CMR INSTITUTE OF TECHNOLOGY	USN										

Sub:	Internal Assessment Test II July 2024 Optimization Techniques						Code:	BCS405C	
Date:	08/07/2024	Duration:	90 mins	Max Marks:	50	Sem:	IV	Branch:	**CSDS/CSML**

Answer any five of the following.		Marks	OBE	
			CO	RBT
1	Define a)convex set b)S.T a non-negative weighted sum of convex functions is convex.	2+8	CO3	L1,L2
2	Explain Stochastic Gradient Descent. S.T negative entropy for $f(x) = x\log_2 x$ is convex for x>0 for 2 points x = 2 & x = 4.	5+5	CO3,4	L2,L3
3	Minimise $f(x) = x(x-1.5)$ in [0,1] within the interval of uncertainty 0.25 using Fibonacci search method.	10	CO3	L3
4	$A = \begin{pmatrix} 1 & -1 & 0 \\ 0 & 1 & 1 \end{pmatrix}, X = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} B = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ Solve by optimization using Gradient descent.	10	CO3	L3
5	Define convex function. Explain optimization using Gradient Descent.	2+8	CO3	L2
6	Explain quadratic cost. Find the max of $f(x) = x(5\pi - x)$ in [0,20] with $\in = 0.1$using three point search method.	5+5	CO3	L2,L3

CMR INSTITUTE OF TECHNOLOGY	USN										

Sub:	Internal Assessment Test II July 2024 Optimization Techniques						Code:	BCS405C	
Date:	08/07/2024	Duration:	90 mins	Max Marks:	50	Sem:	IV	Branch:	**CSDS/CSML**

Answer any five of the following.		Marks	OBE	
			CO	RBT
1	Define a)convex set b)S.T a non-negative weighted sum of convex functions is convex.	2+8	CO3	L1,L2
2	Explain Stochastic Gradient Descent. S.T negative entropy for $f(x) = x\log_2 x$ is convex for x>0 for 2 points x = 2 & x = 4.	5+5	CO3,4	L2,L3
3	Minimise $f(x) = x(x-1.5)$ in [0,1] within the interval of uncertainty 0.25 using Fibonacci search method.	10	CO3	L3
4	$A = \begin{pmatrix} 1 & -1 & 0 \\ 0 & 1 & 1 \end{pmatrix}, X = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} B = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ Solve by optimization using Gradient descent.	10	CO3	L3
5	Define convex function. Explain optimization using Gradient Descent.	2+8	CO3	L2
6	Explain quadratic cost. Find the max of $f(x) = x(5\pi - x)$ in [0,20] with $\in = 0.1$using three point search method.	5+5	CO3	L2,L3

(1) A set C is a convex set if for any $x, y \in C$ and for any scalar θ with $0 \le \theta \le 1$ $\quad \theta x + (1-\theta) y \in C$

If f_1 and f_2 are convex functions then we have
$f_1(\theta x + (1-\theta) y) \le \theta f_1(x) + (1-\theta) f_1(y)$ — (1)
$f_2(\theta x + (1-\theta) y) \le \theta f_2(x) + (1-\theta) f_2(y)$ — (2)

(1) + (2) $f_1(\theta x + (1-\theta)y) + f_2(\theta x + (1-\theta) y)$
$\le \theta f_1(x) + (1-\theta) f_1(y) + \theta f_2(x) + (1-\theta) f_2(y)$
$\le \theta (f_1(x) + f_2(x)) + (1-\theta)(f_1(y) + f_2(y))$

This shows that the sum of convex functions is convex.

Q2 $\quad f(x) = x \log_2 x$
Let $x_1, x_2 \in C$
$f(\theta x_1 + (1-\theta) x_2) \le \theta f(x_1) + (1-\theta) f(x_2) \quad 0 \le \theta \le 1$
Let $x_1 = 2, \quad x_2 = 4 \quad \theta = 0.5$
$\theta x_1 + (1-\theta) x_2 = 2(0.5) + (1-0.5)4 = 3$
LHS $f(\theta x_1 + (1-\theta) x_2) = f(3) = 3 \log_2 3 \quad \sim 4.75$
$f(x_1) = f(2) = 2 \log_2 2 = 2$
$f(x_2) = f(4) = 4 \log_2 4 = 8$
RHS $\theta f(x_1) + (1-\theta) f(x_2) = 0.5(2) + (1-0.5)8 = 5$
$4.75 < 5 \Rightarrow f$ is convex

Q5 Let $f : \mathbb{R}^D \to \mathbb{R}$ be a function whose domain is a convex set. The function f is a convex function if $\forall x, y$ in the domain of f and for any scalar θ with $0 \le \theta \le 1$ we have
$f(\theta x + (1-\theta) y) \le \theta f(x) + (1-\theta) f(y)$

Q3 Min $f(x) = x(x-1.5)$ in $[0,1]$

k	F_{n-k}/F_{n-k+1}	L	R	x_1	x_2	$f(x_1)$	$f(x_2)$	L/R
1	3/5	0	1	0.4	0.6	-0.44	-0.54	R
2	2/3	0.4	1	0.6	0.8	-0.54	-0.56	R
3	1/2	0.6	1	0.8	0.8	-0.56	-0.56	L
4	1/1	0.6	0.8	0.8	0.6	-0.56	-0.54	

$x_{min} \in [0.6, 0.8]$

$$x^* = \frac{0.6+0.8}{2} = 0.7$$

$$f(x^*) = -0.56$$

Solve a system of linear eqns by Optimisation using Gradient Descent

<u>Ans</u> Given a matrix A of size $m \times n$, a variable vector $\overset{X}{}$ of size $n \times 1$, a right hand side vector $\overset{B}{}$ of size $m \times 1$, we have the system of m linear eqns in n unknowns

$$A = \begin{pmatrix} 1 & -1 & 0 \\ 0 & 1 & 1 \end{pmatrix} \quad X = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \quad B = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

To find X that min

$$f(x) = \|Ax - b\|^2$$

(enought to find an app soln to $AX - B = 0$
$f(x)$ error b/w exact + app soln)

$$= (Ax-b)^T (Ax-b)$$

$$(x_1 - x_2 - 1)^2 + (x_2 + x_3 - 1)^2$$

$$\nabla_x f(x) = 2(Ax-b)^T A$$

$$= 2 \begin{pmatrix} x_1 - x_2 - 1 \\ x_2 + x_3 - 1 \end{pmatrix}^T \begin{pmatrix} 1 & -1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$

$$= 2 \begin{pmatrix} x_1 - x_2 - 1 & x_2 + x_3 - 1 \end{pmatrix}_{1\times 2} \begin{pmatrix} 1 & -1 & 0 \\ 0 & 1 & 1 \end{pmatrix}_{2\times 3}$$

$$= 2 \begin{pmatrix} x_1 - x_2 - 1 & -x_1 + x_2 + 1 + x_2 + x_3 - 1 & x_2 + x_3 - 1 \end{pmatrix}_{1\times 3}$$

$$= \begin{pmatrix} 2x_1 - 2x_2 - 2 & -2x_1 + 4x_2 + 2x_3 & 2x_3 + 2x_2 - 2 \end{pmatrix}$$

A simple gradient descent algorithm is described by

(i) entropy measures the level of disorder or uncertainty in a given dataset or sys

$$x_{i+1} = x_i - \nu_i \left(\nabla f (x_i) \right)^T \quad i = 0, 1, 2, \cdots$$

Let $x_0 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$ $\quad \nu_i = 0.05 \quad i = 0, 1, 2, \cdots$

$$X_d = \left(x_1^{(0)} \; x_2^{(0)} \; x_3^{(0)} \right) = \left(0 \;\; 0 \;\; 0 \right)$$

$$f(x_0) = (-1)^2 + (-1)^2 = 2$$

$\underline{i=0}$ $$x_{i+1} = \begin{pmatrix} x_1^{i+1} \\ x_2^{i+1} \\ x_3^{i+1} \end{pmatrix} = \begin{pmatrix} x_1^i \\ x_2^i \\ x_3^i \end{pmatrix} - 0.05 \begin{pmatrix} 2x_1 - 2x_2 - 2 \\ -2x_1 + 4x_2 + 2x_3 \\ 2x_1 + 2x_3 - 2 \end{pmatrix}$$

$$x_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} - 0.05 \begin{pmatrix} -2 \\ 0 \\ -2 \end{pmatrix} = \begin{pmatrix} 0.1 \\ 0 \\ 0.1 \end{pmatrix}$$

$$f(x_1) = (0.1 - 1)^2 + (0.1 - 1)^2 = \underline{1.62}$$

$$x_2 = \begin{pmatrix} 0.1 \\ 0 \\ 0.1 \end{pmatrix} - 0.05 \begin{pmatrix} 2(0.1) - 2 \\ -0.2 + 0.2 \\ 0.2 - 2 \end{pmatrix} = \begin{pmatrix} 0.19 \\ 0 \\ 0.19 \end{pmatrix}$$

$$f(x_2) = \underline{1.3122}$$

$$x_3 = \begin{pmatrix} 0.19 \\ 0 \\ 0.19 \end{pmatrix} - 0.05 \begin{pmatrix} 2(0.19) - 2 \\ -2(0.19) + 4 \times 0 + 2(0.19) \\ 2 \times 0 + 2 \times 0.19 - 2 \end{pmatrix}$$

$$= \begin{pmatrix} 0.271 \\ 0 \\ 0.271 \end{pmatrix}$$

$$f(x_3) = (0.271 - 1)^2 + (0.271 - 1)^2$$
$$= \underline{1.063}$$

App soln	Error function
$(0 \ 0 \ 0)^T$	2
$(0.1 \ 0 \ 0.1)^T$	1.62
$(0.19 \ 0 \ 0.19)^T$	1.3122
$(0.271 \ 0 \ 0.271)^T$	1.063

Three point Search

This method is used for unconstraint optimisation. In this method we divide the interval into 4 equal parts. We select the central point as functional value which contained max/min. Then we select its interval around its central functional value. We expect this process until we reach the ϵ tolerance value that shows less value as

$$|f(x_c) - f(x_{cnm})| < \epsilon$$

The above step is called stopping criteria.

Q By using 3 point interval search to find max $f(x) = x(5\pi - x)$ in $[0, 20]$ with $\epsilon = 0.1$

Ans $f(x) = 5\pi x - x^2$ $\quad n = \frac{20-0}{4} = 5$

	x	$f(x)$
a	0	0
x_0	5	53.54
x_1	10	57.08
x_2	15	10.62
b	20	-85.84

Centre is x_1 SI $[5, 15]$

max value at $x_1 = 10$

$|57.08 - 53.54| = 3.51 \nless \epsilon$

ration 1

$f(x) = 5\pi x - x^2$ $\quad n = \frac{15-5}{4} = 2.5$

$[5, 15]$

x_0	5	53.54 ✓
x_3	7.5	61.56 .
x_1	10	57.08 ✓
x_4	12.5	40.1
x_2	15	10.62

Centre is at x_3 $\quad SI = [5, 10]$

max value at $x_3 = 7.5$

$|f(x_3) - f(x_1)| = |61.56 - 57.08| = 4.48 \not< \epsilon$

Iteration 2

$f(x) = 5\pi x - x^2$ $\quad n = \frac{10-5}{4} = 1.25$

x_0	5	53.54
x_5	6.25	59.8 ✓
x_3	7.5	61.56
x_6	8.75	60.88 ✓
x_1	10	57.08

Centre is at x_3 $\quad SI \; [6.25, 8.75]$

$|f(x_3) - f(x_6)| = |61.56 - 60.88| = 0.68 \not< \epsilon$

ation 3

$f(x) = 5\pi x - x^2, \quad [6.25, 8.75]$

$h = \dfrac{8.75 - 6.25}{4} = 0.625$

~~0.625~~

x_5	6.25	59.11
x_7	6.875	60.73
x_3	7.5	61.56
x_8	8.125	~~60.88~~ 61.61
x_6	8.75	~~[illegible]~~ 60.88

Centre is at x_8 . SI $[7.5, 8.75]$

$|f(x_8) - f(x_3)| =$ ~~$|61.61 - 60.|$~~ ~~$|61.56 - 60.88|$~~ $= 0.05 < G$

$|61.61 - 61.56|$

Ans max value is 61.61 at x_8 //

Understanding Quadratic Cost

The quadratic Cost function is $TC = a + bQ + Q^2$, where TC is the Total Cost and Q represents the Total Production.

$$\text{Average Cost} = \frac{= \text{Total Cost}}{Q}$$

$$Marginal\ Cost = First\ Derivative\ of\ (Total\ Cost)$$

Understanding Fixed Cost and Variable Cost

For example, if you are producing some item, rent is 10000 for the place. Then this can be considered as a fixed cost. If the production is higher, the variable cost is higher. If the production is more the variable cost is less.

$$Total\ cost = Total\ Fixed\ Cost + Total\ Variable\ Cost$$

The gradient of the Quadratic Cost

- The term "quadratic cost" generally refers to any cost function that is quadratic in nature. This means the cost function has the form of a quadratic polynomial.
- In the context of linear regression, the Quadratic Cost is given by

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x_i) - y_i)^2$$

where $h_\theta(x_i)$ is the hypothesis (or prediction) for the i-th data point, y_i is the actual value, m is the number of data points, and θ are the parameters of the model.

Mean Squared Error (MSE):

- The Mean Squared Error is a specific example of a quadratic cost function.
- It represents the average of the squared differences between the predicted values and the actual values.

The primary difference is that the quadratic cost function often includes a factor of (1/2) for mathematical convenience in optimization (specifically in gradient descent). This factor does not affect the optimization process but simplifies the derivative calculations.

Summary

While "quadratic cost" is a general term for a cost function that is quadratic in nature, "Mean Squared Error" is a specific form of such a cost function used primarily in regression problems. They are conceptually similar, but the quadratic cost function often includes an extra factor of (1/2) for ease of differentiation.

We can model equality constraints by replacing them with two inequality constraints. That is for each equality constraint $h_j(\boldsymbol{x}) = 0$ we equivalently replace it by two constraints $h_j(\boldsymbol{x}) \leqslant 0$ and $h_j(\boldsymbol{x}) \geqslant 0$. It turns out that the resulting Lagrange multipliers are then unconstrained.

Therefore, we constrain the Lagrange multipliers corresponding to the inequality constraints in (7.28) to be non-negative, and leave the Lagrange multipliers corresponding to the equality constraints unconstrained. ◇

7.3 Convex Optimization

We focus our attention of a particularly useful class of optimization problems, where we can guarantee global optimality. When $f(\cdot)$ is a convex function, and when the constraints involving $g(\cdot)$ and $h(\cdot)$ are convex sets, this is called a *convex optimization problem*. In this setting, we have *strong duality*: The optimal solution of the dual problem is the same as the optimal solution of the primal problem. The distinction between convex functions and convex sets are often not strictly presented in machine learning literature, but one can often infer the implied meaning from context.

convex optimization problem

strong duality

convex set

Definition 7.2. A set $\mathcal{C}$ is a *convex set* if for any $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}$ and for any scalar θ with $0 \leqslant \theta \leqslant 1$, we have

$$\theta \boldsymbol{x} + (1 - \theta)\boldsymbol{y} \in \mathcal{C}\,. \tag{7.29}$$

Figure 7.5 Example of a convex set.

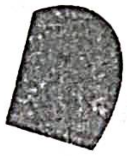
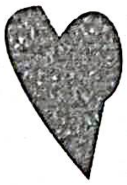


Figure 7.6 Example of a nonconvex set.



Convex sets are sets such that a straight line connecting any two elements of the set lie inside the set. Figures 7.5 and 7.6 illustrate convex and nonconvex sets, respectively.

Convex functions are functions such that a straight line between any two points of the function lie above the function. Figure 7.2 shows a nonconvex function, and Figure 7.3 shows a convex function. Another convex function is shown in Figure 7.7.

Definition 7.3. Let function $f : \mathbb{R}^D \to \mathbb{R}$ be a function whose domain is a convex set. The function f is a *convex function* if for all $\boldsymbol{x}, \boldsymbol{y}$ in the domain of f, and for any scalar θ with $0 \leqslant \theta \leqslant 1$, we have

$$f(\theta \boldsymbol{x} + (1 - \theta)\boldsymbol{y}) \leqslant \theta f(\boldsymbol{x}) + (1 - \theta) f(\boldsymbol{y})\,. \tag{7.30}$$

convex function

concave function

Remark. A *concave function* is the negative of a convex function. ◇

The constraints involving $g(\cdot)$ and $h(\cdot)$ in (7.28) truncate functions at a scalar value, resulting in sets. Another relation between convex functions and convex sets is to consider the set obtained by "filling in" a convex function. A convex function is a bowl-like object, and we imagine pouring water into it to fill it up. This resulting filled-in set, called the *epigraph* of the convex function, is a convex set.

epigraph

If a function $f : \mathbb{R}^n \to \mathbb{R}$ is differentiable, we can specify convexity in

We can model equality constraints by replacing them with two inequality constraints. That is for each equality constraint $h_j(\boldsymbol{x}) = 0$ we equivalently replace it by two constraints $h_j(\boldsymbol{x}) \leqslant 0$ and $h_j(\boldsymbol{x}) \geqslant 0$. It turns out that the resulting Lagrange multipliers are then unconstrained.

Therefore, we constrain the Lagrange multipliers corresponding to the inequality constraints in (7.28) to be non-negative, and leave the Lagrange multipliers corresponding to the equality constraints unconstrained. ◇

7.3 Convex Optimization

We focus our attention of a particularly useful class of optimization problems, where we can guarantee global optimality. When $f(\cdot)$ is a convex function, and when the constraints involving $g(\cdot)$ and $h(\cdot)$ are convex sets, this is called a *convex optimization problem*. In this setting, we have *strong duality*: The optimal solution of the dual problem is the same as the optimal solution of the primal problem. The distinction between convex functions and convex sets are often not strictly presented in machine learning literature, but one can often infer the implied meaning from context.

convex optimization problem

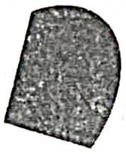
strong duality

convex set

Definition 7.2. A set $\mathcal{C}$ is a *convex set* if for any $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}$ and for any scalar θ with $0 \leqslant \theta \leqslant 1$, we have

$$\theta\boldsymbol{x} + (1-\theta)\boldsymbol{y} \in \mathcal{C}\,. \tag{7.29}$$

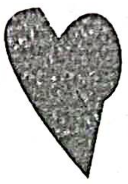
Figure 7.5 Example of a convex set.



Convex sets are sets such that a straight line connecting any two elements of the set lie inside the set. Figures 7.5 and 7.6 illustrate convex and nonconvex sets, respectively.

Convex functions are functions such that a straight line between any two points of the function lie above the function. Figure 7.2 shows a nonconvex function, and Figure 7.3 shows a convex function. Another convex function is shown in Figure 7.7.

Figure 7.6 Example of a nonconvex set.



Definition 7.3. Let function $f : \mathbb{R}^D \to \mathbb{R}$ be a function whose domain is a convex set. The function f is a *convex function* if for all $\boldsymbol{x}, \boldsymbol{y}$ in the domain of f, and for any scalar θ with $0 \leqslant \theta \leqslant 1$, we have

$$f(\theta\boldsymbol{x} + (1-\theta)\boldsymbol{y}) \leqslant \theta f(\boldsymbol{x}) + (1-\theta) f(\boldsymbol{y})\,. \tag{7.30}$$

convex function

concave function

Remark. A *concave function* is the negative of a convex function. ◇

The constraints involving $g(\cdot)$ and $h(\cdot)$ in (7.28) truncate functions at a scalar value, resulting in sets. Another relation between convex functions and convex sets is to consider the set obtained by "filling in" a convex function. A convex function is a bowl-like object, and we imagine pouring water into it to fill it up. This resulting filled-in set, called the *epigraph* of the convex function, is a convex set.

epigraph

If a function $f : \mathbb{R}^n \to \mathbb{R}$ is differentiable, we can specify convexity in

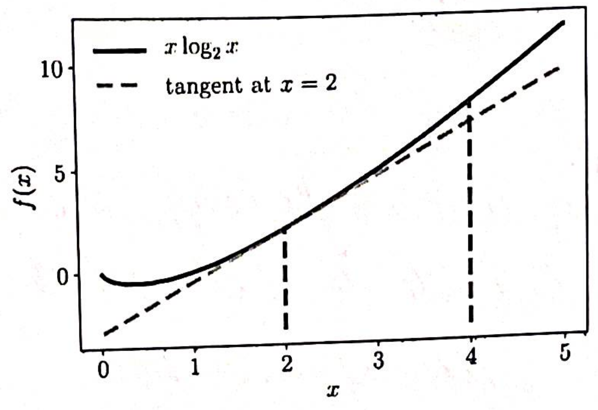


Figure 7.8 The negative entropy function (which is convex) and its tangent at $x = 2$.

We can check that a function or set is convex from first principles by recalling the definitions. In practice, we often rely on operations that preserve convexity to check that a particular function or set is convex. Although the details are vastly different, this is again the idea of closure that we introduced in Chapter 2 for vector spaces.

Example 7.4
A nonnegative weighted sum of convex functions is convex. Observe that if f is a convex function, and $\alpha \geqslant 0$ is a nonnegative scalar, then the function αf is convex. We can see this by multiplying α to both sides of the equation in Definition 7.3, and recalling that multiplying a nonnegative number does not change the inequality.

If f_1 and f_2 are convex functions, then we have by the definition

$$f_1(\theta x + (1-\theta)y) \leqslant \theta f_1(x) + (1-\theta)f_1(y) \tag{7.34}$$
$$f_2(\theta x + (1-\theta)y) \leqslant \theta f_2(x) + (1-\theta)f_2(y)\,. \tag{7.35}$$

Summing up both sides gives us

$$\begin{aligned} &f_1(\theta x + (1-\theta)y) + f_2(\theta x + (1-\theta)y) \\ &\leqslant \theta f_1(x) + (1-\theta)f_1(y) + \theta f_2(x) + (1-\theta)f_2(y)\,, \end{aligned} \tag{7.36}$$

where the right-hand side can be rearranged to

$$\theta(f_1(x) + f_2(x)) + (1-\theta)(f_1(y) + f_2(y))\,, \tag{7.37}$$

completing the proof that the sum of convex functions is convex.

Combining the preceding two facts, we see that $\alpha f_1(x) + \beta f_2(x)$ is convex for $\alpha, \beta \geqslant 0$. This closure property can be extended using a similar argument for nonnegative weighted sums of more than two convex functions.

Remark. The inequality in (7.30) is sometimes called *Jensen's inequality*. In fact, a whole class of inequalities for taking nonnegative weighted sums of convex functions are all called Jensen's inequality. ◇

Jensen's inequality

In summary, a constrained optimization problem is called a *convex optimization problem* if

convex optimization problem

$$\begin{aligned} &\min_{x} f(x) \\ &\text{subject to } g_i(x) \leqslant 0 \quad \text{for all} \quad i = 1, \dots, m \\ &\qquad h_j(x) = 0 \quad \text{for all} \quad j = 1, \dots, n, \end{aligned} \tag{7.38}$$

where all functions $f(x)$ and $g_i(x)$ are convex functions, and all $h_j(x) = 0$ are convex sets. In the following, we will describe two classes of convex optimization problems that are widely used and well understood.

7.3.1 Linear Programming

Consider the special case when all the preceding functions are linear, i.e.,

$$\begin{aligned} &\min_{x \in \mathbb{R}^d} \quad c^\top x \\ &\text{subject to} \quad Ax \leqslant b, \end{aligned} \tag{7.39}$$

where $A \in \mathbb{R}^{m \times d}$ and $b \in \mathbb{R}^m$. This is known as a *linear program*. It has d variables and m linear constraints. The Lagrangian is given by

linear program

Linear programs are one of the most widely used approaches in industry.

$$\mathfrak{L}(x, \lambda) = c^\top x + \lambda^\top (Ax - b), \tag{7.40}$$

where $\lambda \in \mathbb{R}^m$ is the vector of non-negative Lagrange multipliers. Rearranging the terms corresponding to x yields

$$\mathfrak{L}(x, \lambda) = (c + A^\top \lambda)^\top x - \lambda^\top b. \tag{7.41}$$

Taking the derivative of $\mathfrak{L}(x, \lambda)$ with respect to x and setting it to zero gives us

$$c + A^\top \lambda = 0. \tag{7.42}$$

Therefore, the dual Lagrangian is $\mathfrak{D}(\lambda) = -\lambda^\top b$. Recall we would like to maximize $\mathfrak{D}(\lambda)$. In addition to the constraint due to the derivative of $\mathfrak{L}(x, \lambda)$ being zero, we also have the fact that $\lambda \geqslant 0$, resulting in the following dual optimization problem

It is convention to minimize the primal and maximize the dual.

$$\begin{aligned} &\max_{\lambda \in \mathbb{R}^m} \quad -b^\top \lambda \\ &\text{subject to} \quad c + A^\top \lambda = 0 \\ &\qquad\qquad \lambda \geqslant 0. \end{aligned} \tag{7.43}$$

This is also a linear program, but with m variables. We have the choice of solving the primal (7.39) or the dual (7.43) program depending on

A
CR22
3

whether m or d is larger. Recall that d is the number of variables and m is the number of constraints in the primal linear program.

Example 7.5 (Linear Program)
Consider the linear program

$$\begin{aligned} \min_{\boldsymbol{x}\in\mathbb{R}^2} \quad & -\begin{bmatrix}5\\3\end{bmatrix}^\top \begin{bmatrix}x_1\\x_2\end{bmatrix} \\ \text{subject to} \quad & \begin{bmatrix}2 & 2\\ 2 & -4\\ -2 & 1\\ 0 & -1\\ 0 & 1\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix} \leqslant \begin{bmatrix}33\\8\\5\\-1\\8\end{bmatrix} \end{aligned} \tag{7.44}$$

with two variables. This program is also shown in Figure 7.9. The objective function is linear, resulting in linear contour lines. The constraint set in standard form is translated into the legend. The optimal value must lie in the shaded (feasible) region, and is indicated by the star.

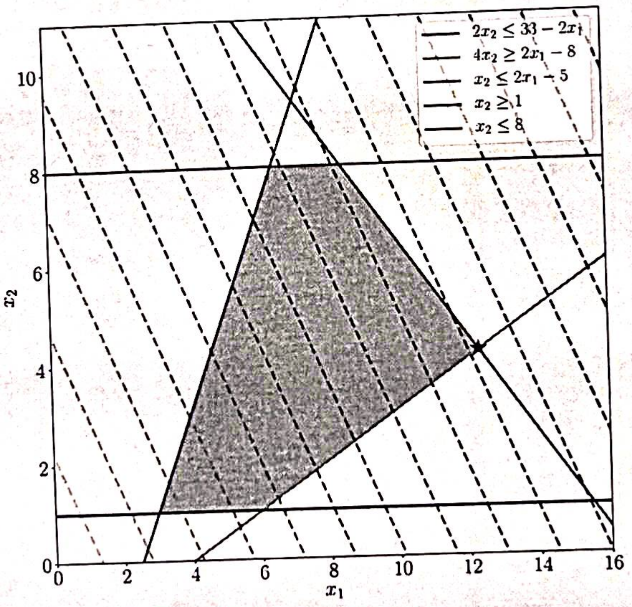


Figure 7.9 Illustration of a linear program. The unconstrained problem (indicated by the contour lines) has a minimum on the right side. The optimal value given the constraints are shown by the star.

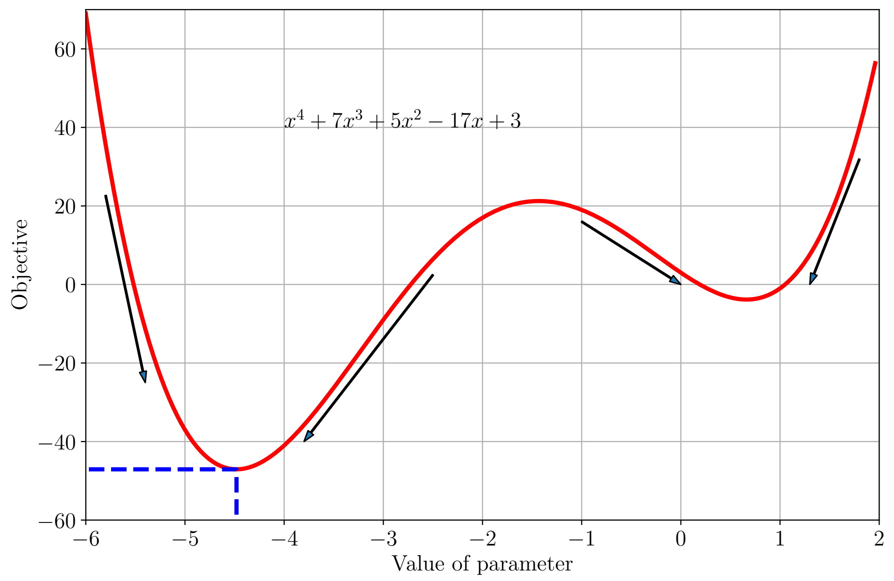


Figure 7.2 Example objective function. Negative gradients are indicated by arrows, and the global minimum is indicated by the dashed blue line.

right, but not how far (this is called the step-size). Furthermore, if we had started at the right side (e.g., $x_0 = 0$) the negative gradient would have led us to the wrong minimum. Figure 7.2 illustrates the fact that for $x > -1$, the negative gradient points toward the minimum on the right of the figure, which has a larger objective value.

According to the Abel–Ruffini theorem, there is in general no algebraic solution for polynomials of degree 5 or more (Abel, 1826).

In Section 7.3, we will learn about a class of functions, called convex functions, that do not exhibit this tricky dependency on the starting point of the optimization algorithm. For convex functions, all local minimums are global minimum. It turns out that many machine learning objective functions are designed such that they are convex, and we will see an example in Chapter 12.

For convex functions all local minima are global minimum.

The discussion in this chapter so far was about a one-dimensional function, where we are able to visualize the ideas of gradients, descent directions, and optimal values. In the rest of this chapter we develop the same ideas in high dimensions. Unfortunately, we can only visualize the concepts in one dimension, but some concepts do not generalize directly to higher dimensions, therefore some care needs to be taken when reading.

7.1 Optimization Using Gradient Descent

We now consider the problem of solving for the minimum of a real-valued function

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) , \tag{7.4}$$

where $f : \mathbb{R}^d \to \mathbb{R}$ is an objective function that captures the machine learning problem at hand. We assume that our function f is differentiable, and we are unable to analytically find a solution in closed form.

Gradient descent is a first-order optimization algorithm. To find a local minimum of a function using gradient descent, one takes steps proportional to the negative of the gradient of the function at the current point. Recall from Section 5.1 that the gradient points in the direction of the steepest ascent. Another useful intuition is to consider the set of lines where the function is at a certain value ($f(\boldsymbol{x}) = c$ for some value $c \in \mathbb{R}$), which are known as the contour lines. The gradient points in a direction that is orthogonal to the contour lines of the function we wish to optimize.

We use the convention of row vectors for gradients.

Let us consider multivariate functions. Imagine a surface (described by the function $f(\boldsymbol{x})$) with a ball starting at a particular location $\boldsymbol{x}_0$. When the ball is released, it will move downhill in the direction of steepest descent. Gradient descent exploits the fact that $f(\boldsymbol{x}_0)$ decreases fastest if one moves from $\boldsymbol{x}_0$ in the direction of the negative gradient $-((\nabla f)(\boldsymbol{x}_0))^\top$ of f at $\boldsymbol{x}_0$. We assume in this book that the functions are differentiable, and refer the reader to more general settings in Section 7.4. Then, if

$$\boldsymbol{x}_1 = \boldsymbol{x}_0 - \gamma((\nabla f)(\boldsymbol{x}_0))^\top \tag{7.5}$$

for a small *step-size* $\gamma \geqslant 0$, then $f(\boldsymbol{x}_1) \leqslant f(\boldsymbol{x}_0)$. Note that we use the transpose for the gradient since otherwise the dimensions will not work out.

This observation allows us to define a simple gradient descent algorithm: If we want to find a local optimum $f(\boldsymbol{x}_*)$ of a function $f : \mathbb{R}^n \to \mathbb{R}$, $\boldsymbol{x} \mapsto f(\boldsymbol{x})$, we start with an initial guess $\boldsymbol{x}_0$ of the parameters we wish to optimize and then iterate according to

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i - \gamma_i((\nabla f)(\boldsymbol{x}_i))^\top . \tag{7.6}$$

For suitable step-size γ_i, the sequence $f(\boldsymbol{x}_0) \geqslant f(\boldsymbol{x}_1) \geqslant \dots$ converges to a local minimum.

Example 7.1

Consider a quadratic function in two dimensions

$$f\left(\begin{bmatrix}x_1\\x_2\end{bmatrix}\right) = \frac{1}{2}\begin{bmatrix}x_1\\x_2\end{bmatrix}^\top \begin{bmatrix}2 & 1\\1 & 20\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix} - \begin{bmatrix}5\\3\end{bmatrix}^\top \begin{bmatrix}x_1\\x_2\end{bmatrix} \tag{7.7}$$

with gradient

$$\nabla f\left(\begin{bmatrix}x_1\\x_2\end{bmatrix}\right) = \begin{bmatrix}x_1\\x_2\end{bmatrix}^\top \begin{bmatrix}2 & 1\\1 & 20\end{bmatrix} - \begin{bmatrix}5\\3\end{bmatrix}^\top . \tag{7.8}$$

Starting at the initial location $\boldsymbol{x}_0 = [-3, -1]^\top$, we iteratively apply (7.6) to obtain a sequence of estimates that converge to the minimum value

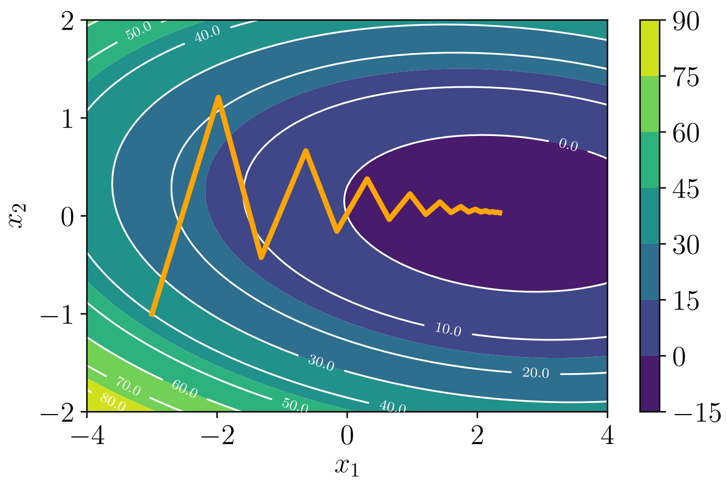


Figure 7.3 Gradient descent on a two-dimensional quadratic surface (shown as a heatmap). See Example 7.1 for a description.

(illustrated in Figure 7.3). We can see (both from the figure and by plugging $\boldsymbol{x}_0$ into (7.8) with $\gamma = 0.085$) that the negative gradient at $\boldsymbol{x}_0$ points north and east, leading to $\boldsymbol{x}_1 = [-1.98, 1.21]^\top$. Repeating that argument gives us $\boldsymbol{x}_2 = [-1.32, -0.42]^\top$, and so on.

Remark. Gradient descent can be relatively slow close to the minimum: Its asymptotic rate of convergence is inferior to many other methods. Using the ball rolling down the hill analogy, when the surface is a long, thin valley, the problem is poorly conditioned (Trefethen and Bau III, 1997). For poorly conditioned convex problems, gradient descent increasingly "zigzags" as the gradients point nearly orthogonally to the shortest direction to a minimum point; see Figure 7.3. $\diamondsuit$

7.1.1 Step-size

As mentioned earlier, choosing a good step-size is important in gradient descent. If the step-size is too small, gradient descent can be slow. If the step-size is chosen too large, gradient descent can overshoot, fail to converge, or even diverge. We will discuss the use of momentum in the next section. It is a method that smoothes out erratic behavior of gradient updates and dampens oscillations.

The step-size is also called the learning rate.

Adaptive gradient methods rescale the step-size at each iteration, depending on local properties of the function. There are two simple heuristics (Toussaint, 2012):

- When the function value increases after a gradient step, the step-size was too large. Undo the step and decrease the step-size.
- When the function value decreases the step could have been larger. Try to increase the step-size.

a moving average. The momentum-based method remembers the update $\Delta \boldsymbol{x}_i$ at each iteration i and determines the next update as a linear combination of the current and previous gradients

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i - \gamma_i((\nabla f)(\boldsymbol{x}_i))^\top + \alpha \Delta \boldsymbol{x}_i \tag{7.11}$$

$$\Delta \boldsymbol{x}_i = \boldsymbol{x}_i - \boldsymbol{x}_{i-1} = \alpha \Delta \boldsymbol{x}_{i-1} - \gamma_{i-1}((\nabla f)(\boldsymbol{x}_{i-1}))^\top , \tag{7.12}$$

where $\alpha \in [0, 1]$. Sometimes we will only know the gradient approximately. In such cases, the momentum term is useful since it averages out different noisy estimates of the gradient. One particularly useful way to obtain an approximate gradient is by using a stochastic approximation, which we discuss next.

7.1.3 Stochastic Gradient Descent

Computing the gradient can be very time consuming. However, often it is possible to find a "cheap" approximation of the gradient. Approximating the gradient is still useful as long as it points in roughly the same direction as the true gradient.

stochastic gradient descent

Stochastic gradient descent (often shortened as SGD) is a stochastic approximation of the gradient descent method for minimizing an objective function that is written as a sum of differentiable functions. The word stochastic here refers to the fact that we acknowledge that we do not know the gradient precisely, but instead only know a noisy approximation to it. By constraining the probability distribution of the approximate gradients, we can still theoretically guarantee that SGD will converge.

In machine learning, given $n = 1, \ldots, N$ data points, we often consider objective functions that are the sum of the losses L_n incurred by each example n. In mathematical notation, we have the form

$$L(\boldsymbol{\theta}) = \sum_{n=1}^{N} L_n(\boldsymbol{\theta}) , \tag{7.13}$$

where $\boldsymbol{\theta}$ is the vector of parameters of interest, i.e., we want to find $\boldsymbol{\theta}$ that minimizes L. An example from regression (Chapter 9) is the negative log-likelihood, which is expressed as a sum over log-likelihoods of individual examples so that

$$L(\boldsymbol{\theta}) = -\sum_{n=1}^{N} \log p(y_n | \boldsymbol{x}_n, \boldsymbol{\theta}) , \tag{7.14}$$

where $\boldsymbol{x}_n \in \mathbb{R}^D$ are the training inputs, y_n are the training targets, and $\boldsymbol{\theta}$ are the parameters of the regression model.

Standard gradient descent, as introduced previously, is a "batch" optimization method, i.e., optimization is performed using the full training set by updating the vector of parameters according to

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i - \gamma_i (\nabla L(\boldsymbol{\theta}_i))^\top = \boldsymbol{\theta}_i - \gamma_i \sum_{n=1}^{N} (\nabla L_n(\boldsymbol{\theta}_i))^\top \tag{7.15}$$

for a suitable step-size parameter γ_i. Evaluating the sum gradient may require expensive evaluations of the gradients from all individual functions L_n. When the training set is enormous and/or no simple formulas exist, evaluating the sums of gradients becomes very expensive.

Consider the term $\sum_{n=1}^{N}(\nabla L_n(\boldsymbol{\theta}_i))$ in (7.15). We can reduce the amount of computation by taking a sum over a smaller set of L_n. In contrast to batch gradient descent, which uses all L_n for $n = 1, \ldots, N$, we randomly choose a subset of L_n for mini-batch gradient descent. In the extreme case, we randomly select only a single L_n to estimate the gradient. The key insight about why taking a subset of data is sensible is to realize that for gradient descent to converge, we only require that the gradient is an unbiased estimate of the true gradient. In fact the term $\sum_{n=1}^{N}(\nabla L_n(\boldsymbol{\theta}_i))$ in (7.15) is an empirical estimate of the expected value (Section 6.4.1) of the gradient. Therefore, any other unbiased empirical estimate of the expected value, for example using any subsample of the data, would suffice for convergence of gradient descent.

Remark. When the learning rate decreases at an appropriate rate, and subject to relatively mild assumptions, stochastic gradient descent converges almost surely to local minimum (Bottou, 1998). $\diamondsuit$

Why should one consider using an approximate gradient? A major reason is practical implementation constraints, such as the size of central processing unit (CPU)/graphics processing unit (GPU) memory or limits on computational time. We can think of the size of the subset used to estimate the gradient in the same way that we thought of the size of a sample when estimating empirical means (Section 6.4.1). Large mini-batch sizes will provide accurate estimates of the gradient, reducing the variance in the parameter update. Furthermore, large mini-batches take advantage of highly optimized matrix operations in vectorized implementations of the cost and gradient. The reduction in variance leads to more stable convergence, but each gradient calculation will be more expensive.

In contrast, small mini-batches are quick to estimate. If we keep the mini-batch size small, the noise in our gradient estimate will allow us to get out of some bad local optima, which we may otherwise get stuck in. In machine learning, optimization methods are used for training by minimizing an objective function on the training data, but the overall goal is to improve generalization performance (Chapter 8). Since the goal in machine learning does not necessarily need a precise estimate of the minimum of the objective function, approximate gradients using mini-batch approaches have been widely used. Stochastic gradient descent is very effective in large-scale machine learning problems (Bottou et al., 2018),

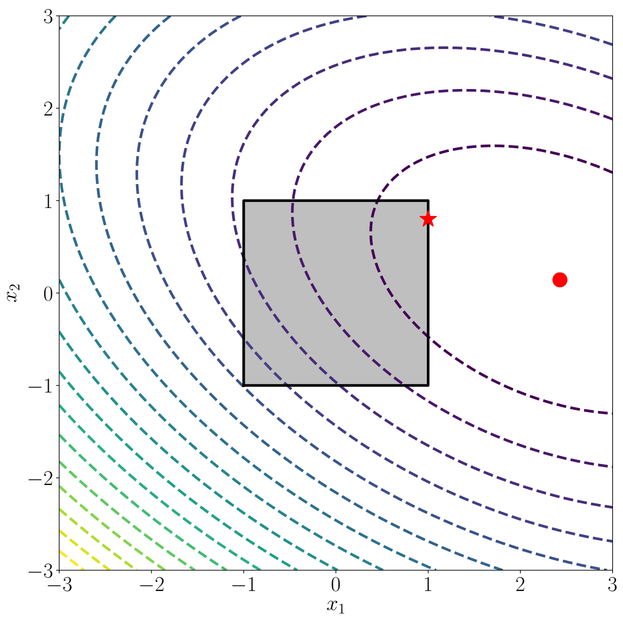


Figure 7.4 Illustration of constrained optimization. The unconstrained problem (indicated by the contour lines) has a minimum on the right side (indicated by the circle). The box constraints ($-1 \leqslant x \leqslant 1$ and $-1 \leqslant y \leqslant 1$) require that the optimal solution is within the box, resulting in an optimal value indicated by the star.

such as training deep neural networks on millions of images (Dean et al., 2012), topic models (Hoffman et al., 2013), reinforcement learning (Mnih et al., 2015), or training of large-scale Gaussian process models (Hensman et al., 2013; Gal et al., 2014).

7.2 Constrained Optimization and Lagrange Multipliers

In the previous section, we considered the problem of solving for the minimum of a function

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) , \tag{7.16}$$

where $f : \mathbb{R}^D \to \mathbb{R}$.

In this section, we have additional constraints. That is, for real-valued functions $g_i : \mathbb{R}^D \to \mathbb{R}$ for $i = 1, \ldots, m$, we consider the constrained optimization problem (see Figure 7.4 for an illustration)

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & f(\boldsymbol{x}) \\ \text{subject to} \quad & g_i(\boldsymbol{x}) \leqslant 0 \quad \text{for all} \quad i = 1, \ldots, m . \end{aligned} \tag{7.17}$$

It is worth pointing out that the functions f and g_i could be non-convex in general, and we will consider the convex case in the next section.

One obvious, but not very practical, way of converting the constrained problem (7.17) into an unconstrained one is to use an indicator function

$$J(\boldsymbol{x}) = f(\boldsymbol{x}) + \sum_{i=1}^{m} \mathbf{1}(g_i(\boldsymbol{x})) , \tag{7.18}$$