CMR INSTITUTE OF TECHNOLOGY	USN	
	Internal Assessment Test I June 2024	CMRIT

Sub:	**Optimization Techniques**						Code:	BCS405C	
Date:	03/06/2024	Duration:	90 mins	Max Marks:	50	Sem:	IV	Branch:	**CSDS/CSML**

Answer any five of the following.

		Marks	OBE CO	OBE RBT
1	Explain Gradient of a Least Squares Loss in a linear model.	10	CO1	L2
2	Explain Gradient of Vectors with respect to Matrices.	10	CO1	L2
3	Find the Taylor's series expansion of $f(x) = \exp(xy)$ plane up to 3rd degree term about the point (1,1).	10	CO1	L3
4	Explain Gradients in a deep network.	10	CO2	L2
5	a)Consider the function $h = fog,\ f(x,y) = \exp(xy^2), x = tcost, y = tsint$ Find the gradient b) Find the gradient of $f(xy) = xy^2 + x^3y$.	6+4	CO1	L3
6	a)Define multivariate Taylor's series. b)Find the derivative of $f(x) = (2x+1)^4$ using chain rule. c) Find the partial derivative of $f(x) = (y + 2x^3)$.	3+4+3	CO1.2	L3

CMR INSTITUTE OF TECHNOLOGY	USN	
	Internal Assessment Test I June 2024	CMRIT

Sub:	**Optimization Techniques**						Code:	BCS405C	
Date:	03/062024	Duration:	90 mins	Max Marks:	50	Sem:	IV	Branch:	**CSDS/CSML**

Answer any five of the following.

		Marks	OBE CO	OBE RBT
1	Explain Gradient of a Least Squares Loss in a linear model.	10	CO1	L2
2	Explain Gradient of Vectors with respect to Matrices.	10	CO1	L2
3	Find the Taylor's series expansion of $f(x) = \exp(xy)$ plane up to 3rd degree term about the point (1,1).	10	CO1	L3
4	Explain Gradients in a deep network.	10	CO2	L2
5	a)Consider the function $h = fog,\ f(x,y) = \exp(xy^2), x = tcost, y = tsint$ Find the gradient b) Find the gradient of $f(xy) = xy^2 + x^3y$.	6+4	CO1	L3
6	a)Define multivariate Taylor's series. b)Find the derivative of $f(x) = (2x+1)^4$ using chain rule. c) Find the partial derivative of $f(x) = (y + 2x^3)$.	3+4+3	CO1.2	L3

Q3 TS

$$f(x,y) = f(a,b) + \frac{1}{1!}\left[(x-a)f_x + (y-b)f_y\right]$$
$$+ \frac{1}{2!}\left[(x-a)^2 f_{xx} + 2(x-a)(y-b)f_{xy} + (y-b)^2 f_{yy}\right]$$
$$+ \frac{1}{3!}\left[(x-a)^3 f_{xxx} + 3(x-a)^2(y-b)f_{xxy} + 3(x-a)(y-b)^2 f_{xyy}\right.$$
$$\left. + (y-b)^3 f_{yyy}\right] + \ldots \quad (1)$$

$f(x,y) = e^{xy}$ $\quad f(1,1) = e$

$f_x = ye^{xy}$ $\quad f_x(1,1) = e$ $\quad f_y = xe^{xy}$ $\quad f_y(1,1) = e$

$f_{xx} = y^2e^{xy}$ $\quad f_{xx}(1,1) = e$ $\quad f_{yy} = x^2e^{xy}$ $\quad f_{yy}(1,1) = e$

$f_{xy} = \frac{\partial}{\partial y}(f_x) = \frac{\partial}{\partial y}(ye^{xy}) = e^{xy} + y.xe^{xy}$

$f_{xy}(1,1) = 2e$ $\quad f_{xxx} = y^3e^{xy}$ $\quad f_{xxx}(1,1) = e$

$f_{yyy} = x^3e^{xy}$ $\quad f_{yyy}(1,1) = e$

$f_{xxy} = \frac{\partial}{\partial y}(f_{xx}) = \frac{\partial}{\partial y}(y^2e^{xy})$

$= 2ye^{xy} + y^2.xe^{xy}$

$f_{xxy}(1,1) = 2e + e = 3e$

$f_{xyy} = \frac{\partial}{\partial x}(f_{yy}) = \frac{\partial}{\partial x}(x^2e^{xy})$

$= 2xe^{xy} + x^2.ye^{xy}$

$f_{xyy}(1,1) = 3e$

① is

$$f(x,y) = f(1,1) + \frac{1}{1!}\left[(x-1)e + (y-1)e\right]$$

$$+ \frac{1}{2!}\left[(x-1)^2 e + 2(x-1)(y-1)(2e) + (y-1)^2 e\right]$$

$$+ \frac{1}{3!}\left[(x-1)^3 e + 3(x-1)^2(y-1)(3e) + 3(x-1)(y-1)^2 3e + (y-1)^3 e\right] + \cdots$$

$f(x,y) = \exp(xy^2)$ $\quad x = t\cos t \quad y = t\sin t$

Q5

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

$$= \exp(xy^2)\cdot y^2 (1\cdot\cos t - t\sin t)$$
$$+ \exp(xy^2)(2xy)(1\cdot\sin t + t\cos t)$$

$$= \exp(t\cos t\cdot t^2\sin^2 t)\cdot(t^2\sin^2 t)(\cos t - t\sin t)$$

$$+ \exp(t\cos t\cdot t^2\sin^2 t)(2t\cos t\, t\sin t)(\sin t + t\cos t)$$

$$= \exp(t^3\sin^2 t\cos t)\left\{t^2\sin^2 t(\cos t - t\sin t) + 2t^2\sin t\cos t(\sin t + t\cos t)\right\}$$

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

$$\frac{\partial f}{\partial x} = 1\cdot y^2 + 3x^2 y \qquad \frac{\partial f}{\partial y} = 2xy + x^3$$

$$\frac{df}{dx} = \left[y^2 + 3x^2 y \quad 2xy + x^3\right] \in \mathbb{R}^{1\times 2}$$

6) a) Consider a function $f : \mathbb{R}^D \to \mathbb{R}$
$x \mapsto f(x), \; x \in \mathbb{R}^D$.
that is smooth at x_0. When we define the difference vector $\delta = x - x_0$, the multivariate Taylor series of f at x_0 is defined as

$$f(x) = \sum_{k=0}^{\infty} \frac{D_x^k f(x_0)}{k!} \delta^k$$

where $D_x^k f(x_0)$ is the k^{th} derivative of f w.r.to x, evaluated at x_0.

b) $h(x) = (2x+1)^4 = g(f(x))$ where $f(x) = 2x+1$

$g(f) = f^4 \quad f'(x) = 2 \quad g'(f) = 4f^3$

$h'(x) = g'(f) \, f'(x) = 4f^3 \cdot 2$
$= 8(2x+1)^3$

We will discuss this model in much more detail in Chapter 9 in the context of linear regression, where we need derivatives of the least-squares loss L with respect to the parameters θ.

Example 5.11 (Gradient of a Least-Squares Loss in a Linear Model)

Let us consider the linear model

$$y = \Phi\theta\,, \tag{5.75}$$

where $\theta \in \mathbb{R}^D$ is a parameter vector, $\Phi \in \mathbb{R}^{N\times D}$ are input features and $y \in \mathbb{R}^N$ are the corresponding observations. We define the functions

$$L(e) := \|e\|^2\,, \tag{5.76}$$

$$e(\theta) := y - \Phi\theta\,. \tag{5.77}$$

least-squares loss

We seek $\frac{\partial L}{\partial \theta}$, and we will use the chain rule for this purpose. L is called a *least-squares loss* function.

Before we start our calculation, we determine the dimensionality of the gradient as

$$\frac{\partial L}{\partial \theta} \in \mathbb{R}^{1\times D}\,. \tag{5.78}$$

The chain rule allows us to compute the gradient as

$$\frac{\partial L}{\partial \theta} = \frac{\partial L}{\partial e}\frac{\partial e}{\partial \theta}\,, \tag{5.79}$$

```
dLdtheta =
np.einsum(
'n,nd',
dLde,dedtheta)
```

where the dth element is given by

$$\frac{\partial L}{\partial \theta}[1, d] = \sum_{n=1}^{N} \frac{\partial L}{\partial e}[n]\frac{\partial e}{\partial \theta}[n, d]\,. \tag{5.80}$$

We know that $\|e\|^2 = e^\top e$ (see Section 3.2) and determine

$$\frac{\partial L}{\partial e} = 2e^\top \in \mathbb{R}^{1\times N}\,. \tag{5.81}$$

Furthermore, we obtain

$$\frac{\partial e}{\partial \theta} = -\Phi \in \mathbb{R}^{N\times D}\,, \tag{5.82}$$

such that our desired derivative is

$$\frac{\partial L}{\partial \theta} = -2e^\top\Phi \overset{(5.77)}{=} -\underbrace{2(y^\top - \theta^\top\Phi^\top)}_{1\times N}\underbrace{\Phi}_{N\times D} \in \mathbb{R}^{1\times D}\,. \tag{5.83}$$

Remark. We would have obtained the same result without using the chain rule by immediately looking at the function

$$L_2(\theta) := \|y - \Phi\theta\|^2 = (y - \Phi\theta)^\top(y - \Phi\theta)\,. \tag{5.84}$$

This approach is still practical for simple functions like L_2 but becomes impractical for deep function compositions. ◇

Example 5.12 (Gradient of Vectors with Respect to Matrices)
Let us consider the following example, where

$$\boldsymbol{f} = \boldsymbol{A}\boldsymbol{x}\,, \quad \boldsymbol{f} \in \mathbb{R}^{M}, \quad \boldsymbol{A} \in \mathbb{R}^{M\times N}, \quad \boldsymbol{x} \in \mathbb{R}^{N} \tag{5.85}$$

and where we seek the gradient $\mathrm{d}\boldsymbol{f}/\mathrm{d}\boldsymbol{A}$. Let us start again by determining the dimension of the gradient as

$$\frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}\boldsymbol{A}} \in \mathbb{R}^{M\times(M\times N)}\,. \tag{5.86}$$

By definition, the gradient is the collection of the partial derivatives:

$$\frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}\boldsymbol{A}} = \begin{bmatrix} \frac{\partial f_1}{\partial \boldsymbol{A}} \\ \vdots \\ \frac{\partial f_M}{\partial \boldsymbol{A}} \end{bmatrix}, \quad \frac{\partial f_i}{\partial \boldsymbol{A}} \in \mathbb{R}^{1\times(M\times N)}\,. \tag{5.87}$$

To compute the partial derivatives, it will be helpful to explicitly write out the matrix vector multiplication:

$$f_i = \sum_{j=1}^{N} A_{ij}x_j, \quad i = 1, \dots, M\,, \tag{5.88}$$

and the partial derivatives are then given as

$$\frac{\partial f_i}{\partial A_{iq}} = x_q\,. \tag{5.89}$$

This allows us to compute the partial derivatives of f_i with respect to a row of $\boldsymbol{A}$, which is given as

$$\frac{\partial f_i}{\partial A_{i,:}} = \boldsymbol{x}^\top \in \mathbb{R}^{1\times 1\times N}\,, \tag{5.90}$$

$$\frac{\partial f_i}{\partial A_{k\neq i,:}} = \mathbf{0}^\top \in \mathbb{R}^{1\times 1\times N} \tag{5.91}$$

where we have to pay attention to the correct dimensionality. Since f_i maps onto $\mathbb{R}$ and each row of A is of size $1 \times N$, we obtain a $1 \times 1 \times N$-sized tensor as the partial derivative of f_i with respect to a row of A.

We stack the partial derivatives (5.91) and get the desired gradient in (5.87) via

$$\frac{\partial f_i}{\partial A} = \begin{bmatrix} \mathbf{0}^\top \\ \vdots \\ \mathbf{0}^\top \\ \boldsymbol{x}^\top \\ \mathbf{0}^\top \\ \vdots \\ \mathbf{0}^\top \end{bmatrix} \in \mathbb{R}^{1\times(M\times N)} . \tag{5.92}$$

Example 5.13 (Gradient of Matrices with Respect to Matrices)
Consider a matrix $\boldsymbol{R} \in \mathbb{R}^{M\times N}$ and $\boldsymbol{f} : \mathbb{R}^{M\times N} \to \mathbb{R}^{N\times N}$ with

$$\boldsymbol{f}(\boldsymbol{R}) = \boldsymbol{R}^\top \boldsymbol{R} =: \boldsymbol{K} \in \mathbb{R}^{N\times N} , \tag{5.93}$$

where we seek the gradient $\mathrm{d}\boldsymbol{K}/\mathrm{d}\boldsymbol{R}$.

To solve this hard problem, let us first write down what we already know: The gradient has the dimensions

$$\frac{\mathrm{d}\boldsymbol{K}}{\mathrm{d}\boldsymbol{R}} \in \mathbb{R}^{(N\times N)\times(M\times N)} , \tag{5.94}$$

which is a tensor. Moreover,

$$\frac{\mathrm{d}K_{pq}}{\mathrm{d}\boldsymbol{R}} \in \mathbb{R}^{1\times M\times N} \tag{5.95}$$

for $p, q = 1, \dots, N$, where K_{pq} is the (p, q)th entry of $\boldsymbol{K} = \boldsymbol{f}(\boldsymbol{R})$. Denoting the ith column of $\boldsymbol{R}$ by $\boldsymbol{r}_i$, every entry of $\boldsymbol{K}$ is given by the dot product of two columns of $\boldsymbol{R}$, i.e.,

$$K_{pq} = \boldsymbol{r}_p^\top \boldsymbol{r}_q = \sum_{m=1}^{M} R_{mp} R_{mq} . \tag{5.96}$$

When we now compute the partial derivative $\frac{\partial K_{pq}}{\partial R_{ij}}$ we obtain

$$\frac{\partial K_{pq}}{\partial R_{ij}} = \sum_{m=1}^{M} \frac{\partial}{\partial R_{ij}} R_{mp} R_{mq} = \partial_{pqij} , \tag{5.97}$$

5.6.1 *Gradients in a Deep Network*

An area where the chain rule is used to an extreme is deep learning, where the function value $\boldsymbol{y}$ is computed as a many-level function composition

$$\boldsymbol{y} = (f_K \circ f_{K-1} \circ \cdots \circ f_1)(\boldsymbol{x}) = f_K(f_{K-1}(\cdots(f_1(\boldsymbol{x}))\cdots)), \tag{5.111}$$

where $\boldsymbol{x}$ are the inputs (e.g., images), $\boldsymbol{y}$ are the observations (e.g., class labels), and every function f_i, $i = 1, \ldots, K$, possesses its own parameters.

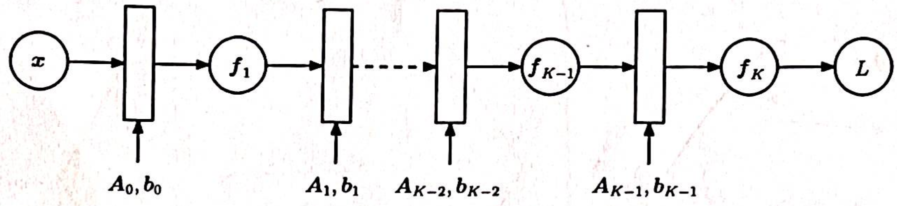

Figure 5.8 Forward pass in a multi-layer neural network to compute the loss L as a function of the inputs $\boldsymbol{x}$ and the parameters $\boldsymbol{A}_i$, $\boldsymbol{b}_i$.

We discuss the case, where the activation functions are identical in each layer to unclutter notation.

In neural networks with multiple layers, we have functions $f_i(\boldsymbol{x}_{i-1}) = \sigma(\boldsymbol{A}_{i-1}\boldsymbol{x}_{i-1} + \boldsymbol{b}_{i-1})$ in the ith layer. Here $\boldsymbol{x}_{i-1}$ is the output of layer $i-1$ and σ an activation function, such as the logistic sigmoid $\frac{1}{1+e^{-x}}$, tanh or a rectified linear unit (ReLU). In order to train these models, we require the gradient of a loss function L with respect to all model parameters $\boldsymbol{A}_j, \boldsymbol{b}_j$ for $j = 1, \dots, K$. This also requires us to compute the gradient of L with respect to the inputs of each layer. For example, if we have inputs $\boldsymbol{x}$ and observations $\boldsymbol{y}$ and a network structure defined by

$$\boldsymbol{f}_0 := \boldsymbol{x} \tag{5.112}$$

$$\boldsymbol{f}_i := \sigma_i(\boldsymbol{A}_{i-1}\boldsymbol{f}_{i-1} + \boldsymbol{b}_{i-1}), \quad i = 1, \dots, K, \tag{5.113}$$

see also Figure 5.8 for a visualization, we may be interested in finding $\boldsymbol{A}_j, \boldsymbol{b}_j$ for $j = 0, \dots, K-1$, such that the squared loss

$$L(\boldsymbol{\theta}) = \|\boldsymbol{y} - \boldsymbol{f}_K(\boldsymbol{\theta}, \boldsymbol{x})\|^2 \tag{5.114}$$

is minimized, where $\boldsymbol{\theta} = \{\boldsymbol{A}_0, \boldsymbol{b}_0, \dots, \boldsymbol{A}_{K-1}, \boldsymbol{b}_{K-1}\}$.

To obtain the gradients with respect to the parameter set $\boldsymbol{\theta}$, we require the partial derivatives of L with respect to the parameters $\boldsymbol{\theta}_j = \{\boldsymbol{A}_j, \boldsymbol{b}_j\}$ of each layer $j = 0, \dots, K-1$. The chain rule allows us to determine the partial derivatives as

A more in-depth discussion about gradients of neural networks can be found in Justin Domke's lecture notes `https://tinyurl.com/yalcxgtv`.

$$\frac{\partial L}{\partial \boldsymbol{\theta}_{K-1}} = \frac{\partial L}{\partial \boldsymbol{f}_K} \frac{\partial \boldsymbol{f}_K}{\partial \boldsymbol{\theta}_{K-1}} \tag{5.115}$$

$$\frac{\partial L}{\partial \boldsymbol{\theta}_{K-2}} = \frac{\partial L}{\partial \boldsymbol{f}_K} \boxed{\frac{\partial \boldsymbol{f}_K}{\partial \boldsymbol{f}_{K-1}} \frac{\partial \boldsymbol{f}_{K-1}}{\partial \boldsymbol{\theta}_{K-2}}} \tag{5.116}$$

$$\frac{\partial L}{\partial \boldsymbol{\theta}_{K-3}} = \frac{\partial L}{\partial \boldsymbol{f}_K} \frac{\partial \boldsymbol{f}_K}{\partial \boldsymbol{f}_{K-1}} \boxed{\frac{\partial \boldsymbol{f}_{K-1}}{\partial \boldsymbol{f}_{K-2}} \frac{\partial \boldsymbol{f}_{K-2}}{\partial \boldsymbol{\theta}_{K-3}}} \tag{5.117}$$

$$\frac{\partial L}{\partial \boldsymbol{\theta}_i} = \frac{\partial L}{\partial \boldsymbol{f}_K} \frac{\partial \boldsymbol{f}_K}{\partial \boldsymbol{f}_{K-1}} \cdots \boxed{\frac{\partial \boldsymbol{f}_{i+2}}{\partial \boldsymbol{f}_{i+1}} \frac{\partial \boldsymbol{f}_{i+1}}{\partial \boldsymbol{\theta}_i}} \tag{5.118}$$

The orange terms are partial derivatives of the output of a layer with respect to its inputs, whereas the **blue** terms are partial derivatives of the output of a layer with respect to its parameters. Assuming, we have already computed the partial derivatives $\partial L / \partial \boldsymbol{\theta}_{i+1}$, then most of the computation can be reused to compute $\partial L / \partial \boldsymbol{\theta}_i$. The additional terms that we

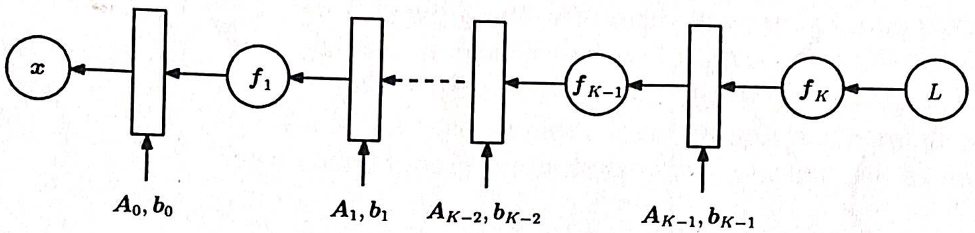

Figure 5.9 Backward pass in a multi-layer neural network to compute the gradients of the loss function.

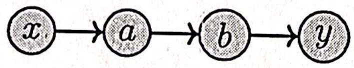

Figure 5.10 Simple graph illustrating the flow of data from x to y via some intermediate variables a, b.

need to compute are indicated by the boxes. Figure 5.9 visualizes that the gradients are passed backward through the network.