Indian Institute of Technology Bombay
CS 419 Quiz 1

Course Instructor: Prof. Abir De

Date: September 11, 2024 **Max Marks: 30** **Duration: 60 minutes**

Instructions:

- Answer all questions.
- Show all work clearly, be as clear and precise as possible.

Useful Identities

1. Gradient of the dot product

$$\nabla_{\mathbf{u}}(\mathbf{a}^T\mathbf{u}) = \mathbf{a}$$

2. Gradient of a Quadratic Form

$$\nabla_{\mathbf{u}}(\mathbf{u}^T\mathbf{A}\mathbf{u}) = (\mathbf{A} + \mathbf{A}^T)\mathbf{u}$$

where:

- $\mathbf{a}$ is a vector of size $d \times 1$.
- $\mathbf{u}$ is a vector of size $d \times 1$.
- $\mathbf{A}$ is a matrix of size $d \times d$.

Question 1 : LINEAR REGRESSION

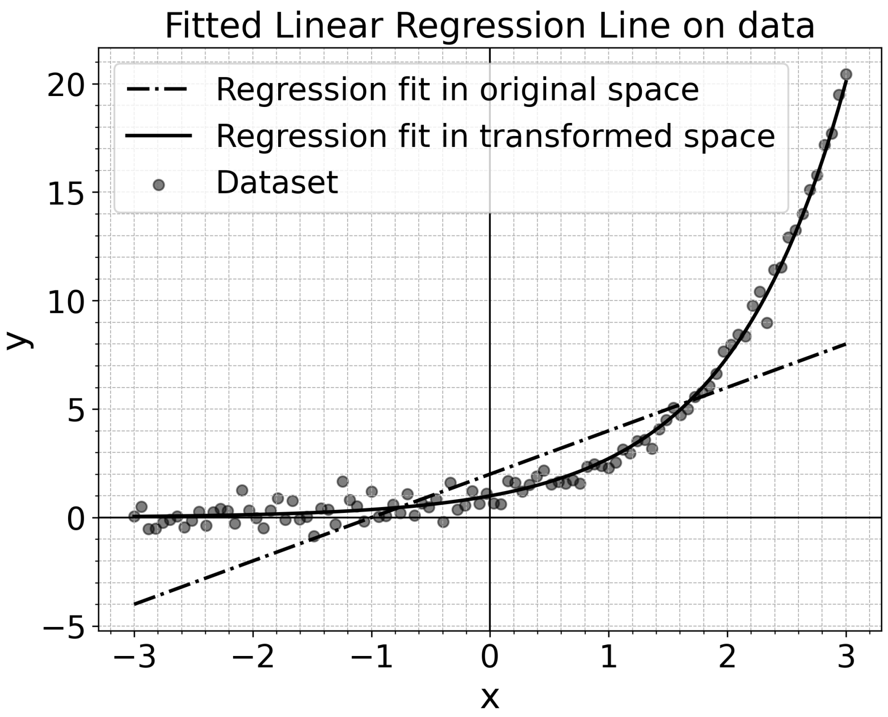
A function f is **linear** in x if it satisfies the following properties:

- **Additivity:** $f(x_1 + x_2) = f(x_1) + f(x_2)$ for all x_1 and x_2 in the domain of f.
- **Homogeneity:** $f(\alpha x) = \alpha f(x)$ for all x in the domain of f and all scalars α.

A **feature map** ϕ is a function that transforms an input vector $\mathbf{x}$ from the original feature space into a new feature space. It can be used to extract or create features that may make certain problems easier to solve. Formally, the feature map ϕ is defined as:

$$\phi : \mathbb{R}^n \to \mathcal{F}$$

where $\mathbb{R}^n$ is the original feature space and $\mathcal{F}$ is the transformed feature space (like $\mathbb{R}^m$). A problem that might appear difficult to solve in the original feature space, might easily be solved in a different feature space. As an example observe the figure below:



The straight line is a regression fit of the form $y(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$. Clearly this is not a good fit. Transforming our problem into a new feature space using the map $\phi(x) = e^x$ easily solves this problem as can be seen in the "regression fit in the transformed space" curve above. The fit will now be of form $y(\mathbf{x}) = \mathbf{w_{new}}^\top \phi(\mathbf{x}) + b_{new}$.

For the model of form:

$$y(\mathbf{x}) = \phi(\mathbf{x})^\top \mathbf{w}$$

(a) Is the function $y(\mathbf{x})$ linear in variable $\mathbf{x}$. If NO provide proper justification, if YES prove it.
[2 marks]

(b) Under what conditions is the function $y(\mathbf{x})$ linear in $\mathbf{x}$. Provide proper justification.
[2 marks]

(c) Is the function $y(\mathbf{x})$ linear in variable $\mathbf{w}$. If NO provide proper justification, if YES prove it. Comment on why the model is considered a **linear** model even if the feature map $\phi(\mathbf{x})$ can be chosen as a nonlinear function [4 marks]

Some data was collected and a regression model of the above form was fit after transforming it using an appropriate feature map. The data is plotted as a scatter plot along with the fitted curve in the figures below. Look at the plots closely and suggest the feature maps that were used for the data. Be as mathematically precise as possible

(d) Suggest a feature map $\phi(\mathbf{x})$ for data in figure 1 below [2 mark]

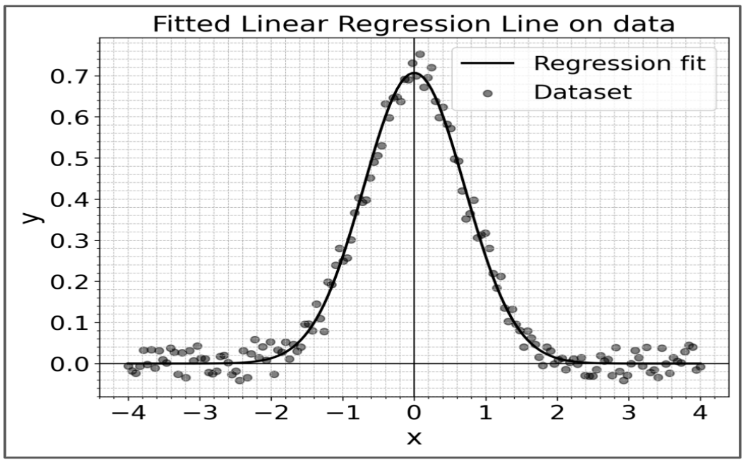


Figure 1

(e) Suggest a feature map $\phi(\mathbf{x})$ for data in figure 2 below [2 mark]

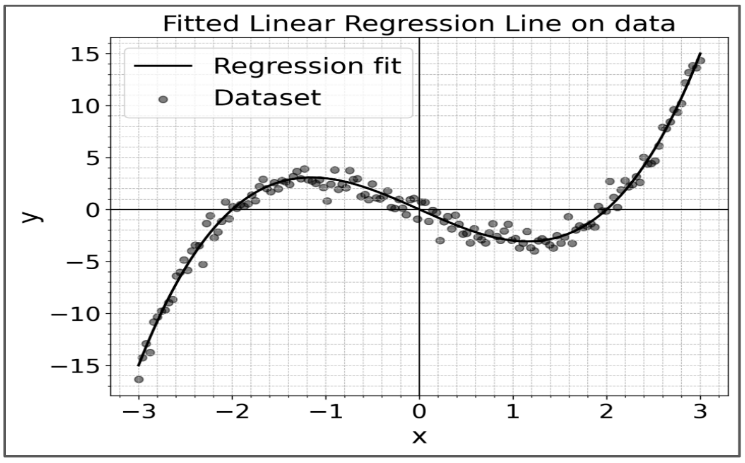


Figure 2

(f) Suggest a feature map $\phi(\mathbf{x})$ for data in figure 3 below [2 mark]

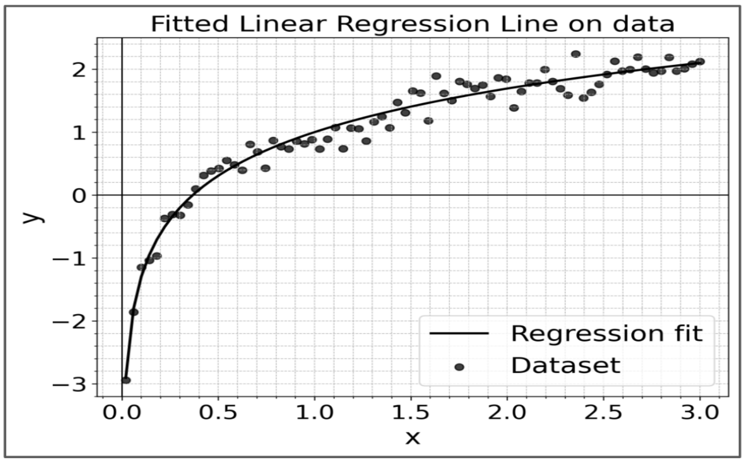


Figure 3

In order to process the data in parallel, instead of doing calculation for each sample data point, we first arrange the data in matrices and do the calculations specifically for

$$\text{Dataset } \{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$$

$$\mathbf{X} = \begin{bmatrix} \ldots\ldots\mathbf{x}^{(1)T}\ldots\ldots \\ \ldots\ldots\mathbf{x}^{(2)T}\ldots\ldots \\ \vdots \\ \ldots\ldots\mathbf{x}^{(n)T}\ldots\ldots \end{bmatrix}$$

where each $\mathbf{x}^{(i)}$ is a $d \times 1$ vector (a column vector) and $\mathbf{x}^{(i)T}$ is a $1 \times d$ row vector.

$$\mathbf{Y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$

where $y^{(i)}$ is the corresponding target value for each data point. The linear model is defined as $\widehat{\mathbf{Y}} = \mathbf{X}\mathbf{w}$ in vector form. Using a given feature map the model can be transformed as $\hat{\mathbf{Y}} = \Phi(\mathbf{X})\mathbf{w}$

(g) Clearly formulate how the matrix $\Phi(\mathbf{X})$ is written in terms of matrix $\mathbf{X}$ above [2 marks]

(h) The cost function for the transformed model $J(\mathbf{w})$ is defined as:

$$J(\mathbf{w}) = \frac{1}{2n}\|\hat{\mathbf{Y}} - \mathbf{Y}\|^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

DERIVE a closed form expression for the optimal $\mathbf{w}^*$ [6 marks]

Question 2 : TORCH

In **PyTorch** tutorial, we used the MNIST dataset, which contains grayscale images of size 28×28. Since the images have 1 channel (grayscale), a batch of images has the shape `torch.Size([batch_size, 1, 28, 28])`. Here:

- `batch_size`: Number of images in the batch.
- 1: Number of channels (1 for grayscale).
- 28, 28: Image height and width.

PyTorch in general can be used for different machine learning tasks.

(a) **TEXTUAL DATA**. Suppose we have a batch of 500 sentences, where each sentence contains 100 words, and each word is represented as a d-dimensional vector (embedding). Let this batch be denoted by **X**. What will be the output of **X**.**shape**() and **X**.**ndim**()
[1 mark]

(b) **VIDEO DATA**. Suppose we have a batch of 500 video clips, where each video contains a recording of 1 minute color camera feed, The frequency of recording is 10 frames/second. Each frame is essentially an RGB image. Let this batch be denoted by **V**. What will be the output of **V**.**shape**() and **V**.**ndim**()
[1 mark]

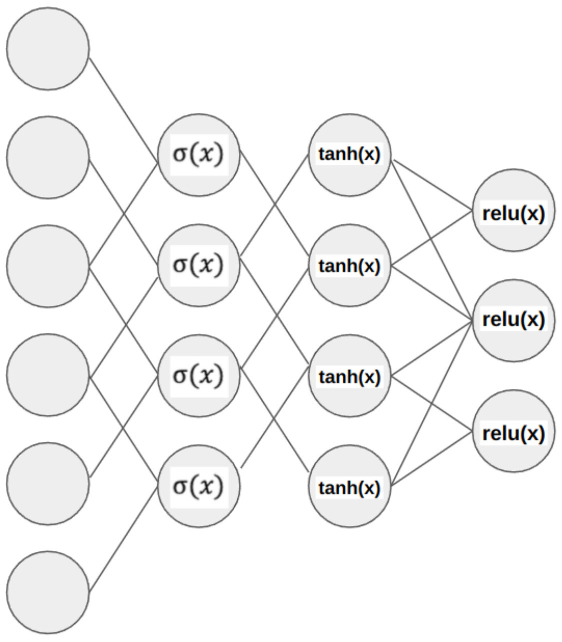
The Gradient Descent equation is written below:

$$\boxed{\theta^{(t+1)} = \theta^{(t)} - \alpha \nabla_\theta J(\theta)}$$

As discussed in the tutorials, **loss** calculates the $J(\theta)$ above according to a suitable criterion() function described for the problem.

(c) Clearly elaborate what calculation of the above equation is performed when **loss.step()** is called in the training code [1 mark]

(d) Clearly elaborate what calculation of the above equation is performed when **optimiser.step()** is called in the training code [1 mark]



The architecture of a neural network is shown in the figure above. Below is the code to implement the neural net above

```
import torch
import torch.nn as nn
import torch.nn.functional as F

class CustomNet(nn.Module):
    def __init__(self):
        super(CustomNet, self).__init__()

        self.fc1 = nn.Linear(a, b)
        self.fc2 = nn.Linear(c, d)
        self.fc3 = nn.Linear(e, f)

        self.activation1 = F.____
        self.activation2 = F.____
        self.activation3 = F.____

    def forward(self, x):
        x = self.fc1(x)
        x = self.activation1(x)
        x = self.fc2(x)
        x = self.activation3(x)
        x = self.fc3(x)
        x = self.activation2(x)
        return x
```

(e) What are the values of :
a, b, c, d, e, f, self.activation1, self.activation2 and self.activation3 [4 marks]

Quiz 1 : Solution

$\boxed{1}$

(a) $y(x) = \phi(x)^T \omega$

$\boxed{\text{NO}}$ as $\phi(\alpha x)^T \omega \neq \alpha\, \phi(x)^T \omega$
for any arbitrary map $\phi(\cdot)$, $y(x)$ is not linear in x

(b) If $\boxed{\phi(x) \text{ is itself a linear function}}$ ie

$\phi(\alpha x) = \alpha\, \phi(x)$
$\phi(x_1 + x_2) = \phi(x_1) + \phi(x_2)$

then $y(x) = \phi^T(x)\omega$ will satisfy

i) $y(\alpha x) = \phi(\alpha x)^T \omega = \alpha\, \phi(x)^T \omega = \alpha\, y(x)$

ii) $y(x_1 + x_2) = \phi(x_1 + x_2)^T \omega = (\phi(x_1) + \phi(x_2))^T \omega = \phi(x_1)^T \omega + \phi(x_2)^T \omega$
$= y(x_1) + y(x_2)$

(c) $\boxed{y(x) \text{ is Linear in } \omega}$

$\underset{\omega}{y}(x) = \phi(x)^T \omega \equiv$ ~~$y(x,\omega)$~~ $f(\omega)$

$f(\alpha \omega) = \phi(x)^T (\alpha \omega) = \alpha\, \phi(x)^T \omega = \alpha f(\omega)$
$f(\omega_1 + \omega_2) = \phi(x)^T (\omega_1 + \omega_2) = \phi(x)^T \omega_1 + \phi(x)^T \omega_2 = f(\omega_1) + f(\omega_2)$

The model is Linear (called linear) because irrespective of the feature map $\phi(\cdot)$ being linear / non linear in x the model $y = \phi(x)^T \omega$ is always linear in ω. All the formulation of Linear regression is applicable for any valid $\phi(\cdot)$ map.

(d) $$\frac{1}{\sqrt{2\pi}}\, e^{-\frac{x^2}{2}} \equiv \phi(x)$$

(e) $\phi(x) \equiv x(x-2)(x+2)$

(f) $\phi(x) = \log(ax)$

As $\phi(x) = 0$ at around $x = .4$

$\log(a * .4) = 0$

$a = \frac{1}{.4} =$ ~~2.5~~ $\frac{10}{4}$

(g)
6

$$X = \begin{bmatrix} \text{———} \; x^{(1)T} \; \text{———} \\ \text{———} \; x^{(2)T} \; \text{———} \\ \vdots \\ \text{———} \; x^{(n)T} \; \text{———} \end{bmatrix}$$

$\phi(x) : \mathbb{R}^d \rightarrow \mathbb{R}^m$ ~~→ this m can be d~~

Note that the model is $\hat{Y} = X\omega$ as the transformed model is $\Phi(X)\omega$, due to a minor error this ω should have been written

as ω_{new} that would have been of any arbitrary dimension m

If that is inferred the solution will be fine

Even if not inferred you can infer from the dot product that $m \equiv d$ for this problem. What ever the assumption is the

$$\Phi(X) = \begin{bmatrix} \text{———} \; \phi(x^{(1)})^T \; \text{———} \\ \text{———} \; \phi(x^{(2)})^T \; \text{———} \\ \vdots \\ \text{———} \; \phi(x^{(n)})^T \; \text{———} \end{bmatrix}$$

$n \times m$

Can be d
or any general m

The main idea is every vector $x^{(i)}$ needs to be passed through $\phi(\cdot)$ and stacks vertically to form $\Phi(X)$

(h) Deriving the Optimal $\mathbf{w}^*$

The cost function $J(\mathbf{w})$ is defined as:

$$J(\mathbf{w}) = \frac{1}{2n}\|\hat{\mathbf{Y}} - \mathbf{Y}\|^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

where $\hat{\mathbf{Y}} = \Phi(\mathbf{X})\mathbf{w}$. Substituting $\hat{\mathbf{Y}}$ into the cost function:

$$J(\mathbf{w}) = \frac{1}{2n}\|\Phi(\mathbf{X})\mathbf{w} - \mathbf{Y}\|^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

Expanding the squared norm term:

$$\|\Phi(\mathbf{X})\mathbf{w} - \mathbf{Y}\|^2 = (\Phi(\mathbf{X})\mathbf{w} - \mathbf{Y})^\top(\Phi(\mathbf{X})\mathbf{w} - \mathbf{Y})$$

Expanding this:

$$(\Phi(\mathbf{X})\mathbf{w} - \mathbf{Y})^\top(\Phi(\mathbf{X})\mathbf{w} - \mathbf{Y}) = \mathbf{w}^\top\Phi(\mathbf{X})^\top\Phi(\mathbf{X})\mathbf{w} - 2\mathbf{Y}^\top\Phi(\mathbf{X})\mathbf{w} + \mathbf{Y}^\top\mathbf{Y}$$

So the cost function becomes:

$$J(\mathbf{w}) = \frac{1}{2n}\left(\mathbf{w}^\top\Phi(\mathbf{X})^\top\Phi(\mathbf{X})\mathbf{w} - 2\mathbf{Y}^\top\Phi(\mathbf{X})\mathbf{w} + \mathbf{Y}^\top\mathbf{Y}\right) + \frac{\lambda}{2}\mathbf{w}^\top\mathbf{w}$$

Combining terms:

$$J(\mathbf{w}) = \frac{1}{2n}\mathbf{w}^\top\Phi(\mathbf{X})^\top\Phi(\mathbf{X})\mathbf{w} - \frac{1}{n}\mathbf{Y}^\top\Phi(\mathbf{X})\mathbf{w} + \frac{1}{2n}\mathbf{Y}^\top\mathbf{Y} + \frac{\lambda}{2}\mathbf{w}^\top\mathbf{w}$$

To minimize $J(\mathbf{w})$, we take the gradient with respect to $\mathbf{w}$ and set it to zero:

$$\nabla_{\mathbf{w}}J(\mathbf{w}) = \frac{1}{n}\Phi(\mathbf{X})^\top\Phi(\mathbf{X})\mathbf{w} - \frac{1}{n}\Phi(\mathbf{X})^\top\mathbf{Y} + \lambda\mathbf{w} = 0$$

Solving for $\mathbf{w}$:

$$\frac{1}{n}\Phi(\mathbf{X})^\top\Phi(\mathbf{X})\mathbf{w} + \lambda\mathbf{w} = \frac{1}{n}\Phi(\mathbf{X})^\top\mathbf{Y}$$

$$(\Phi(\mathbf{X})^\top\Phi(\mathbf{X}) + n\lambda I)\mathbf{w} = \Phi(\mathbf{X})^\top\mathbf{Y}$$

Thus, the closed-form solution for $\mathbf{w}^*$ is:

$$\mathbf{w}^* = (\Phi(\mathbf{X})^\top\Phi(\mathbf{X}) + n\lambda I)^{-1}\Phi(\mathbf{X})^\top\mathbf{Y}$$

$\boxed{2}$ TORCH

(a) batch size = 500
No of words/sentence = 100
~~each sentence is d-dim vector~~
word
Each word is d- dimensional vector

X. shape() = torch. Size ([500, 100, d])
X. ndim() = 3

(b) batch size = 500
(60s) 1min vid * $\frac{10\text{ frames}}{\text{second}}$ = 600 frames
RGB image = 3 Channels
Say image is of shape h x w

X. shape () = ~~500~~ torch. size ([500, 600, 3, h, w])
X. ndim() = 5

(c) A minor correction
loss. step () is not a command, loss. backward () is
> If the above fact is stated and then no solution is provided full marks will be awarded
> If loss. step() is assumed synonymous to loss. backward() and the calculation is $\vec{\nabla}_\theta J(\theta)$ as answer then full marks are awarded
> If nothing is written → NO MARKS AWARDED

(d) optimiser. step() performs the whole gradient descent update step
ie $\{ \theta^{(t+1)} := \theta^{(t)} - \alpha \vec{\nabla}_\theta J(\theta) \} \rightarrow$ this whole calculation

(e)

$a = 6$

$b = c = 4$

$d = e = 4$

$f = 3$

```
self.activation 1 = F.sigmoid()
self.activation 3 = F.tanh()
self.activation 2 = F.relu()
```

Note this

The forward pass decides the order of functions, not the naming convention that you use.