Problem 1: (Linear Regression)

Assuming our input training data is: (X1 = 2, Y1 = 8), (X2 = 5, Y2 = 25), (X3 = 3, Y3 = 9), (X3 = 10, Y3 = 40). let us consider that we use the least square to fit the data.

Problem : 1

Assuming our input training data is : $(x_1 = 2, y_1 = 8)$, $(x_2 = 5, y_2 = 25)$, $(x_3 = 3, y_3 = 9)$, $(x_3 = 10, y_3 = 40)$

Let us consider that we use the least square to fit the data.

Linear Regression :-

1a) over all the loss function :-

$$X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \qquad \beta = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} \qquad y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}$$

Here $y = X\beta + \epsilon$

$$e(\beta) = y - X\beta$$

Loss function

$$= e^T e$$

$$= (y - X\beta)^T (y - X\beta)$$

$$= (y^T - \beta^T X^T)(y - X\beta) \Rightarrow y^T y - 2\beta^T X^T y + \beta^T X^T X \beta$$

a) **e.**

The closed-form Equation is also connected to the "Project" operation

i.e $\hat{Y} = X\hat{\theta}$

$$\mathrm{Proj}_X(Y) = X(X^TX)^{-1}X^TY$$

Now, $X(X^TX)^{-1}X^T = UU^T$

$$X(X^TX^{-1})X^TY = (UU^T)Y$$

$$= U_1U_1^TY + \text{------} + U_nU_n^TY.$$

b) closed form normal equation

$$= X^T X \beta - X^T y = 0$$

$$\therefore \hat{\beta} = (X^T X)^{-1} X^T y$$

So Here $y = \begin{bmatrix} 8 \\ 25 \\ 9 \\ 40 \end{bmatrix}$; $X = \begin{bmatrix} 1 & 2 \\ 1 & 5 \\ 1 & 3 \\ 1 & 10 \end{bmatrix}$

c) closed - form equation:-

The closed-form equation is mainly used to obtain the value of θ; so that this value can be used as the prediction for the implementation of the linear Regression.

d. **Please implement and visualize your solution in Python.**

Problem 2: (PCA)

In this problem, you will use the classic MNIST digit dataset. MNIST is made up of 70,000 28 × 28 images of the handwritten digits 0–9, where 60,000 of the images have been designated for use in training algorithms and 10,000 images have been designated for testing/evaluating algorithms.

Go to this webpage http://yann.lecun.com/exdb/mnist/ and download these four files:

train-images-idx3-ubyte.gz: training set images (9912422 bytes)

train-labels-idx1-ubyte.gz: training set labels (28881 bytes)

t10k-images-idx3-ubyte.gz: test set images (1648877 bytes)

t10k-labels-idx1-ubyte.gz: test set labels (4542 bytes)

After downloading them, you likely will need to unzip them and place them in a folder. You can reshape each one of these image vectors to be 28 × 28 to display it.

You need perform PCA analysis and visualize the 2-dimensional data. You can (https://www.analyticsvidhya.com/blog/2021/11/pca-on-mnist-dataset/) to perform PCA analysis and submit your results

Problem 2: (PCA) (b)

Principal Component Analysis (PCA) is typically explained using an eigen decomposition of the $d \times d$ data covariance matrix C; however, due to finite-precision arithmetic on a computer this algorithm for PCA can be numerically unstable. In practice, Singular Value Decomposition (SVD) of the data itself (instead of the covariance matrix) is typically used, i.e., $X = U\Sigma VT$. Given a data matrix X show mathematically how SVD can be used to compute the principal components instead of using the eigen decomposition of C. Assume that X is mean zero (i.e., centered). What would be the formula for the eigenvalues using the SVD algorithm for PCA?

Hint: ***This problem requires you to use the formula for PCA using the covariance matrix C to find a formula for PCA using SVD instead.***

Let the data matrix X be of *$n \times p$* size, where *n* is the number of samples and p is the number of variables. Let us assume that it is *centered*, i.e. column means have been subtracted and are now equal to zero.

Problem(2) (PCA) (b)

"X" be a data matrix of size $n \times d$.

Co-variance Matrix C

$$C = \frac{X^T X}{(n-1)}$$

It is an Symmetric matrix & and it can be diagonalized

$C = ULV^T$ ("L" diagonal matrix with Eigen Values of λ_i)

Let $X = USV^T$: (SVD)

U = Left Singular Vector matrix

V = Right Singular Vector matrix

S = Diagonal matrix of Singular Values S_i

$$C = (VSU^T)(USV^T) / (n-1)$$

$$= V \frac{S^2}{(n-1)} U^T$$

Hence, the eigen values are given as

$$\lambda_i = \frac{S_i^2}{(n-1)}$$

Problem 3: (PCA) (d)

Plot the mean reconstruction error (squared distance between the original data and the reconstructed data) as a function of the number of principal components, i.e., from 1 to 783. To do this efficiently, you should make a new version of your function for PCA to in which k is varied. Use vectorization to ensure your code runs fast. Even if you use vectorization, expect this script to take decent amount of time to run (30-90 minutes). Make sure to label your plot's axes.

a) **Please write down the multivariate Gaussian distribution parametrized by the mean and the covariance matrix.**

b) **Can you define the Multivariate Gaussians based on linear map?**

c) **Given n independent draw of samples x1 · · · xn, derive the explicit form of the log-likelihood?**

d) **Try to derive the closed form solution based on MLE for the mean.**

3) Maximum Likehood Estimation for multivariate Gaussians

For multi variable Gausion

a) Multi variable Gaussion Distribution

$$P(X; \mu, \Sigma)$$

$$= \frac{1}{(2\pi)^{n/2} |\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)\right)$$

b) Consider

$$y = ax + b \quad \text{where } X \sim N(\mu \Sigma)$$

$$P(y) = \mu(a\mu + b, b\Sigma b^T)$$

c) For n independent Samples $x_1 \ldots\ldots x_n$

where $P(x_i) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x_i-\mu)^2\right)$

$i(x_1, x_2 \ldots\ldots x_n)$

$= P(x_1)\ P(x_2) \ldots\ldots P(x_n)$

$$= \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^n \exp\left(\frac{-1}{2\pi\sigma^2}\sum_{i=1}^{n}(x_i-\mu)^2\right)$$

d) MLE Estimation is given as

$$\mu_{MLE} = \max\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^n \exp\left[-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i-\mu)^2\right]$$

$$= \max_{\mu} -\frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i-\mu)^2$$

$$= \min_{\mu} \sum_{i=1}^{n}(x_i-\mu)^2$$

Differentiating w.r.t μ and equating to μ

then we get

$$\sum_{i=1}^{n} 2\ (\mu_i - \mu) = 0$$

$$\Rightarrow \mu = \frac{1}{n} \sum_{i=1}^{n} x_i$$