is given by

$$E[\boldsymbol{\alpha}^R - \boldsymbol{\alpha}] = \left[(\mathbf{x}^T\mathbf{x} + \lambda\mathbf{I})^{-1}\mathbf{x}^T\mathbf{x} - \mathbf{I}\right]\boldsymbol{\alpha}$$

and the covariance matrix is given by

$$\text{Var-Cov}[\boldsymbol{\alpha}^R] = (\mathbf{x}^T\mathbf{x} + \lambda\mathbf{I})^{-1}\mathbf{x}^T\mathbf{x}(\mathbf{x}^T\mathbf{x} + \lambda\mathbf{I})^{-1}$$

It can be shown that there exists a value of k for which the mean squared error (the variance plus the bias squared) of the ridge estimator is less than that of the least squares estimator.

F. Weight Decay

We have to minimize the following cost function for ridge regression

$$J = \sum_{i=1}^{N}\left[y_i - \sum_{j=0}^{d}\alpha_j x_{ij}\right]^2 + \lambda\sum_{j=1}^{d}\alpha_j^2$$

The gradient of this function is given by

$$\frac{\partial J}{\partial \alpha_j} = -2\sum_{i=1}^{N} x_{ij}\left[y_i - \sum_{k=0}^{d}\alpha_k x_{ik}\right] + 2\lambda\alpha_j$$

Now we use the gradient descent algorithm to find the optimal value of the parameters. Here we start with some initial $\boldsymbol{\alpha}$ and update the parameters according to the rule

$$\alpha^i = \alpha^{i-1} - \eta^{i-1}\frac{\partial J}{\partial \alpha_j}$$

Substituting the value of the derivative, we obtain

$$\alpha_j^i \;=\; \alpha_j^{i-1} + 2\eta^{i-1}\sum_{i=1}^{N}\left[y_i - \sum_{k=0}^{d}\alpha_k x_{ik}\right]x_{ij} - 2\eta^{i-1}\lambda\alpha_j^{i-1}$$

Hence

$$\boxed{\alpha_j^i = \alpha_j^{i-1}(1 - 2\eta^{i-1}\lambda) + 2\eta^{i-1}\sum_{i=1}^{N}\left[y_i - \sum_{k=0}^{p}\alpha_k x_{ik}\right]x_{ij}}$$

which is similar to the linear regression.

X. LOGISTIC REGRESSION

The dependent variable in this case is binary or discrete and use linear regression algorithm to try to predict y given x. For example

- Given occupation, age, education, income, loan amount, etc., what is the probability that a homeowner will default on his mortgage payments?

Note that in this case, $y_i \in \{0,\ 1\}$. To build a model, let us first try the simple linear regression model.

$$y_i = \alpha_0 + \alpha_1 x_i + \epsilon_i \tag{9}$$

Now the conditional expectation of y_i given x_i, $E[y_i|x_i]$, can be interpreted as the conditional probability that the event will occur given x_i, that is, Pr $(y_i = 1|x_i)$. To obtain the unbiased estimator, we assume that

$$E[\epsilon_i] = 0$$

Hence

$$E[y_i|x_i] = \alpha_0 + \alpha_1 x_i \tag{10}$$

Let $y_i = 1$ is probability p of occurrence of an event and $1 - p_i$ is the probability that $y_i = 0$. That is, y_i follows the Bernoulli probability distribution. Now, by the definition of mathematical expectation, we obtain:

$$E[y_i] = 1 \cdot p_i + 0(1 - p_i) = p_i \tag{11}$$

Comparing (10) and (11), we obtain

$$E[y_i|x_i] = \alpha_0 + \alpha_1 x_i = p_i$$

Since the probability p_i must lie between 0 and 1, we have the restriction

$$0 \leq E[y_i|x_i] \leq 1$$

Expression (9) is also called as linear probability model (LPM). The variance of y_i is given by

$$\text{Var}[y_i] = p_i(1 - p_i) \tag{12}$$

From here we can see that the variance is a function of the mean. Therefore the variance of ϵ_i ultimately depends on the values of x and hence is not homoscedastic. In case if there are m explanatory variables, we may write the probability of an event as

$$p_i = \boldsymbol{\alpha}^T \mathbf{x} = \sum_{j=0}^{m} \alpha_j x_{ij}, \quad \text{as before} \quad x_{i0} = 1$$

Since $y_i \in \{0,\ 1\}$ and hence we cannot use the simple OLS method (LPM) because when the event is very likely or very unlikely, the response may well be a probability larger than 1 or smaller than 0, respectively.

A. Non-Normality of the Disturbances ϵ_i

If we write (9) as

$$\epsilon_i = y_i - \alpha_0 - \alpha_1 x_i$$

Now

$$\begin{aligned} \epsilon_i &= 1 - \alpha_0 - \alpha_1 x_i, \quad \text{when} \quad y_i = 1, \quad p_i \\ \epsilon_i &= -\alpha_0 - \alpha_1 x_i, \quad \text{when} \quad y_i = 0, \quad 1 - p_i \end{aligned}$$

Hence, ϵ_i cannot be assumed to be normally distributed; they follow the Bernoulli distribution. The nonfulfillment of the normality assumption may not be so critical as it appears because we know that the OLS point estimates still remain unbiased.

B. Logistic Function

To overcome the difficulty of LPM, we will use the logistic or sigmoid function and is given by

$$p(z) = p(\boldsymbol{\alpha}^T \mathbf{x}) = \frac{e^z}{1 + e^z}$$

where

$$z = \boldsymbol{\alpha}^T \mathbf{x} = \alpha_0 + \sum_{j=1}^{m} \alpha_j x_j$$

and therefore

$$z_i = \sum_{j=0}^{m} \alpha_j x_{ij}, \qquad x_{i0} = 1$$

If there is only one example (x, y), then

$$z_i = \sum_{j=0}^{1} \alpha_j x_{ij} = \alpha_0 + \alpha_1 x_{i1}, \qquad x_{i0} = 1$$

Notice that $p(z)$ tends towards 1 as $z \;\rightarrow\; \infty$, and $p(z)$ tends towards 0 as $z \;\rightarrow\; -\infty$. Moreover, $p(z)$, is always bounded between 0 and 1. This equation may be rewritten as

$$\ln\left(\frac{p}{1-p}\right) = \boldsymbol{\alpha}^T \mathbf{x}$$

The logarithm of the odds ratio $\left(\frac{p}{1-p}\right)$ is known as a logit function. If

$$\frac{p}{1-p} = 10 \qquad \Rightarrow \qquad p = 0.909$$

Let us now take the derivative of the sigmoid function

$$\begin{aligned} p'(z) &= \frac{d}{dz}\left(\frac{e^z}{1+e^z}\right) = \frac{(1+e^z)e^z - e^z \cdot e^z}{(1+e^z)^2} \\ &= \frac{e^z}{(1+e^z)^2} = \frac{p(z)}{1+e^z} = p(z)(1-p(z)) \end{aligned}$$

Let us now fit the parameters via the maximum likelihood estimation.

C. Maximum Likelihood Estimation

To build the maximum-likelihood function, we observe that y_i is the realization of a Bernoulli variable, which may be regarded as a binomial variable when only one experiment is carried out. Hence

$$\begin{aligned} p(y=1|\mathbf{x},\boldsymbol{\alpha}) &= p \\ p(y=0|\mathbf{x},\boldsymbol{\alpha}) &= 1-p \end{aligned}$$

Note that this can be written more compactly as

$$p(y_i|x_i,\alpha_i) = p_i^{y_i}(1-p_i)^{1-y_i}$$

The probability p_i depends on observation x_i and parameters $\boldsymbol{\alpha}$. We assume the independence of the errors, so observations are independent and the likelihood function is just the product of individual probabilities:

$$L = \prod_{i=1}^{n} p_i^{y_i}(1-p_i)^{1-y_i}$$

The task of maximizing L can be somewhat simplified by taking its logarithm:

$$\begin{aligned} \ln L = l(\mathbf{x},\boldsymbol{\alpha}) &= \sum_{i=1}^{n}[y_i\ln(p(x_i)) + (1-y_i)[\ln(1-p(x_i))]] \\ &= \sum_{i=1}^{n}\ln[1-p(x_i)] + \sum_{i=1}^{n} y_i \ln\left[\frac{p(x_i)}{1-p(x_i)}\right] \\ &= \sum_{i=1}^{n}\ln\left[\frac{1}{1+\exp\left(\sum\limits_{j=0}^{m}\alpha_j x_{ij}\right)}\right] + \sum_{i=1}^{n} y_i\left(\sum_{j=0}^{m}\alpha_j x_{ij}\right) \\ &= -\sum_{i=1}^{n}\ln\left[1+\exp\left(\sum_{j=0}^{m}\alpha_j x_{ij}\right)\right] + \sum_{i=1}^{n} y_i\left(\sum_{j=0}^{m}\alpha_j x_{ij}\right) \end{aligned}$$

Now our aim is to maximize the log-likelihood. To do that we will use the gradient descent method. The update rule in this case can be written as

$$\boldsymbol{\alpha}_i = \boldsymbol{\alpha}_{i-1} + \eta_{i-1}\nabla l(\boldsymbol{\alpha}_{i-1})$$

Note the positive rather than negative sign in the update formula, since we're maximizing, rather than minimizing, a function now. In case if we have only one training example (x,y) and $\boldsymbol{\alpha} = \{\alpha_0,\alpha_1\}$, then

$$\begin{aligned} l(\mathbf{x},\boldsymbol{\alpha}) &= -\ln\left[1+\exp\left(\boldsymbol{\alpha}^T\mathbf{x}\right)\right] + y(\boldsymbol{\alpha}^T\mathbf{x}) \\ &= -\ln\left[1+\exp\left(\alpha_0+\alpha_1 x\right)\right] + y[\alpha_0+\alpha_1 x] \end{aligned}$$

Therefore

$$\begin{aligned}\frac{\partial l}{\partial \alpha_0} &= -\frac{\exp(\alpha_0+\alpha_1 x)}{1+\exp(\alpha_0+\alpha_1 x)} + y = y - p(x)\\ \frac{\partial l}{\partial \alpha_1} &= -\frac{\exp(\alpha_0+\alpha_1 x)}{1+\exp(\alpha_0+\alpha_1 x)}\ x + y\ x = [y - p(x)]x\end{aligned}$$

These two equations can be combined into a single equation as

$$\frac{\partial l}{\partial \alpha_j} = [y - p(x)]x_j, \qquad j = 0, 1$$

If there are n examples, then

$$\frac{\partial l}{\partial \alpha_j} = \sum_{i=1}^{n}[y_i - p(x_i)]x_{ij}$$

Hence the update rule (in ith iteration) according to the gradient descent method can be written as

$$\alpha_j^i = \alpha_j^{i-1} + 2\eta^{i-1}\sum_{i=1}^{n}[y_i - p(x_i)]x_{ij}$$

This is identical to the LMS update rule, but this is not the same algorithm, because $p(x_i)$ is now defined as a non-linear function of $\boldsymbol{\alpha}^T x_i$. The update rule according to the Newton's method can be written as

$$\boldsymbol{\alpha}_i = \boldsymbol{\alpha}_{i-1} - \frac{\nabla l(\boldsymbol{\alpha}_{i-1})}{\nabla^2 l(\boldsymbol{\alpha}_{i-1})}$$

where $\nabla^2 l(\boldsymbol{\alpha}_{i-1})$ is Hessian of the matrix. When Newton's method is applied to maximize the logistic regression log likelihood function $l(\boldsymbol{\alpha})$, the resulting method is also called **Fisher scoring**.

D. Summary Of Logistic Regression

- The model consists of a vector $\boldsymbol{\alpha}$ in n-dimensional feature space.

- For a point $\mathbf{x}$ in feature space, project it onto $\boldsymbol{\alpha}$ to convert it into a real number $\mathbf{z}$ in the range in the range $-\infty$ to $+\infty$

$$\mathbf{z} = \boldsymbol{\alpha}^T\mathbf{x} = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 + \cdots + \alpha_n x_n$$

- Map $\mathbf{z}$ to the range 0 to 1 using the logistic function

$$p = \frac{e^z}{1+e^z}$$

- Overall, logistic regression maps a point $\mathbf{x}$ in n-dimensional feature space to a value in the range 0 to 1.
- Need to optimize $\boldsymbol{\alpha}$ so the model gives the best possible reproduction of training set labels possible reproduction of training set labels. On really large data sets, may use stochastic gradient descent method.

[1] Harikrishna Narasimhan, `http://drona.csa.iisc.ernet.in/~e0270/Jan-2015/Tutorials/lecture-notes-2.pdf`.

[2] Edwin K. P. Chong and Stanislaw H. Żak, *An Introduction to Optimization*␣4th Ed., Wiley 2014.

[3] `https://github.com/dtnewman/gradient_descent/blob/master/stochastic_gradient_descent.ipynb`.

[4] Andrew Ng, `http://cs229.stanford.edu/notes/cs229-notes1.pdf`.

[5] Steven C. Chapra & Raymond P. Canale, *Numerical Methods for Engineers*␣Sixth Ed.

[6] Damodar N. Gujrati, *Basic Econometrics*.

Quadratic Programming

XI. QUADRATIC PROGRAMMING

In this chapter we will discuss the steepest descent method for quadratic programming. A quadratic function is of the form

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} - \mathbf{b}^T\mathbf{x}$$

where $\mathbf{Q} \in R^{m\times n}$ is a symmetric positive define matrix, $\mathbf{b} \in R^n$ and $\mathbf{x} \in R^n$. The unique minimizer of f can be found by setting the gradient of f to zero, where

$$\nabla f = \mathbf{Q}\mathbf{x} - \mathbf{b}$$

and Hessian of f is

$$\nabla^2 f = \mathbf{Q} > 0$$

Let us define

$$\mathbf{g} = \nabla f$$

Then, for the steepest descent algorithm for the quadratic function can be represented as

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \eta_i \mathbf{g}_i$$

where

$$\begin{aligned}\eta_i &= \arg\min_{\eta\geq 0}\; f(\mathbf{x}_i - \eta\mathbf{g}_i)\\ &= \arg\min_{\eta\geq 0}\left(\frac{1}{2}(\mathbf{x}_i - \eta\mathbf{g}_i)^T\mathbf{Q}(\mathbf{x}_i - \eta\mathbf{g}_i) - \mathbf{b}^T(\mathbf{x}_i - \eta\mathbf{g}_i)\right)\end{aligned}$$

Now assuming that $\mathbf{g}_i \neq 0$, for if $\mathbf{g}_i = 0$, then $\mathbf{x}_i = \mathbf{x}^*$ and the algorithm stops. Since $\eta_i \geq 0$ is a minimizer of

$$\phi_i(\eta) = f(\mathbf{x}_i - \eta\mathbf{g}_i)$$

We obtain

$$\frac{d\phi_i}{d(\eta)} = (\mathbf{x}_i - \eta\mathbf{g}_i)^T\mathbf{Q}(-\mathbf{g}_i) - \mathbf{b}^T(-\mathbf{g}_i) = 0$$

Or, equivalently

$$\eta\mathbf{g}_i^T\mathbf{Q}\mathbf{g}_i = (\mathbf{x}_i^T\mathbf{Q} - \mathbf{b}^T)\mathbf{g}_i$$

Since

$$\mathbf{g}_i = \mathbf{Q}\mathbf{x} - \mathbf{b}$$

Hence

$$\mathbf{g}_i^T = \mathbf{x}^T\mathbf{Q} - \mathbf{b}^T$$

Hence

$$\eta_i = \frac{\mathbf{g}_i^T\mathbf{g}_i}{\mathbf{g}_i^T\mathbf{Q}\mathbf{g}_i}$$

In summary, the method of steepest descent for the quadratic programming takes the form

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \frac{\mathbf{g}_i^T\mathbf{g}_i}{\mathbf{g}_i^T\mathbf{Q}\mathbf{g}_i}\ \mathbf{g}_i$$

where

$$\mathbf{g}_i = \nabla f(\mathbf{x}_i) = \mathbf{Q}\mathbf{x} - \mathbf{b}$$

XII. QUASI-NEWTON'S METHOD

We know that the update rule according to Newton's method (for unit step-size) is given by

$$\mathbf{x}_i = \mathbf{x}_{i-1} - \frac{\nabla f(\mathbf{x}_{i-1})}{\nabla^2 f(\mathbf{x}_{i-1})} = \mathbf{x}_{i-1} - \frac{\mathbf{g}(\mathbf{x}_{i-1})}{\mathbf{g}'(\mathbf{x}_{i-1})}$$

where

$$\mathbf{g}(\mathbf{x}_{i-1}) = \nabla f(\mathbf{x}_{i-1})$$

If step size is not unit, then

$$\mathbf{x}_i = \mathbf{x}_{i-1} - \eta_{i-1}\frac{\mathbf{g}(\mathbf{x}_{i-1})}{\mathbf{g}'(\mathbf{x}_{i-1})}$$

where η_{i-1} is chosen to ensure that

$$f(\mathbf{x}_{i+1}) < f(\mathbf{x}_i)$$

We may choose η as

$$\eta_i = \arg\min_{\eta \geq 0} f\left(\mathbf{x}_i - \frac{\mathbf{g}(\mathbf{x}_i)}{\mathbf{g}'(\mathbf{x}_i)}\right)$$

A computational drawback of Newton's method is the need to evaluate the inverse of Hessian matrix, i.e., $g'(\mathbf{x}_{i-1})^{-1}$. To avoid the computation of $g'(\mathbf{x}_{i-1})^{-1}$, the quasi-Newton methods use an approximation to $g'(\mathbf{x}_{i-1})^{-1}$ in place of true inverse.

[1] Wei-Ta Chu, `https://www.cs.ccu.edu.tw/~wtchu/courses/2014s_OPT/Lectures/Chapter%208%20Gradient%20Methods.pdf`.

Constrained Optimization

XIII. CONSTRAINED OPTIMIZATION

In this lecture we will discuss the constrained optimization problem. In general, the constrained optimization problem in mathematical form can be written as

$$\min_{\mathbf{x} \in \mathcal{C}} f(\mathbf{x}), \qquad \mathcal{C} \subseteq R^n$$

Furthermore, the constraint set can be represented using a set of equality and inequality constraints:

$$\min_{\mathbf{x} \in R^n} f(\mathbf{x})$$

such that

$$\begin{aligned} g_i(\mathbf{x}) &= 0, \qquad i = 1, 2, \ldots, p \\ h_j(\mathbf{x}) &\leq 0, \qquad j = 1, 2, \ldots, q \end{aligned}$$

Recall that in an unconstrained problem, a necessary condition for a point to be a local minimizer of a function of interest is simply that the gradient of the function is zero. However, in a constrained problem setting, it is possible that the gradient of the given function is non-zero at every point that satisfies the specified constraints. First we will discuss the case of equality constraint. Then in the later section, we will discuss the case of inequality constraint.

A. Lagrange Method of Undetermined Multiplier

This method is also known as **Karush-Kuhn-Tucker** (KKT) Conditions for equality constraint. For a function for single variable $f(x)$, there is no constrained optimization problem. In case of function of one variable there is only one independent variable x and hence we cannot put constraint on it. Let us try to maximize or minimize a function of two variable $f(x, y)$ subject to the condition that (x, y) also satisfies an equation

$$g(x, y) = c$$

where c is a constant.

- Here x and y are not independent variable, but are constrained by the relation $g(x, y) = c$. The set of (x, y) is a level curve for g.

- Thus we have $g(x, y) = c$ is a curve and we maximize or minimize f along that curve.

To do this, first solve $g(x, y)$ for one variable, say $y(x)$. Then plug that value in $f(x, y)$, i.e., we have $f(x, y(x))$. Now it is only a function of x. Then finally differentiating it with respect to x and setting it equal to zero as we do in the one variable calculus. Since $f = f(x, y(x))$, we have by chain rule

$$df = \frac{\partial f}{\partial x} + \frac{\partial f}{\partial y}\frac{dy}{dx} = 0 \tag{13}$$

Since $g(x, y) = c$, we have

$$dg = \frac{\partial g}{\partial x}dx + \frac{\partial g}{\partial y}dy = 0,$$

or

$$\frac{dy}{dx} = -\frac{\partial g/\partial x}{\partial g/\partial y}$$

Using this value of dy/dx in (13), we obtain

$$\frac{\partial f}{\partial x} - \frac{\partial f}{\partial y}\frac{\partial g/\partial x}{\partial g/\partial y} = 0$$

or

$$\frac{\partial g}{\partial y}\frac{\partial f}{\partial x} - \frac{\partial f}{\partial y}\frac{\partial g}{\partial x} = 0 \tag{14}$$

Let us illustrate this method with an example.

Ex: Let $f(x, y) = x^2+y^2$ and $g(x, y) : x+2y = 1$. Thus we have to maximize $f(x, y) = x^2+y^2$ subject to the constraint $x + 2y = 1$. To solve this equation, we have $x = 1 - 2y$. Hence

$$f(x, y) = x^2 + y^2 = (1 - 2y)^2 + y^2,$$

and maximize f with respect to y. Using equation (14), we have

$$4x - 2y = 0 \quad \text{or} \quad y = 2x$$

Using this value of y in the constraining equation $x + 2y = 1$, we obtain

$$x + 4x = 1 \quad \text{or} \quad x = \frac{1}{5}$$

and $y = 2/5$. Thus the point $(x, y) = (1/5, 2/5)$ gives the stationary point or critical point. However, this procedure becomes very tedious for function of more than two variables.

1. Lagrange Method

A systematic way to solve such problems is the method of Lagrange undetermined multiplier. Since we have a function of two variable $f(x, y)$, we know from elementary calculus that

$$df = \frac{\partial f}{\partial x}dx + \frac{\partial f}{\partial y}dy$$

Since x and y are not independent variable, we cannot put $f_x = 0$ and $f_y = 0$. We also have

$$dg = \frac{\partial g}{\partial x}dx + \frac{\partial g}{\partial y}dy = 0$$

Now to maximize this function we will define a new constant λ, also known as **Lagrange undermined multiplier**. Multiplying dg by an unknown number λ and subtracting (addition or subtraction does not change the final result) it from df we obtain

$$d(f - \lambda g) = \left(\frac{\partial f}{\partial x} - \lambda\frac{\partial g}{\partial x}\right)dx + \left(\frac{\partial f}{\partial y} - \lambda\frac{\partial g}{\partial y}\right)dy = 0$$

Now choose λ such that

$$\frac{\partial f}{\partial x} - \lambda\frac{\partial g}{\partial x} = 0, \tag{15}$$

$$\frac{\partial f}{\partial y} - \lambda\frac{\partial g}{\partial y} = 0, \tag{16}$$

where λ is such that it satisfies the given constraint $g(x, y) = c$. Let us try to solve Ex (1) using method of Lagrange undetermined multiplier. We have $f(x, y) = x^2 + y^2$ and $g(x, y) = x + 2y = 1$. From (15) and (16), we have

$$\begin{aligned} 2x - \lambda &= 0, \\ 2y - 2\lambda &= 0. \end{aligned}$$

From last two equations we have $x = \lambda/2$ and $y = \lambda$. Now choose λ such that $g(x, y) = 1$ is satisfied. Thus the constraint

$$x + 2y - 1 = 0$$

is satisfied only when $x = \lambda/2$ and $y = \lambda$ or

$$\frac{\lambda}{2} + 2\lambda = 1$$

which in turn yield $\lambda = 2/5$. Using this value of λ, we have $x = 1/5$ and $y = 2/5$. Thus we obtain the same result using Lagrange undetermined multiplier. Thus one can get exactly the same expression using the method of Lagrange undetermined multiplier. From (15), we have

$$\lambda = \frac{\partial f/\partial x}{\partial g/\partial x}$$

and putting this value in expression (16), we obtain

$$\frac{\partial f}{\partial y} - \frac{\partial g}{\partial y}\frac{\partial f/\partial x}{\partial g/\partial x} = 0$$

which in turn yield

$$\frac{\partial f}{\partial y}\frac{\partial g}{\partial x} - \frac{\partial g}{\partial y}\frac{\partial f}{\partial x} = 0$$

So it is the same expression (14).

Alternatively, one can define

$$F(x,y) = f(x,y) - \lambda g(x,y)$$

This is sometimes written as

$$F(x,y) = f(x,y) + \lambda g(x,y)$$

We will see that the sign does not matter. Then we set

$$\frac{\partial F}{\partial x} = 0 \quad \text{and} \quad \frac{\partial F}{\partial y} = 0,$$

and finally solve these two equations together with the constraint equation to find x and y. Now the requirement that $f(x,y)$ is an extremum and the constraint equation lead to two differential relations:

$$\begin{aligned} df &= \frac{\partial f}{\partial x}dx + \frac{\partial f}{\partial y}dy = 0 \\ dg &= \frac{\partial g}{\partial x}dx + \frac{\partial g}{\partial y}dy = 0 \end{aligned}$$

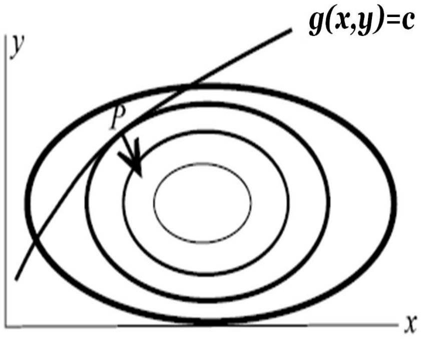
Rearranging and taking ratios, we find that $f_x/g_x = f_y/g_y$, and we call this ratio λ:

$$\frac{f_x}{g_x} = \frac{f_y}{g_y} = \lambda$$

Rearranging again, we obtain the fundamental equations of the method,

$$\frac{\partial f}{\partial x} - \lambda\frac{\partial g}{\partial x} = 0 \quad \text{and} \quad \frac{\partial f}{\partial y} - \lambda\frac{\partial g}{\partial y} = 0$$

again we obtain the same expressions (15) and (16).



B. General Case

For more than one variable, the condition for constrained maxima and minima at any point P becomes

$$\nabla f|_P = \lambda \nabla g|_P$$

Here ∇ is gradient and point P where this equation holds is said to be a critical point. For function of three variable $f = f(x, y, z)$ and $g = g(x, y, z)$ this equation becomes

$$\frac{\partial f}{\partial x} - \lambda \frac{\partial g}{\partial x} = 0, \tag{17}$$

$$\frac{\partial f}{\partial y} - \lambda \frac{\partial g}{\partial y} = 0, \tag{18}$$

$$\frac{\partial f}{\partial z} - \lambda \frac{\partial g}{\partial z} = 0, \tag{19}$$

C. Geometrical Significance of Lagrange Undermined Multiplier

Let us consider a function of two variable $f(x, y)$. As shown in Fig. the closed loops are contours of $f(x, y)$, i.e. curves of $f(x, y) =$ constant, in the xy plane. The bold curve is the graph of the constraint equation $g(x, y) = c$. Assume that f increases toward the center of the figure. The problem can be restated as follows:

- To find an extremum of subject to the constraint $g(x, y) = c$, move along the constraint curve until we arrive at the point P at which the constraint curve is tangent to the local contour of $f(x, y)$.

- One can draw a contour through any point x, y.

- If we move forward or backward from that point along the constraint curve, the value of $f(x, y)$ will decrease from its value at P, so point P is the desired maximum of f.

To show that this condition is equivalent to the Lagrange multiplier method, we first note that if the two curves are tangent at P, the normals to the two curves are parallel at P. To find the normal to a curve, we have

$$\begin{aligned} df &= \frac{\partial f}{\partial x}dx + \frac{\partial f}{\partial y}dy \\ &= \left(\hat{i}\frac{\partial f}{\partial x} + \hat{j}\frac{\partial f}{\partial y}\right) \cdot \left(\hat{i}dx + \hat{j}dy\right) \\ &= \nabla f \cdot d\mathbf{r} = |\nabla f|\,|d\mathbf{r}|\cos\theta \end{aligned}$$

- The direction of ∇f is the direction (corresponding to $\theta = 0$) in which f is increasing most rapidly, i.e. the direction in which the directional derivative df/dr is largest.
- This is the direction normal to the local contour of f. The arrow in the figure represents this direction.
- Similarly, ∇g is a vector normal to the curve $g(x, y) = c$.
- Since $\nabla f|_P$ is perpendicular to the surface. Also, $\nabla g|_P$ is perpendicular to the surface, we have proved $\nabla f|_P$ is parallel to $\nabla g|_P$.

Hence, the curves $f(x, y) = k$ and $g(x, y) = c$ are tangent to each other, ∇f is parallel to ∇g. This means that the two gradient vectors are proportional to each other, with some constant of proportionality:

$$\begin{aligned} \nabla f &= \text{constant} \times \nabla g \\ \hat{i}f_x + \hat{j}f_y &= \text{constant} \times \left(\hat{i}g_x + \hat{j}g_y\right) \end{aligned}$$

If we call the constant λ, the second of these relations can be rearranged to give

$$\begin{aligned} f_x - \lambda g_x &= 0 \\ f_y - \lambda g_y &= 0 \end{aligned}$$

For function of more than two variables, the constrained curve may become the constrained surface. The method of Lagrange undetermined multiplier is easily generalized to functions of more than two variables, as long as the number of constraints is less than the number of variables. For example, to find the extrema of $f(x, y, z)$ subject to two constraints, $g_1(x, y, z) = c_1$ and $g_2(x, y, z) = c_2$, the equations become

$$\begin{aligned}
\frac{\partial f}{\partial x} - \lambda_1 \frac{\partial g_1}{\partial x} - \lambda_2 \frac{\partial g_2}{\partial x} &= 0 \\
\frac{\partial f}{\partial y} - \lambda_2 \frac{\partial g_1}{\partial y} - \lambda_2 \frac{\partial g_2}{\partial y} &= 0 \\
\frac{\partial f}{\partial x} - \lambda_1 \frac{\partial g_1}{\partial z} - \lambda_2 \frac{\partial g_2}{\partial z} &= 0,
\end{aligned}$$

where λ_1 and λ_2 are Lagrange undetermined multiplier. These three equations, together with the two constraints, may be solved to give λ_1, λ_2, x, y and z.

- The Lagrange method of undetermined multiplier is used to find the stationary points of function of more than two variables, subject to several constraints. Also number of constraint is smaller than number of variables.

D. Case of Several Constraints

If a surface S is defined by a number of constraints, namely,

$$\begin{aligned}
g_1(x_1, x_2, \ldots, x_n) &= c_1 \\
&\vdots \\
g_k(x_1, x_2, \ldots, x_n) &= c_k
\end{aligned}$$

Then f has a maximum or minimum at x_0 on S, there must exist constraints $\lambda_1, \lambda_2, \ldots, \lambda_k$ such that

$$\nabla f|_{x_0} = \lambda_1 \, \nabla g_1|_{x_0} + \lambda_2 \, \nabla g_2|_{x_0} + \cdots + \lambda_k \, \nabla g_k|_{x_0}$$

Here $\nabla g_1|_{x_0} \cdots \nabla g_k|_{x_0}$ are linearly independent.

XIV. KKT OPTIMALITY CONDITIONS FOR INEQUALITY CONSTRAINTS

Let us minimize a function $f(\mathbf{x})$ subject to the constraints

$$\begin{aligned}
\min f(\mathbf{x}) & & f : R^n &\to R \\
\text{Subject to } g(\mathbf{x}) &= 0 & h : R^n &\to R^m, \quad m \leq n \\
h(\mathbf{x}) &\leq 0 & g : R^n &\to R^p, \quad p \leq n
\end{aligned}$$

KKT Condition: Let $\mathbf{x}^*$ be a local minimizer of the above equation. Then there exists a set of Lagrange multipliers $\mu_1, \ldots, \mu_p \in R$ and $\lambda_1, \ldots, \lambda_q \in R$ such that

$$\begin{aligned} g_i(\mathbf{x}^*) &= 0, \qquad i = 1, 2, \ldots, p \\ h_j(\mathbf{x}^*) &\leq 0, \qquad j = 1, 2, \ldots, q \\ \lambda_j &\geq 0, \qquad j = 1, 2, \ldots, q \\ \lambda_j h_j(\mathbf{x}^*) &= 0 \end{aligned}$$

$$\nabla f(\mathbf{x}^*) + \sum_{i=1}^{p} \mu_i \nabla g_i(\mathbf{x}^*) + \sum_{j=1}^{q} \lambda_j \nabla h_j(\mathbf{x}^*) = 0$$

These are called the Karush-Kuhn-Tucker (KKT) conditions, with the fourth condition known as the complementary slackness condition. We have so far seen that KKT conditions are necessary for local optimality of a solution to a constrained optimization problem. It turns out that, under specific assumptions on the function f being optimized and the constraint functions g_i's and h_j's, these conditions are also sufficient for optimality. This is the case, for example, when f is a convex function and each g_i and h_j is affine.

Ex: Let us consider the following inequality constraint problem

$$\begin{aligned} f(\mathbf{x}) &= x_1 + x_2 \\ h(\mathbf{x}) &= x_1^2 + x_2^2 - 1 \leq 0 \end{aligned}$$

Then using the KKT condition

$$\begin{aligned} \frac{\partial f}{\partial x_1} + \lambda \frac{\partial h}{\partial x_1} &= 0 \\ \frac{\partial f}{\partial x_2} + \lambda \frac{\partial h}{\partial x_2} &= 0 \end{aligned}$$

From here, we obtain

$$\begin{aligned} 1 + 2\lambda x_1 &= 0 \\ 1 + 2\lambda x_2 &= 0 \end{aligned}$$

Then using the fourth KKT condition, we obtain

$$\lambda h(\mathbf{x}^*) = \lambda(x_1^2 + x_2^2 - 1) = 0$$

This in turn implies that

$$x_1^2 + x_2^2 = 1$$

and hence

$$\lambda = \frac{1}{\sqrt{2}}$$

XV. SUPPORT VECTOR MACHINES

In case of logistic regression, our hypothesis was

$$h_{\boldsymbol{\alpha}}(\mathbf{x}) = g(\boldsymbol{\alpha}^T\mathbf{x}) = \frac{1}{1 + e^{-\boldsymbol{\alpha}^T\mathbf{x}}}$$

Then we found the probability

$$\begin{aligned} p(y = 1|\mathbf{x}, \boldsymbol{\alpha}) &= h_{\boldsymbol{\alpha}}(\mathbf{x}) \\ p(y = 0|\mathbf{x}, \boldsymbol{\alpha}) &= 1 - h_{\boldsymbol{\alpha}}(\mathbf{x}) \end{aligned}$$

These two equation can be compactly written as

$$p(y|\mathbf{x}, \boldsymbol{\alpha}) = [h_{\boldsymbol{\alpha}}(\mathbf{x})]^y [1 - h_{\boldsymbol{\alpha}}(\mathbf{x})]^{1-y}$$

Then we maximize the log-likelihood function to obtain the parameters (using stochastic gradient descent method). Since

$$h_{\boldsymbol{\alpha}}(\mathbf{x}) = g(\boldsymbol{\alpha}^T\mathbf{x})$$

We would predict 1 on input $\mathbf{x}$ if

$$h_{\boldsymbol{\alpha}}(\mathbf{x}) \geq 0.5$$

This in turn implies that

$$\boldsymbol{\alpha}^T\mathbf{x} \geq 0$$

The larger the $\boldsymbol{\alpha}^T\mathbf{x}$ is, the larger also is

$$h_{\boldsymbol{\alpha}}(\mathbf{x}) = p(y = 1|\mathbf{x}, \boldsymbol{\alpha})$$

Thus our classification would be a confident one if

$$\begin{aligned} y &= 1 \quad \text{if} \quad \boldsymbol{\alpha}^T\mathbf{x} >> 0 \\ y &= 0 \quad \text{if} \quad \boldsymbol{\alpha}^T\mathbf{x} << 0 \end{aligned}$$

Support vector machines are an example of a linear two-class classifier. For convenience we assume the labels are $+1$ (positive examples) or -1 (negative examples). In what follows boldface $\mathbf{x}$ denotes a vector with components x_i. The notation $\mathbf{x}_i$ will denote the ith vector in a data set

$$\{x_i, y_i\}, \qquad i = 1, 2, \ldots$$

where y_i is the label ($y \in \{-1, 1\}$) associated with $\mathbf{x}_i$. The objects $\mathbf{x}_i$ are called patterns or examples. The dot product between two vectors, also referred to as an inner product or scalar product, defined as

$$\mathbf{w}^T\mathbf{x} = \sum_i w_i x_i$$

A linear classifier is based on a linear discriminant function of the form

$$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$$

The vector $\mathbf{w}$ is known as the weight vector, and b is called the bias. Consider the case $b = 0$ first. The set of points $\mathbf{x}$ such that

$$\mathbf{w}^T\mathbf{x} = 0$$

are all points that are perpendicular to $\mathbf{w}$ and go through the origin – a line in two dimensions, a plane in three dimensions, and more generally, a hyperplane. The bias b translates the hyperplane away from the origin (plays the role of α_0 in logistic regression). The hyperplane

$$\{\mathbf{x} : \; g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = 0\}$$

divides the space into two.

- The sign of the discriminant function $g(\mathbf{x})$ denotes the side of the hyperplane a point is on.

- The boundary between regions classified as positive and negative is called the decision boundary of the classifier.

- The decision boundary defined by a hyperplane is said to be linear because it is linear in the input examples.

- A classifier with a linear decision boundary is called a linear classifier. Conversely, when the decision boundary of a classifier depends on the data in a non-linear way, the classifier is said to be non-linear.

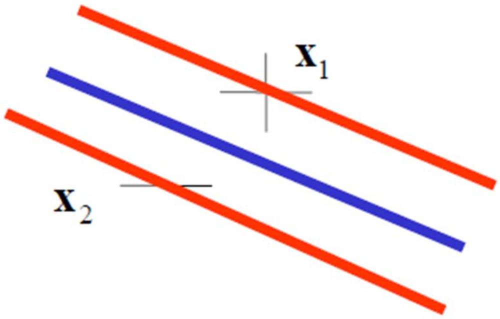


FIG. 7. Positive and Negative Examples.

Now classify unknown $\mathbf{x}$ as

$$g(\mathbf{x}) = \mathrm{sign}(\mathbf{w}^T\mathbf{x} + b)$$

where

$$g(z) = \begin{cases} 1 & \text{if} \quad z \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

Then, constrain, for all plus sample vectors:

$$g(\mathbf{x}_+) = \mathbf{w}^T\mathbf{x}_+ + b \geq 1$$

And for all minus sample vectors

$$g(\mathbf{x}_-) = \mathbf{w}^T\mathbf{x}_- + b \leq -1$$

A. Distance Between The Street

First we will show that $\mathbf{w}$ is perpendicular to the plane. We know that

$$\begin{aligned} \mathbf{w}^T\mathbf{x}_n + b &= +1, \qquad n = 1, 2, \ldots, N \\ \mathbf{w}^T\mathbf{x}_n + b &= -1 \end{aligned}$$

Take two points $\mathbf{x}$ and $\mathbf{x}''$ on the plane. Then

$$\begin{aligned} \mathbf{w}^T\mathbf{x} + b &= 0 \\ \mathbf{w}^T\mathbf{x}'' + b &= 0 \end{aligned}$$

Therefore

$$\mathbf{w}^T(\mathbf{x} - \mathbf{x}'') = 0$$

Hence $\mathbf{w}$ is perpendicular to every vector on the plane. Now take any point $\mathbf{x}$ on the plane and compute the distance between $\mathbf{x}$ and $\mathbf{x}_n$. Since

$$\widehat{\mathbf{w}} = \frac{\mathbf{w}}{||\mathbf{w}||}$$

Hence

$$\begin{aligned}\text{Distance} &= |\widehat{\mathbf{w}}^T(\mathbf{x}_n - \mathbf{x})| \\ &= \frac{1}{||\mathbf{w}||}|\widehat{\mathbf{w}}^T\mathbf{x}_n - \widehat{\mathbf{w}}^T\mathbf{x}| \\ &= \frac{1}{||\mathbf{w}||}|\widehat{\mathbf{w}}^T\mathbf{x}_n + b - \widehat{\mathbf{w}}^T\mathbf{x} - b| = \frac{1}{||\mathbf{w}||}\end{aligned}$$

Hence

$$\text{Distance} = \max\frac{1}{||\mathbf{w}||}$$

With the condition that there are no data points between planes. Thus the final problem can be written as

$$\max\frac{1}{||\mathbf{w}||} \qquad \text{subject to} \qquad \min_{n=1,2,\ldots,N}|\widehat{\mathbf{w}}^T\mathbf{x} + b| = 1$$

Note that constraint has minimum in it. Instead of maximizing this quantity, we will minimize

$$\min\frac{||\mathbf{w}||^2}{2}$$

Squaring is done for mathematical convenience. In order to maximize the margin, we thus need to minimize $\frac{||\mathbf{w}||^2}{2}$. The constraint now is

$$y_n\left(\mathbf{w}^T\mathbf{x}_n + b\right) \geq 1$$

Where y_n is 1 for pluses and -1 for minuses. Here

$$\begin{aligned}\mathbf{w}^T\mathbf{x}_n + b &\geq +1 \qquad \text{when} \qquad y_n = +1 \\ \mathbf{w}^T\mathbf{x}_n + b &\leq -1 \qquad \text{when} \qquad y_n = -1\end{aligned}$$

From here we get

$$y_n\left(\mathbf{w}^T\mathbf{x}_n + b\right) - 1 \geq 0, \qquad n = 1, 2, \ldots, N$$

and for points in the streets

$$y_n \left(\mathbf{w}^T \mathbf{x}_n + b\right) - 1 = 0$$

- Support vectors are the data points that lie closest to the decision surface (or hyperplane).
- SVM maximize the margin around the separating hyperplane.
- Support vectors are the elements of the training set that would change the position of the dividing hyperplane if removed.
- Support vectors are the critical elements of the training set.
- This becomes a Quadratic programming problem that is easy to solve by standard methods.

B. Quadratic Programming Problem

Minimize $\frac{||\mathbf{w}||^2}{2}$ such that

$$y_n \left(\mathbf{w}^T \mathbf{x}_n + b\right) - 1 \geq 0$$

This is a constrained optimization problem. It can be solved by the Lagrangian multiplier method. The Lagrangian is

$$L_P = \frac{||\mathbf{w}||^2}{2} - \sum_{n=1}^{N} \alpha_n \left[y_n \left(\mathbf{w}^T \mathbf{x}_n + b\right) - 1\right]$$

where $\alpha_n \geq 0$. The right hand side of this equation may be written as

$$L_P = \frac{||\mathbf{w}||^2}{2} - \sum_{n=1}^{N} \alpha_n y_n \left(\mathbf{w}^T \mathbf{x}_n + b\right) + \sum_{n=1}^{N} \alpha_n$$

Here N is number of training points. From the property of derivatives

$$\begin{aligned} \frac{\partial L_P}{\partial \mathbf{w}} &= \mathbf{w} - \sum_n \alpha_n y_n \mathbf{x}_n = 0 \\ \frac{\partial L_P}{\partial b} &= \sum_n \alpha_n y_n = 0 \end{aligned} \tag{20}$$

From here we get

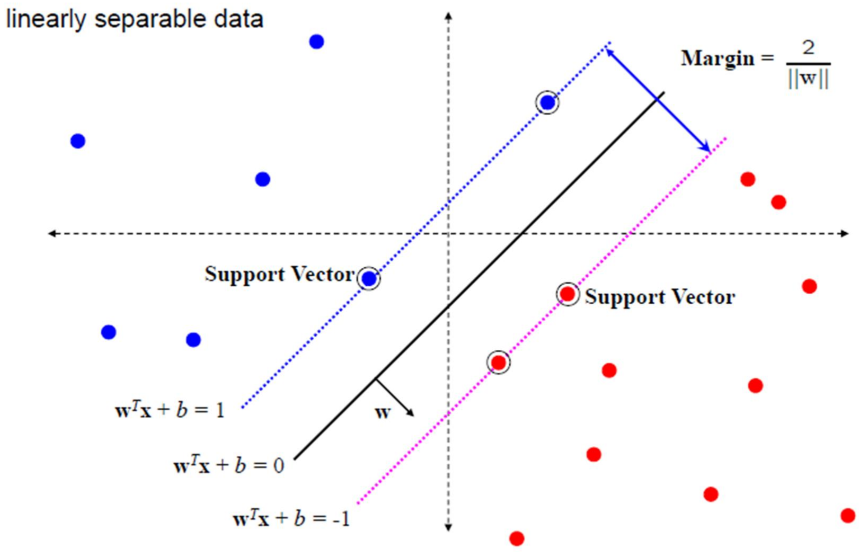

linearly separable data
Margin = $\frac{2}{||\mathbf{w}||}$
Support Vector
Support Vector
$\mathbf{w}^T\mathbf{x} + b = 1$
$\mathbf{w}$
$\mathbf{w}^T\mathbf{x} + b = 0$
$\mathbf{w}^T\mathbf{x} + b = -1$

FIG. 8. Support vector machine and separation of margin.

$$\mathbf{w} = \sum_n \alpha_n y_n \mathbf{x}_n$$
$$\sum_n \alpha_n y_n = 0$$

We will actually solve the optimization problem by now solving for the dual of this original problem.

C. The Lagrangian Dual Problem

Instead of minimizing over $\mathbf{w},\ b$ subject to constraints involving α's, we can maximize over α (the dual variable) subject to the relations obtained previously for $\mathbf{w}$ and b.

- Our solution must satisfy the relations (20).
- By substituting for $\mathbf{w}$ and b back in the original equation we can get rid of the dependence on $\mathbf{w}$ and b.
- Note that weights $\mathbf{w}$ are a linear combination of the training inputs and the training outputs, $\mathbf{x}_n$ and y_n and the values of α. We will now solve for the α's by differentiating the dual problem wrt α, and setting it to zero.
- Most of the α's will turn out to have the value zero. The non-zero α's will correspond to the support vectors.

Primal Problem: We have to minimize

$$L_P = \min \frac{||\mathbf{w}||^2}{2} - \sum_{n=1}^{N} \alpha_n y_n \left(\mathbf{w}^T\mathbf{x}_n + b\right) + \sum_{n=1}^{N} \alpha_n, \qquad \alpha_n \geq 0$$

with

$$\mathbf{w} \ = \sum_n \alpha_n y_n \mathbf{x}_n \quad \text{and} \quad \sum_n \alpha_n y_n = 0$$

Then dual problem can be written as

$$\begin{aligned} L_P \ &= \min_{\mathbf{w},\ b} \ \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{n=1}^{N} \alpha_n y_n \left(\mathbf{w}^T\mathbf{x}_n + b\right) + \sum_{n=1}^{N} \alpha_n, \qquad \alpha_n \geq 0 \\ &= \min_{\mathbf{w},\ b} \ \frac{1}{2}\sum_n \mathbf{w}^T \alpha_n y_n \mathbf{x}_n - \sum_n \mathbf{w}^T \alpha_n y_n \mathbf{x}_n + \sum_n \alpha_n \end{aligned}$$

This in turn can be converted to dual problem as

$$\begin{aligned} L_D \ &= \max_{\boldsymbol{\alpha}} \sum_{n=1}^{N} \alpha_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} y_n y_m \alpha_n \alpha_m \mathbf{x}_n^T \mathbf{x}_m \\ &= \min_{\boldsymbol{\alpha}} \ \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} y_n y_m \alpha_n \alpha_m \mathbf{x}_n^T \mathbf{x}_m - \sum_{n=1}^{N} \alpha_n \\ &= \min_{\boldsymbol{\alpha}} \ \frac{1}{2}\boldsymbol{\alpha}^T \begin{bmatrix} y_1 y_1 \mathbf{x}_1^T\mathbf{x}_1 & y_1 y_2 \mathbf{x}_1^T\mathbf{x}_2 & \cdots & y_1 y_N \mathbf{x}_1^T\mathbf{x}_N \\ y_2 y_1 \mathbf{x}_2^T\mathbf{x}_1 & y_2 y_2 \mathbf{x}_2^T\mathbf{x}_2 & \cdots & y_2 y_N \mathbf{x}_2^T\mathbf{x}_N \\ \vdots & \vdots & \ddots & \vdots \\ y_N y_1 \mathbf{x}_N^T\mathbf{x}_1 & y_N y_2 \mathbf{x}_N^T\mathbf{x}_2 & \cdots & y_N y_N \mathbf{x}_N^T\mathbf{x}_N \end{bmatrix} \boldsymbol{\alpha} - \mathbf{1}^T\boldsymbol{\alpha} \end{aligned}$$

subject to

$$\mathbf{y}^T\boldsymbol{\alpha} = 0 \qquad \mathbf{0} \leq \boldsymbol{\alpha} < \infty$$

Note that we have removed the dependence on $\mathbf{w}$ and b.

- The Karush-Kuhn-Tucker theorem: the solution we find here will be the same as the solution to the original problem.

- But why are we doing this? Why not just solve the original problem?

- Because this will let us solve the problem by computing the just the inner products of $\mathbf{x}_n,\ \mathbf{x}_m$ (which will be very important later on when we want to solve non-linearly separable classification problems).

Remark: Let's take a more careful look at equation

$$\mathbf{w} \;=\; \sum_n \alpha_n y_n \mathbf{x}_n$$

which gives the optimal value of $\mathbf{w}$ in terms of the optimal value of $\boldsymbol{\alpha}$. Suppose we've fit our model's parameters to a training set, and now wish to make a prediction at a new point input $\mathbf{x}$. We would then calculate $\mathbf{w}^T\mathbf{x} + b$, and predict $y = 1$ if and only if this quantity is bigger than zero. This equation can also be written:

$$\begin{aligned} \mathbf{w}^T\mathbf{x} + b \;&=\; \left(\sum_n \alpha_n y_n \mathbf{x}_n\right)^T \mathbf{x} + b \\ &=\; \sum_n \alpha_n y_n (\mathbf{x}_n \cdot \mathbf{x}) + b \qquad (21) \end{aligned}$$

Hence, if we've found the α_n's, in order to make a prediction, we have to calculate a quantity that depends only on the inner product between $\mathbf{x}$ and the points $\mathbf{x}_n$ in the training set. Moreover, we saw earlier that the α_n's will all be zero except for the support vectors. Thus, many of the terms in the sum above will be zero, and we really need to find only the inner products between $\mathbf{x}$ and the support vectors (of which there is often only a small number) in order to calculate (21) and make our prediction.

1. The Quadratic Programming

The dual problem can be written as

$$L_D(\boldsymbol{\alpha}) = \min_{\boldsymbol{\alpha}} \; \frac{1}{2}\boldsymbol{\alpha}^T Q \boldsymbol{\alpha} - \mathbf{1}^T\boldsymbol{\alpha} \quad \text{subject to} \quad \mathbf{y}^T\boldsymbol{\alpha} = 0, \quad \boldsymbol{\alpha} \geq 0$$

The quadratic programming will give us $\alpha_1, \alpha_2, \ldots, \alpha_N$. Because of the KKT condition most of the $\boldsymbol{\alpha}$'s are zero. Then we can find

$$\mathbf{w} = \sum_{\mathbf{x}_n \in \text{SV}} \alpha_n y_n \mathbf{x}_n$$

Then solve for b using any support vector using the equation

$$y_n(\mathbf{w}^T\mathbf{x}_n + b) = 1$$

D. Inner Product and Nonlinear Transformation

We have

$$\begin{aligned} L_D(\boldsymbol{\alpha}) &= \sum_{n=1}^{N} \alpha_n - \frac{1}{2} y_n y_m \alpha_n \alpha_m \mathbf{x}_n^T \mathbf{x}_m \\ &= \sum_{n=1}^{N} \alpha_n - \frac{1}{2} y_n y_m \alpha_n \alpha_m (\mathbf{x}_n \cdot \mathbf{x}_m) \end{aligned}$$

The claim is that this function will be maximized if we give nonzero values to $\boldsymbol{\alpha}$'s that correspond to the support vectors, ie, those that matter in fixing the maximum width margin. Note first from the constraint condition that all the $\boldsymbol{\alpha}$'s are positive (KKT condition). Now let's think about a few cases.

1. If two features $\mathbf{x}_i$, $\mathbf{x}_j$ are completely dissimilar (orthogonal), their dot product is 0, and they don't contribute to L_D.

2. If two features $\mathbf{x}_i$, $\mathbf{x}_j$ are completely alike, their dot product is 0. There are 2 subcases.

   Subcase 1: Both $\mathbf{x}_i$ and $\mathbf{x}_j$ predict the same output value y_i (either $+1$ or -1). Then $y_i y_j$ is always 1, and the value of $\alpha_i \alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j$ will be positive. But this would decrease the value of L (since it would subtract from the first term sum). So, the algorithm downgrades similar feature vectors that make the same prediction.

   Subcase 2: $\mathbf{x}_i$ and $\mathbf{x}_j$ make opposite predictions about the output value y_i (ie, one is $+1$, the other -1), but are otherwise very closely similar: then the product $\alpha_i \alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j$ is negative and we are subtracting it, so this adds to the sum, maximizing it. This is precisely the examples we are looking for: the critical ones that tell the two classes apart.

E. Nonlinear SVM

In this section we are going to discuss the nonlinearly separable case. For nonlinear case, we get linear separation by mapping the data to a higher dimensional space. The following set can't be separated by a linear function, but can be separated by a quadratic one (see Fig. 11). So if we map

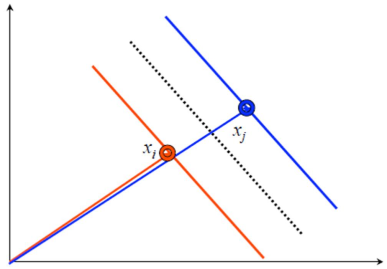


FIG. 9. Two similar $\mathbf{x}_i,\ \mathbf{x}_j$ vectors that predict different classes tend to maximize the width.

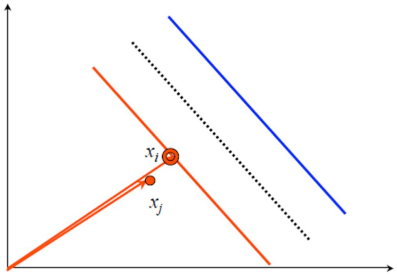


FIG. 10. Two vectors that are similar but predict the same class are redundant.

$$\mathbf{x} \ \longmapsto\ \{\mathbf{x}^2,\ \mathbf{x}\}$$

we gain linear separation. Thus the idea is to make a transformation from $\mathbf{X}$ space to $\mathbf{Z}$ space. So the Lagrangian finally becomes

$$\begin{aligned} L_D(\boldsymbol{\alpha}) &= \max\ \sum_{n=1}^{N}\alpha_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} y_n y_m \alpha_n \alpha_m \mathbf{x}_n^T\mathbf{x}_m \\ &= \max\ \sum_{n=1}^{N}\alpha_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} y_n y_m \alpha_n \alpha_m \mathbf{z}_n^T\mathbf{z}_m, \qquad \alpha_n \geq 0 \end{aligned}$$

subject to

$$\sum_{n=1}^{N}\alpha_n y_n = 0$$

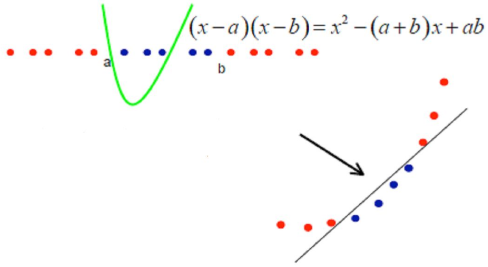


FIG. 11. Nonlinear separable case.

The functional margin of $(\mathbf{w}, b)$ with respect to the training example in this case becomes

$$g(\mathbf{z}) = \text{sign}(\mathbf{w}^T\mathbf{z} + b)$$

where

$$\mathbf{w} = \sum_{\mathbf{z}_n \in \text{SV}} \alpha_n y_n \mathbf{z}_n$$

and to find b, take any support vector

$$y_m(\mathbf{w}^T\mathbf{z}_m + b) = 1$$

F. Inner Product and Kernel Method

Given two points $\mathbf{x},\ \mathbf{x}' \in \mathbf{X}$, we define

$$\mathbf{z}^T\mathbf{z}' = K(\mathbf{x}, \mathbf{x}')$$

- If there is a kernel function K such that $K(\mathbf{x}, \mathbf{x}') = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}')$, then we do not need to know or compute Φ at all!!
- Note that the kernel function defines inner products in the transformed space. Or, it defines similarity in the transformed space.

So, the function we end up optimizing is:

$$L_D(\boldsymbol{\alpha}) = \max\ \sum_{n=1}^{N} \alpha_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} y_n y_m \alpha_n \alpha_m K(\mathbf{x}_n, \mathbf{x}_m)$$

Ex: Let

$$\mathbf{x} = (x_1, x_2)$$

and

$$\mathbf{z} = \Phi(\mathbf{x}) = (1, x_1, x_2, x_1^2, x_2^2, x_1x_2)$$

Then

$$\begin{aligned} K(\mathbf{x},\ \mathbf{x}') &= \mathbf{z}^T\mathbf{z}' = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}') = \begin{pmatrix} 1 & x_1 & x_2 & x_1^2 & x_2^2 & x_1x_2 \end{pmatrix} \begin{pmatrix} 1 \\ x_1' \\ x_2' \\ x_1'^2 \\ x_2'^2 \\ x_1'x_2' \end{pmatrix} \\ &= 1 + x_1x_1' + x_2x_2' + x_1^2x_1'^2 + x_2^2x_2'^2 + x_1x_1'x_2x_2' \end{aligned}$$

The above equation may also be written as (for $\mathbf{x} = (x_1, x_2)$)

$$\begin{aligned} K(\mathbf{x}, \mathbf{x}') &= (1 + \mathbf{x}^T\mathbf{x}')^2 = (1 + x_1x_1' + x_2x_2')^2 \\ &= 1 + 2x_1x_1' + 2x_2x_2' + x_1^2x_1'^2 + x_2^2x_2'^2 + 2x_1x_1'x_2x_2' \end{aligned}$$

This is again an inner product. Note that the $\Phi(x)$ in this case becomes

$$\mathbf{z} = \Phi(\mathbf{x}) = (1, \sqrt{2}\ x_1, \sqrt{2}\ x_2, x_1^2, x_2^2, \sqrt{2}\ x_1x_2)$$

This is an example of polynomial kernel. In general

$$K(\mathbf{x}, \mathbf{x}') = (1 + \mathbf{x}^T\mathbf{x}')^Q$$

can also be written in scale adjusted form as

$$K(\mathbf{x}, \mathbf{x}') = (a\ \mathbf{x}^T\mathbf{x}' + b)^Q$$

Kernel of this form does corresponds to inner product in higher space. Evaluating K only requires one addition and one exponentiation more than the original dot product.

- Intuitively, if $\Phi(\mathbf{x})$ and $\Phi(\mathbf{x}')$ are close together, then we might expect $K(\mathbf{x}, \mathbf{x}') = \Phi(\mathbf{x})^T\Phi(\mathbf{x}')$ to be large.

- Conversely, if $\Phi(\mathbf{x})$ and $\Phi(\mathbf{x}')$ are far apart – say nearly orthogonal to each other – then $K(\mathbf{x}, \mathbf{x}') = \Phi(\mathbf{x})^T\Phi(\mathbf{x}')$ will be small. So, we can think of $K(\mathbf{x}, \mathbf{x}')$ as some measurement of how similar are $\Phi(\mathbf{x})$ and $\Phi(\mathbf{x}')$, or of how similar are $\mathbf{x}$ and $\mathbf{x}'$.

G. Gaussian Kernel

Let

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{|\mathbf{x} - \mathbf{x}'|^2}{2\sigma^2}\right)$$

This is a reasonable measure of $\mathbf{x}$ and $\mathbf{x}'$'s similarity, and is close to 1 when $\mathbf{x}$ and $\mathbf{x}'$ are close, and near 0 when $\mathbf{x}$ and $\mathbf{x}'$ are far apart. Can we use this definition of K as the kernel in an SVM? In this particular example, the answer is yes. This kernel is called the Gaussian kernel, and corresponds to an infinite dimensional feature mapping Φ.

H. Infinite Dimensional Case – Radial Basis Function

Let us consider an inner product in infinite dimensional $\mathbf{Z}$ space.

$$K(\mathbf{x}, \mathbf{x}') = e^{-\gamma|\mathbf{x}-\mathbf{x}'|^2}$$

This kernel (also called as radial basis function) corresponds to inner product in infinite dimensional space. Let us consider the one dimensional case

$$\begin{aligned} K(\mathbf{x}, \mathbf{x}') &= e^{-|x-x'|^2} = e^{-x^2}e^{-x'^2}e^{2xx'} \\ &= e^{-x^2}e^{-x'^2}\sum_{k=0}^{\infty}\frac{(\sqrt{2})^k(\sqrt{2})^k(x)^k(x')^k}{k!} \end{aligned}$$

This is an inner product in infinite dimensional space.

I. Regularization and the Non-Separable Case

[1] Harikrishna Narasimhan, http://drona.csa.iisc.ernet.in/~e0270/Jan-2015/Tutorials/lecture-notes-2.pdf.

[2] Harikrishna Narasimhan, http://drona.csa.iisc.ernet.in/~e0270/Jan-2015/Tutorials/lecture-notes-3.pdf.

[3] James Stewart, *Calculus*_6th Ed.

[4] Asa Ben-Hur and Jason Weston, *A User's Guide to Support Vector Machines* http://pyml.sourceforge.net/doc/howto.pdf.

Random Search Methods

XVI. RANDOM SEARCH METHODS

In this lecture we will discuss some of the random search method to find the global minima of the objective function. We discuss various search methods that attempts to search throughout the entire feasible set. These methods use only objective function values and do not require derivatives.

- These methods can be used to generate good initial points for the iterative methods discussed in earlier chapters.

A. Naïve Random Search Method

A randomized search method, also called a probabilistic search method, is an algorithm that searches the feasible set of an optimization problem by considering randomized samples of candidate points in the set. Suppose that we wish to solve an optimization problem

$$\min_{\mathbf{x}\in\mathcal{C}} f(x)$$

Typically, we start a randomized search process by selecting a random initial point $\mathbf{x}_0 \in \mathcal{C}$. Then, we select a random next candidate point, usually close to $\mathbf{x}_0$.

- We assume that for any $\mathbf{x} \in \mathcal{C}$, there is a set $N(\mathbf{x}) \subset \mathcal{C}$ such that we can generate a random sample from this set. Typically, $N(\mathbf{x})$is a set of points that are close to $\mathbf{x}$, and for this reason we usually think of $N(\mathbf{x})$ as a neighborhood of $\mathbf{x}$.
- Since we are generating random number from the set $N(\mathbf{x})$, we have to specify the distribution of $N(\mathbf{x})$. Often, this distribution is chosen to be uniform over $N(\mathbf{x})$, other distributions are often used, including Gaussian and Cauchy.

The algorithm of naïve random search is described below:

- Set $i = 0$. Select an initial point $\mathbf{x}_0 \in \mathcal{C}$
- Pick a candidate point $\mathbf{z}_i$ at random from $N(\mathbf{x})$.

- If $f(\mathbf{z}_i) < f(\mathbf{x}_i)$, then set

  $$\mathbf{x}_{i+1} = \mathbf{z}_i$$

  else set

  $$\mathbf{x}_{i+1} = \mathbf{x}_i$$

- If stopping criterion satisfied, then stop. Otherwise

- Set $i = i + 1$, go to step 2.

The update rule in random search method is

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{d}_i$$

Here the direction $\mathbf{d}_i$ is randomly generated. Also, by design the direction $\mathbf{d}_i$ is either 0 or is a descent direction.

> The random search may get stuck in a region around the local minima.

B. Simulated Annealing Algorithm

The simulation annealing method aims to find the global minima of the function. Unlike random search method, the simulated annealing method also search for the point outside of local minima. We modify the naïve search algorithm so that we can climb out of local minima region. This means that the algorithm may accept a new point that is worse than the current point. Again, in simulated annealing method we have

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{d}_i$$

But in simulated annealing the direction $\mathbf{d}_i$ might be an ascent direction.

> Annealing is referred to as tempering certain alloys of metal, glass, or crystal by heating above its melting point, holding its temperature, and then cooling it very slowly until it solidifies into a perfect crystalline structure. The defect-free crystal state corresponds to the global minimum energy configuration.

- Simulated annealing is variant of the **Metropolis algorithm**, where temperature changes from high to low.
- Simulated annealing is basically composed of two stochastic processes: one process for the generation of solutions and the other for the acceptance of solutions.
- Simulated annealing is a descent algorithm modified by random ascent moves in order to escape local minima which are not global minima.
- Simulated annealing is a general purpose, serial algorithm for finding a global minimum for a continuous function. It is also a popular Monte Carlo algorithm for any optimization problem.

C. Boltzman Equation

The probability of physical system being in state i with energy E_i is given by

$$P_i = \frac{1}{Z} e^{-E_i/k_B T}$$

where k_B is known as Boltzman' s constant and T is temperature. Here Z is normalization constant and is also known as partition function.

$$Z = \sum_i e^{-E_i/k_B T}$$

- At high T, the Boltzmann distribution exhibits uniform preference for all the states, regardless of the energy.
- When T approaches zero, only the states with minimum energy have nonzero probability of occurrence.
- At high T, the system ignores small changes in the energy and approaches thermal equilibrium rapidly, that is, it performs a coarse search of the space of global states and finds a good minimum.
- As T is lowered, the system responds to small changes in the energy, and performs a fine search in the neighborhood of the already determined minimum and finds a better minimum.

- At $T = 0$, any change in the system states does not lead to an increase in the energy, and thus, the system must reach equilibrium if $T = 0$.

D. Definition of Terms

Let $\mathbf{X}$ be a solution space, i.e., space of all possible solutions. Let

$$f : \mathbf{X} \ \rightarrow \ R$$

be an objective function defined on solution space. Our aim is to find a global minimum $\mathbf{x}^*$. This means that there exist $\mathbf{x}^* \in \mathbf{X}$ such that

$$f(\mathbf{x}) \geq f(\mathbf{x}^*) \qquad \forall \quad \mathbf{x} \in \mathbf{X}$$

Next we define $N(\mathbf{x})$ to be neighborhood for $\mathbf{x} \in \mathbf{X}$. Therefore, associated with every solution, $\mathbf{x} \in \mathbf{X}$, are neighboring solutions, $N(\mathbf{x})$, that can be reached in a single iteration of a local search algorithm.

- Start with initial solution $\mathbf{x} \in \mathbf{X}$.
- Generate the neighboring solution $\mathbf{z} \in N(\mathbf{x})$.
- Based on Metropolis acceptance criterion move from current solution $\mathbf{x} \in \mathbf{X}$ to a candidate solution $\mathbf{z} \in N(\mathbf{x})$.
- The candidate solution, $\mathbf{z}$, is accepted as the current solution based on the acceptance probability

$$P(\text{Accept } \mathbf{z} \text{ as next solution}) = \begin{cases} e^{-\frac{f(\mathbf{z})-f(\mathbf{x})}{T_i}} & \text{if} \quad f(\mathbf{z}) - f(\mathbf{x}) > 0 \\ 1 & \text{if} \quad f(\mathbf{z}) - f(\mathbf{x}) \leq 0 \end{cases}$$

where T_i is temperature parameter at each iteration, such that

$$T_i > 0 \quad \text{for all} \quad i \quad \text{and} \qquad \lim_{i \ \rightarrow \ \infty} T_i = 0$$

If the temperature is reduced sufficiently slowly, then the system can reach an equilibrium (steady state) at each iteration i. The algorithm of simulated annealing is discussed below:

```
1. Set $i = 0$. Select an initial point $\mathbf{x}_0 \in \mathbf{X}$.
2. Pick a candidate point $\mathbf{z}_i$ at random from $N(\mathbf{x}_i)$.
3. Initialize $T$ with a large value and $L$ (The number of iteration at a particular temperature).
4. Evaluate $\Delta E(\mathbf{x}) = f(\mathbf{z}_i) - f(\mathbf{x}_i)$.
        If $\Delta E(\mathbf{x}) < 0$, keep the new state.
        Otherwise, accept the new state with probability $P = e^{-\Delta E/T}$.
        Set $\mathbf{x}_{i+1} = \mathbf{z}_i$ else set $\mathbf{x}_{i+1} = \mathbf{x}_i$.
        until the number of accepted transitions is below a threshold level.
5. Set $T = T - \Delta T$ and also update $L$.
   Until $T$ is small enough.
6. Set $i = i + 2$, go to step 2.
```

The acceptance probability must be chosen appropriately. A typical choice is

$$P = \min\left\{1,\ e^{-\frac{f(\mathbf{z}_i)-f(\mathbf{x}_i)}{T_i}}\right\}$$

Notice that if

$$f(\mathbf{z}_i) \leq f(\mathbf{x}_i)$$

Then

$$P = \min\left\{1,\ e^{-\frac{f(\mathbf{z}_i)-f(\mathbf{x}_i)}{T_i}}\right\} = 1$$

which means that we set

$$\mathbf{x}_{i+1} = \mathbf{z}_i$$

However, if

$$f(\mathbf{z}_i) > f(\mathbf{x}_i)$$

there is still a positive probability of setting

$$\mathbf{x}_{i+1} = \mathbf{z}_i$$

This probability is equal to

$$e^{-\frac{f(\mathbf{z}_i)-f(\mathbf{x}_i)}{T_i}}$$

- Larger the difference $f(\mathbf{z}_i) - f(\mathbf{x}_i)$, the less likely we are to move to the worse point $\mathbf{z}_i$. Similarly, the smaller the value of T_i the less likely we are to move to $\mathbf{z}_i$.

- The temperature T_i monotonically decreases to 0. In other words, as the iteration index i increases, the algorithm becomes increasingly reluctant to move to a worse point.

- Hajek shows that an appropriate cooling schedule is

$$T_i = \frac{T_0}{\ln(i+2)}, \qquad T_0 > 0$$

  where T_0 is large enough to allow the algorithm to climb out of regions around local minimizers that are not global minimizers.

[1] Wei-Ta Chu, https://www.cs.ccu.edu.tw/~wtchu/courses/2012s_OPT/Lectures/Chapter%2014%20Global%20Search%20Algorithms.pdf.

[2] K.-L. Du and M.N.S. Swamy, *Search and Optimization by Metaheuristics*, Springer, 2016.

[3] https://am207.github.io/2017/wiki/lab4.html.

[4] https://pdfs.semanticscholar.org/2726/93df38b60670a8ea788122a7de353a9a7ff0.pdf.

ARCH and GARCH

XVII. ARCH AND GARCH MODELS

In this chapter, we will use some of the previously discussed optimization methods to find the parameters of GARCH(1,1) model. Note that in standard regression model

$$y_i = \alpha_0 + \alpha_1 x_i + \epsilon_i$$

the variance of the residuals, ϵ_i is constant (also known as homoscedastic) and we use ordinary leat square regression to estimate the constants α_0 and α_1. If the variance ϵ_i is not constant (also known as heteroscedastic) then we can use weighted least squares to estimate the regression coefficients. According to ARCH(1,1) model, the log return on the asset at time i is modeled as

$$R_i = \mu_i + \sigma_i \epsilon_i = \mu_i + \varepsilon_i \tag{22}$$

where ϵ_i is a sequence of $N(0, 1)$ i.i.d. random variables and $\varepsilon_i = \sigma_i \epsilon_i$, is also referred to as the shock or innovation of an asset return at time i. The conditional mean and variance of R_i is given by

$$\mu_i = E[R_i|\mathcal{F}_{i-1}] \qquad \sigma_i^2 = \text{Var } [R_i|\mathcal{F}_{i-1}] = E[(R_i - \mu_i)^2|\mathcal{F}_{i-1}] \tag{23}$$

where $\mathcal{F}_{i-1}$ is the information set available at time $i - 1$. Note that

$$\text{Var } (\varepsilon_i) = \sigma_i^2$$

Here the disturbance ε_i is independent of all past and future ε's. Thus there is no serial correlation in ε's, i.e., past price movement gives no information about the sign of the random component of return in period i. Let us define the residual return at time i, $R_i - \mu_i$, as

$$u_i = \sigma_i \epsilon_i = \varepsilon_i$$

Using Eqs. (22) and (23), we can also write

$$\sigma_i^2 = \text{Var } [R_i|\mathcal{F}_{i-1}] = \text{Var } [\varepsilon_i|\mathcal{F}_{i-1}]$$

The positive square root of σ_i is the volatility. The model which study the evolution of σ_i^2 is also known as conditional heteroscedastic model.

A. ARCH Model

In this section we will discuss the ARCH(1) and GARCH (1,1) model. The ARCH(1) model describes the evolution of σ_i^2. In Arch(1) model we assume that

$$u_i = \sigma_i \epsilon_i = \varepsilon_i$$

where $\{\epsilon_i\}$ are iid random variables with mean 0 and variance 1. The evolution equation for variance in ARCH(1) model is written as

$$\sigma_{i+1}^2 = \omega + \alpha u_i^2 = \omega + \alpha \varepsilon_i^2$$

where $\omega > 0$ and $\alpha \geq 0$ to ensure positive variance and $\alpha < 1$ for stationarity. Note that σ_i^2 is a model for volatility of R_i. Thus the basic idea of ARCH model is

- The shock u_i of an asset return is serially uncorrelated, but dependent.

The important result of ARCH model is

1. It can produce volatility clustering.
2. The shocks u_i have heavy tail.

The weakness of ARCH model are

- The volatility in ARCH model depends on square of the previous shocks and hence positive and negative shocks produces the same effect. In reality, the positive and negative shocks have different effects.
- In ARCH model we need large number of parameters to adequately describe the volatility process of asset return.

B. Standardized Residuals

The standardized residuals is defined as

$$\widehat{u}_i = \frac{u_i}{\sigma_i}$$

The sequence $\widehat{u}_i$ is an iid random variables and it can be used to check the adequacy of a fitted ARCH model. The QQ plot of the standardized residuals are used to check the normality of the residuals. The Ljung-Box statistics of $\widehat{u}_i^2$ can be used to test the validity of the volatility equation.

C. GARCH Model

The dynamic variance in GARCH (1,1) variance model is written as

$$\sigma_{i+1}^2 = \omega + \alpha u_i^2 + \beta \sigma_i^2$$

The equation states that conditional variance of tomorrow's return is equal to a constant, plus today's residual squared, plus today's known variance. In this equation $u_i = \sigma_i \epsilon_i$. The common choices for ϵ_i are normal and student's t disturbances. Also

$$\mathbf{u} = (u_1,\ u_2, \ldots, u_n)^T$$

is return on some asset with zero mean. The unconditional, or long-run average, variance, σ^2 turns out to be

$$\sigma^2 = \frac{\omega}{1 - \alpha - \beta}$$

- The GARCH variance model is mean reverting process, i.e., over long time horizon, the conditional variance get pull back to a long-run average level of σ^2.

- The sum $(\alpha + \beta)$ plays a crucial role concerning the forecasting with GARCH models and is commonly called the persistence level/index of the model: a high persistence, $(\alpha + \beta)$ close to 1, implies that shocks which push variance away from its long-run average will persist for a long time, even though eventually the long-horizon forecast will be the long-run average variance, σ^2.

- Also, GARCH(1,1) model is equivalent to ARCH(∞) model.

D. GARCH Parameter Estimation Using Gaussian Errors

To find the parameters of GARCH model, we use maximum likelihood estimation (MLE). MLE for GARCH(1,1) model can be estimated as follows: In GARCH(1,1) model, we have

$$\begin{aligned} u_i &= \mu_i + \sigma_i \epsilon_i = \mu_i + \varepsilon_i \\ \sigma_i^2 &= \omega + \alpha u_{i-1}^2 + \beta \sigma_{i-1}^2 \end{aligned}$$

where $\epsilon_i \sim N(0,1)$ are conditionally i.i.d. normal and $\mu_i = 0$. Since ϵ_i are normal, we take the likelihood function as

$$\begin{aligned} L(\mathbf{u},\boldsymbol{\theta}) &= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(-\frac{(u_i-\mu_i)^2}{2\sigma_i^2}\right) \\ &= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(-\frac{u_i^2}{2\sigma_i^2}\right) \end{aligned}$$

and hence the normal log-likelihood function is

$$l(\mathbf{u},\boldsymbol{\theta}) = \sum_{i=1}^{n} \ln\left[p(u_i|\sigma_i;\boldsymbol{\theta})\right]$$

where p is the probability density and $\boldsymbol{\theta}$ is the parameters of the GARCH(1,1) model. Note that in above equation

$$p(u_i|\sigma_i;\boldsymbol{\theta}) = \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(-\frac{u_i^2}{2\sigma_i^2}\right)$$

Thus we are given u_i and we have to determine $\boldsymbol{\theta} = \{\omega,\ \alpha,\ \beta\}$. Using above equation, we obtain

$$l(\mathbf{u},\boldsymbol{\theta}) \ = \ \ln L(\mathbf{u},\boldsymbol{\theta}) = \sum_{i=1}^{n}\left[-\frac{1}{2}\ln(2\pi) - \frac{1}{2}\ln(\sigma_i^2) - \frac{u_i^2}{2\sigma_i^2}\right]$$

Now to maximize the log-likelihood function, we set

$$\frac{\partial l(\mathbf{u},\boldsymbol{\theta})}{\partial \sigma_i^2} = \sum_{i=1}^{n} -\frac{1}{2\sigma_i^2} + \frac{u_i^2}{2\sigma_i^4} = 0$$

This equation can be written as

$$\frac{\partial l(\mathbf{u},\boldsymbol{\theta})}{\partial \sigma_i^2} = \frac{1}{2}\sum_{i=1}^{n} \frac{1}{\sigma_i^2}\left(\frac{u_i^2}{\sigma_i^2} - 1\right)$$

From here we can see that parameters of the volatility model must be chosen to make $\left(\frac{u_i^2}{\sigma_i^2} - 1\right)$ as close to zero as possible. Now the parameters of the GARCH(1,1) model are $\omega,\ \alpha$ and β and hence we write

$$\frac{\partial l(\mathbf{u},\boldsymbol{\theta})}{\partial \theta_i} = \frac{\partial l(\mathbf{u},\boldsymbol{\theta})}{\partial \sigma_i^2}\frac{\partial \sigma_i^2}{\partial \theta_i}$$

These derivatives can be determined recursively using the relation

$$\begin{aligned} \sigma_i^2 &= \omega + \alpha u_{i-1}^2 + \beta\sigma_{i-1}^2 \\ &= \omega + \alpha u_{i-1}^2 + \beta[\omega + \alpha u_{i-2}^2 + \beta\sigma_{i-2}^2] \end{aligned}$$

Thus

$$\begin{aligned}
\frac{\partial \sigma_i^2}{\partial \omega} &= 1 + \beta \frac{\partial \sigma_{i-1}^2}{\partial \omega} \\
\frac{\partial \sigma_i^2}{\partial \alpha} &= u_{i-1}^2 + \beta \frac{\partial \sigma_{i-1}^2}{\partial \alpha} \\
\frac{\partial \sigma_i^2}{\partial \beta} &= \sigma_{i-1}^2 + \beta \frac{\partial \sigma_{i-1}^2}{\partial \beta}
\end{aligned}$$

Now

$$\frac{\partial \sigma_i^2}{\partial \omega} = 1 + \beta \frac{\partial \sigma_{i-1}^2}{\partial \omega} \approx \frac{1}{1-\beta}$$

Thus using the above equations, we can determine the GARCH(1,1) parameter.

An alternative way to determine GARCH(1,1) model is given in Hull book. According to it we maximize

$$L(\mathbf{u}, \boldsymbol{\theta}) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(-\frac{u_i^2}{2\sigma_i^2}\right)$$

which is equivalent to maximizing

$$l(\mathbf{u}, \boldsymbol{\theta}) = \sum_{i=1}^{n} \left[-\frac{1}{2}\ln(2\pi) - \frac{1}{2}\ln(\sigma_i^2) - \frac{u_i^2}{2\sigma_i^2}\right]$$

also

$$\sigma_i^2 = \omega + \alpha u_{i-1}^2 + \beta \sigma_{i-1}^2$$

Here we search iteratively to find the parameters that maximizes the expression in this equation. Note that we need σ_1^2 for complete definition of $l(\mathbf{u}, \boldsymbol{\theta})$. Thus a reasonable guess of σ_1^2 improves accuracy in finite samples. The exact value of σ_1^2 does not matter in large samples, since σ_i^2 converges to its stationary distribution for large i. A reasonable guess of σ_1^2 is sample unconditional variance.

E. Optimization Method

Newton's method can obtain result of optimization problem. In the case of multidimensional optimization, we seek a zero of the gradient. We know that the iteration scheme for Newton's method has the form

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \frac{\nabla f(\mathbf{x}_i)}{\nabla^2 f(\mathbf{x}_i)}$$

Thus, for the maximum likelihood problem, we have

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}}\ l(\boldsymbol{\theta})$$

We can obtain approximated value of $\boldsymbol{\theta}_i$, after ith iteration as

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i - \frac{\nabla l(\boldsymbol{\theta}_i)}{\nabla^2 l(\boldsymbol{\theta}_i)} = \boldsymbol{\theta}_i - \frac{\nabla l(\boldsymbol{\theta}_i)}{J(\boldsymbol{\theta}_i)}$$

where

$$J = E\left[\nabla^2 l(\boldsymbol{\theta})\right]$$

is the Fisher Information matrix. Now

$$\nabla l(\boldsymbol{\theta}) \ = \ \frac{\partial l(\mathbf{u}, \boldsymbol{\theta})}{\partial \sigma_i^2}\frac{\partial \sigma_i^2}{\partial \boldsymbol{\theta}} = \frac{1}{2}\sum_{i=1}^{n}\frac{1}{\sigma_i^2}\left(\frac{u_i^2}{\sigma_i^2} - 1\right) \times \frac{\partial \sigma_i^2}{\partial \boldsymbol{\theta}}$$

But

$$\frac{\partial \sigma_i^2}{\partial \boldsymbol{\theta}} \ = \ \begin{pmatrix} \frac{\partial \sigma_i^2}{\partial \omega} \\ \frac{\partial \sigma_i^2}{\partial \alpha} \\ \frac{\partial \sigma_i^2}{\partial \beta} \end{pmatrix} = \begin{pmatrix} 1 + \beta\frac{\partial \sigma_{i-1}^2}{\partial \omega} \\ u_{i-1}^2 + \beta\frac{\partial \sigma_{i-1}^2}{\partial \alpha} \\ \sigma_{i-1}^2 + \beta\frac{\partial \sigma_{i-1}^2}{\partial \beta} \end{pmatrix}$$

and Fisher information matrix is

---

[1] Louis H. Ederington and W. Guan, `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.390.9011&rep=rep1&type=pdf`.

[2] Y. Yang, `http://www.ams.stonybrook.edu/~yiyang/research/computational_finance/Parameter_Estimation_of_GARCH_Model.pdf`.

[3] Varun Malik, Ranjit Kumar and Abid Hussain Rather, *Bayesian Estimation of GARCH Coefficients of INR/USD Exchange Rate*, ELK Asia Pacific Journal of Finance and Risk Management, Vol 7, Issue 1, 79-118 (2016)]. `https://www.elkjournals.com/MasterAdmin/UploadFolder/Ranjit-FinanceRanjit-Finance/Ranjit-FinanceRanjit-Finance.pdf`