LAB 7

Q1: This question involves the use of simple linear regression on the Auto data set

Simple Linear regression model on Auto dataset.

(a) Use the lm() function to perform a simple linear regression with mpg as the response and horsepower as the predictor. Use the summary() function to print the results. Comment on the output. For example:

- Is there a relationship between the predictor and the response?
- How strong is the relationship between the predictor and the response?
- Is the relationship between the predictor and the response positive or negative?
- What is the predicted mpg associated with a horsepower of 98? What are the associated 95 % confidence and prediction intervals?

(b) Plot the response and the predictor. Use the abline() function to display the least squares regression line.

(c) Use the plot() function to produce diagnostic plots of the least squares regression fit and interpret these plots where the model fit the data?

Q2: This question involves the use of multiple linear regression on the Auto data set.

(a) Produce a scatterplot matrix which includes all of the variables in the data set.

(b) Compute the matrix of correlations between the variables using the function cor(). You will need to exclude the name variable, which is qualitative.

(c) Perform a multiple linear regression with mpg as the response and all other variables except name as the predictors. Use the summary() function to print the results. Comment on the output. For instance:

- Is there a relationship between the predictors and the response?
- Which predictors appear to have a statistically significant relationship to the response?
- What does the coefficient for the year variable suggest?

Q3: Repeat the following questions using Carseats data set.

(a) Fit a multiple regression model to predict Sales using Price, Urban, and US.

(b) Provide an interpretation of each coefficient in the model.

(g) Obtain 95% confidence intervals for the coefficient(s).

Q4: we will investigate the t-statistic for the null hypothesis H0 : $\beta = 0$ in simple linear regression without an intercept. To begin, we generate a predictor x and a response y as follows.

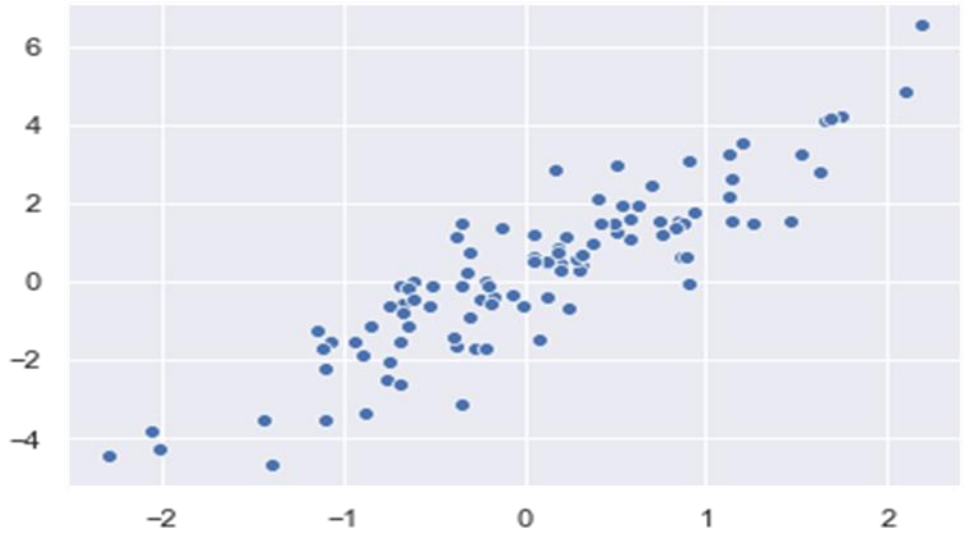
```
np.random.seed(1)

x  = np.random.normal(size=100)

y  = 2*x + np.random.normal(size=100)

df = pd.DataFrame({'x': x, 'y': y})

sns.scatterplot(x=x, y=y, color='b');
```



(a) Perform a simple linear regression of y onto x, without an intercept. Report the coefficient estimate $\hat{\beta}$, the standard error of this coefficient estimate, and the t-statistic and p-value associated with the null hypothesis H0 : $\beta = 0$. Comment on these results. (You can perform regression without an intercept using the command lm(y~x+0).)

(b) Now perform a simple linear regression of x onto y without an intercept, and report the coefficient estimate, its standard error, and the corresponding t-statistic and p-values associated with the null hypothesis H0 : $\beta = 0$. Comment on these results.

Q5: In this question you will create some simulated data and will fit simple linear regression models to it. Make sure to use set.seed(1) prior to starting part (a) to ensure consistent results

(a) Using the rnorm() function, create a vector, x, containing 100 observations drawn from a N(0,1) distribution. This represents a feature, X.

```
np.random.seed(1)

mu, sigma = 0, 1

x = np.random.normal(mu, sigma, 100)
```

(b) Using the rnorm() function, create a vector, eps, containing 100 observations drawn from a N(0,0.25) distribution i.e. a normal distribution with mean zero and variance 0.25.

mu, sigma = 0, 0.25

eps = np.random.normal(mu, sigma, 100)

(c) Using x and eps, generate a vector y according to the model¶

$Y = -1+0.5X+\varepsilon.$

What is the length of the vector y? What are the values of $\beta 0$ and $\beta 1$ in this linear model?

(d) Create a scatterplot displaying the relationship between x and y. Comment on what you observe.

(e) Fit a least squares linear model to predict y using x. Comment on the model obtained. How do $\hat{\beta}0$ and $\hat{\beta}1$ compare to $\beta 0$ and $\beta 1$?

(f) Display the least squares line on the scatterplot obtained in (d). Draw the population regression line on the plot, in a different color. Use the legend() command to create an appropriate legend.

(g) Now fit a polynomial regression model that predicts y using x and x2. Is there evidence that the quadratic term improves the model fit? Explain your answer.

(h) Repeat (a)–(f) after modifying the data generation process in such a way that there is less noise in the data. The model should remain the same. You can do this by decreasing the variance of the normal distribution used to generate the error term ε in (b). Describe your results.

(h) Repeat (a)–(f) after modifying the data generation process in such a way that there is more noise in the data. The model should remain the same. You can do this by increasing the variance of the normal distribution used to generate the error term ε in (b). Describe your results

```
np.random.seed(1)
x1 = np.random.uniform(size=100)
x2 = 0.5*x1 + np.random.randn(100)/10
y  = 2 + 2*x1 + 0.3*x2 + np.random.randn(100)
```

The last line corresponds to creating a linear model in which y is a function of x1 and x2. Write out the form of the linear model. What are the regression coefficients?

Form of the linear model is:

$\hat{y} = \beta 0 + \beta 1 x 1 + \beta 2 x 2 + \varepsilon$

(b) What is the correlation between x1 and x2? Create a scatterplot displaying the relationship between the variables.

(c) Using this data, fit a least squares regression to predict y using x1 and x2. Describe the results obtained. What are βˆ0, βˆ1, and βˆ2? How do these relate to the true β0, β1, and β2? Can you reject the null hypothesis H0 : β1 = 0? How about the null hypothesis H0 : β2 = 0?

(d) Now fit a least squares regression to predict y using only x1. Comment on your results. Can you reject the null hypothesis H0 :β1 =0?

(e) Now fit a least squares regression to predict y using only x2. Comment on your results. Can you reject the null hypothesis H0 :β1 =0?

Logistic regression Models

Q6: This question should be answered using the Weekly data set.

(a) Produce some numerical and graphical summaries of the Weekly data. Do there appear to be any patterns?

The correlation matrix shows that only 2 pairs of features are correlated:

- Year – Volume
- Today – Direction_UP

Looking at the pairplot above suggests:

- Volume increases exponentially with Year
- There is a logistic relationship between Today and Direction_Up

(b) Use the full data set to perform a logistic regression with Direction as the response and the five lag variables plus Volume as predictors. Use the summary function to print the results. Do any of the predictors appear to be statistically significant? If so, which ones?

(c) Compute the confusion matrix and overall fraction of correct predictions. Explain what the confusion matrix is telling you about the types of mistakes made by logistic regression.

(d) Now fit the logistic regression model using a training data period from 1990 to 2008, with Lag2 as the only predictor. Compute the confusion matrix and the overall fraction of correct predictions for the held-out data (that is, the data from 2009 and 2010).

Q7: Please run the notes book based on Pytorch codes for linear regression. You need to check the following theoretical portion for basic understanding.

Regression

- Linear Regression
- Batch Gradient Descent
- Stochastic Gradient Descent

output: continous

- f: actual function
- $\hat{f}$: learned function

Goal is to find a predictor $\hat{f} \in H$ that generalized well, i.e. that predicts well on test data

Train Test Split Training: 80 - 90, Test: 20 - 10

Training data still can be split to 2 more sets:

- main training data: 80 - 90
- validation: 20 - 10

Prevent overfitting

- larger dataset
- early stopping, keep track of test error using validation datase

$$\mathbf{X} = \begin{bmatrix} x_1^{(1)} & \cdots & x_n^{(1)} \\ \vdots & \ddots & \vdots \\ x_1^{(m)} & \cdots & x_n^{(m)} \end{bmatrix}, \mathbf{y} = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{bmatrix} \text{ and } \theta = \begin{bmatrix} \theta^{(1)} \\ \vdots \\ \theta^{(n)} \end{bmatrix}$$

($x^{(i)}$ is each feature column)

Minimize the **empirical risk** (average loss, **Least Squares Error**)

$$\begin{aligned} J(\theta) &= \frac{1}{N} \sum_{i=1}^{N} (\langle \theta, \mathbf{x}^{(i)} \rangle - y^{(i)})^2 \\ &= \frac{1}{N} \sum_{i=1}^{N} (\theta^T \mathbf{x}^{(i)} - y^{(i)})^2 \end{aligned}$$

Linear Regression

Our goal is to find the optimal weights such that

$$\mathcal{J}(\theta) = \underset{\theta \in \mathbb{R}^2}{\operatorname{argmin}} \{ \frac{1}{N} \sum_{i=1}^{N} (\langle \theta, \mathbf{x}^{(i)} \rangle - y^{(i)})^2 \}$$
$$= \underset{\mathbf{w} \in \mathbb{R}^2}{\operatorname{argmin}} \frac{1}{N} ||\mathbf{X}\theta - \mathbf{y}||_2^2,$$

Whenever the matrix $\mathbf{X}^T\mathbf{X}$ is invertible, the optimal weight which minimizes the empirical risk, is obtained as

$$\theta = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

- Finding the value θ that minimizes the distance between $X\theta$ and y
- is the same as finding the projection of y onto the column space of X

This is the same as finding θ such that $(X\theta - y) \perp$ every column of X i.e

$$X^\intercal(X\theta - y) = 0 \Rightarrow X^\intercal X\theta = X^\intercal y$$

which gives $\theta = (X^\intercal X)^{-1} X^\intercal y$

Batch Gradient Descent

https://towardsdatascience.com/gradient-descent-in-python-a0d07285742f

if matrix X is large, inverting $X^\intercal X$ will be computationally costly.

Instead we can find the least squares error using gradient descent. First we compute the gradient of the loss function:

$$\frac{\delta \mathcal{J}(\theta)}{\delta \theta_j} = \sum_{i=1}^{m} (x^{(i)}\theta - y^{(i)}) \frac{\delta x^{(i)}\theta}{\delta \theta_j}$$
$$= \sum_{i=1}^{m} (x^{(i)}\theta - y^{(i)}) \frac{\delta}{\delta \theta_j} \sum_{k=1}^{n} x_k^{(i)} \theta_k$$
$$= \sum_{i=1}^{m} (x^{(i)}\theta - y^{(i)}) x_j^{(i)}$$

We then use it to perform the update rule

$$\theta_j(t+1) = \theta_j(t) - \alpha \sum_{i=1}^{m} (x^{(i)}\theta(t) - y^{(i)}) x_j^{(i)}, \; j = 1, \ldots, n$$

until the change in the loss function is below a certain a preset tolerance. In matrix notation the update role can be written as

$$\theta_j(t+1) = \theta_j(t) - \alpha(X^\intercal X\theta(t) - X^\intercal y)$$

- This is also known as batch gradient descent, where all m training points are used at every step.
- For linear regression, the loss function is convex and has only one global minimum, so convergence is guaranteed unless the learning rate α is too large.

Interpretatiom

- α: Size of Steps took in any direction, Learning rate
- $\frac{\delta J(\theta)}{\delta \theta_j} = (X^\intercal X\theta(t) - X^\intercal y)$: Gradient, The direction of your steps
- $J(\theta)$: Cost function

Update rule

$$\theta_j(t+1) = \theta_j(t) - \alpha(X^\intercal X\theta(t) - X^\intercal y)$$
$$\theta_j(t+1) = \theta_j(t) - \alpha(X^\intercal(X\theta - y))$$

Stochastic Gradient Descent

"Mini batch"

This approach uses random samples but in batches.

We calculate the gradients for a group of observations which results in a faster optimization

A simple way to implement is to shuffle the observations and then create batches and then proceed with gradient descent using batches.

Q8: Run notebook based on Pytorch for logistic regression. Theory is given below.

Sigmoid function

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

Hypothesis function

$$h_\theta(x^{(i)}) = \sigma(\langle \theta, x^{(i)} \rangle)$$

Update rule: for $j = 1, \ldots, d$

$$\theta_j(t+1) = \theta_j(t) + \alpha \cdot gradient_j$$
$$gradient_j = \sum_{i=1}^{N} (y^{(i)} - \alpha(\langle \theta, x^{(i)} \rangle))x_j^{(i)}$$

$$gradient = \begin{bmatrix} gradient_1 \\ gradient_2 \\ \vdots \\ gradient_N \end{bmatrix} = \begin{bmatrix} x^{(1)} \\ x^{(2)} \\ \vdots \\ x^{(N)} \end{bmatrix}^{\mathrm{T}} \left(\begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(N)} \end{bmatrix} - \sigma\left(\begin{bmatrix} x^{(1)} \\ x^{(2)} \\ \vdots \\ x^{(N)} \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_d \end{bmatrix} \right) \right)$$
$$gradient = X^{\mathrm{T}}(y - \sigma(X\theta))$$