Problem Set 1

June 2024

1 Correlation and independence

1.1 Part I

(3pts) Show that if two random variables X and Y are independent, the expression $E(XY) = E(X)E(Y)$

1.2 Part II

(3pts) Show that if two random variables X and Y are independent, the correlation coefficient $\rho_{X\ Y} = 0$.

1.3 Part III

(2pts) Which one(s) of the following plots in figure 1 show(s) two random variables X and Y where $\rho_{XY} \approx 0$?

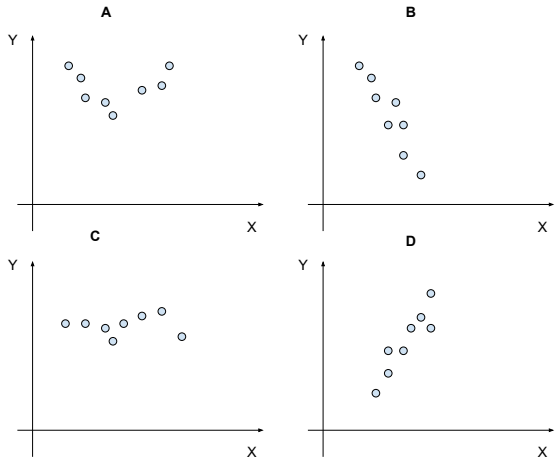

Figure 1: Problem 1 Part III

2 Preliminary de inition (No answer needed)

An MLE estimator given a data set $(x_i, y_i)_{i=1}^n$ and a conditional distribution of Y given X is the parameter β that maximizes the likelihood

$$L_{MLE}(\beta) = f_1(\{y_1, y_2, ..., y_n | \beta, x_1, x_2, ..., x_n\})$$

for easy of computation, we often take the logarithm of both sides (the resulting expression is called the log-likelihood), and since log is a monotonically increasing function over real numbers, we can maximize the log-likelihood instead.

An MAP estimator also maximizes a likelihood function given a data set, but here we assume you know a prior distribution of β_0 and your objective is to maximize the likelihood

$$L_{MAP}(\beta) = f_2(\beta | x_1, x_2, ..., x_n, y_1, ..., y_n)$$

Similarly, for ease of computation, we also often maximize the log-likelihood for $f_2(\beta | x_1, x_2, ..., x_n, y_1, ..., y_n)$

Formally,

$$\beta_{MLE} = \arg\max L_{MLE}(\beta)$$
$$\beta_{MAP} = \arg\max L_{MAP}(\beta)$$

3 OLS and MLE

In this problem, we will explore how the OLS estimator can be derived as the maximum likelihood estimator of a nomral distribution.

In general, the set up of an OLS problem is the following. Take any data matrix $A_{n\times p}$ and $B_{n\times 1}$ where A is the vertical concatenation of n p-dimensional covariate samples (x's), and B is the vertical concatenation of n y's. We try to find a p dimensional vector w that minimizes the mean squared error (MSE)

$$MSE_{A,B} = \frac{1}{n}(Aw - B)'(Aw - B)$$

3.1 Multiple covariates

3.1.1 (6pts)

Suppose we have n i.i.d data points $(x_i, y_i)_{i=1}^n$ where $y_i \in \mathbb{R}$,$x_i \in \mathbb{R}^p$, and $Y \sim N(X'\beta_0, \sigma^2)$. Find the log-likelihood function of this data set given any $\beta \in \mathbb{R}^p$.

3.1.2 (6pts)

Find the maximum likelihood estimator of β_0 where $\beta \in \mathbb{R}^p$. A maximum likelihood estimator $\hat{\beta}$ is a p-dimensional real vector that maximizes the log-likelihood function of the given data set. Compare this result of the close form solution of the OLS estimator derived from MSE minimization.

4 MAP with Ridge Regression

Now consider the setting of problem 1, but with some additional knowledge about the data generating process. Suppose you know prior to running the OLS that the true value of $\beta_0 \sim N(\mu, \Sigma)$. In this setting, we can construct a maximium a posterior estimator (MAP). The MAP estimator maximizes the probability of the estimator β given the data set.

4.1 Set up

Let ν denote the data, use Bayes rule to derive the posterior likelihood $f(\beta|\nu)$ as a function of the conditional probability density distribution $f(\nu|\beta)$ and the prior density distribution $g(b) = Pr(\beta = b)$. Show that maximizing this posterior likelihood is equivalent to maximizing $f(\nu|\cdot)g(\cdot)$

$$Pr(A\&B) = Pr(A)Pr(B|A)$$

4.2 Single covariate (5pts)

Now suppose the data is a set of n single dimensional real pairs $(x_i, y_i)_{i=1}^n$ where $Y \sim N(\beta_0 X, \sigma^2)$ and $\beta_0 \sim N(\mu, \sigma_1^2)$. Formulate the maximum likelihood problem as an equivalent problem of minimizing the ridge MSE of this data set $(A\beta - B)'(A\beta - B) + \lambda(\beta - \mu)^2$.

4.3 Multiple covariates (5pts)

Now suppose each x_i is a p-dimensional data point, $Y \sim N(X'\beta_0, \sigma^2)$, and $\beta_0 \sim N(\mu, \Sigma_1^2)$ where Σ_1^2 is a p by p diagonal matrix with a constant diagonal entry. Show that the MAP estimator is equivalent to the ridge regression estimator. Discuss how λ is related to the prior on β_0 i.e. what happens if you have a very tight versus loose prior on β_0.

5 Case studies (10pts if completed and submitted before the deadline)

In this section we will do a case study using Tesla and a simulated data set. First, we will extract Tesla's stock prices from 2011 to 2015 using the yahoo finance package in Python. We will do some data cleaning and basic calculation of returns and volatility. Then we will use a simulated example to run a classification task using logistic regression and OLS.

[ ]:
```
%matplotlib inline
%config InlineBackend.figure_format = 'retina'
```

[ ]:
```
import matplotlib.pyplot as plt
import warnings

plt.style.use('seaborn')
# plt.style.use('seaborn-colorblind') #alternative
# plt.rcParams['figure.figsize'] = [16, 9]
plt.rcParams['figure.dpi'] = 300
warnings.simplefilter(action='ignore', category=FutureWarning)
```

1 Programming Exercise 1

In this assigment, you will explore some features of real stock data using Tesla as an example.

1.1 Step 1:

Execute the following steps to download data from Yahoo Finance.

1. Install and Import the libraries: pandas and yfinance (1pt)

[ ]:
```
#install libraries
```

[ ]:
```
#Import pandas and yhfinance
```

2. Download the data: Apple stock prices from 2011 to 2015 (2pts)

[ ]:
```

```

3. Inspect the data: Print the dimensions of the dataframe and show the first 5 rows of the dataframe. (1pt)

[ ]:
```

```

2 Converting prices to returns

Create a dataframe called dftmp that only contains the adj close column. Calculate the simple and log return.

[ ]:
```
#create dftmp (1pt)
```

[ ]:
```
#calculate simple return (2pts)
```

[ ]:
```
#calculate log return (1pt)
```

Inspect the output: Display the first 5 rows of dftmp. Your returns of the first row should be Not A Number. Please explain why.

[ ]:
```
#display the first 5 rows (2pts)
```

Why are the returns of the first day NaNs?

3 Annualized volatility

In this problem, we will compute the annualized monthly volatility. First, given we already have the returns of each day, we can compute the volatility (standard deviation) of the stock in each month. Then, we can mutiply a constant factor to these volatilities to annualize these monthly volatility numbers.

1. Import the libraries: (Optional can be done that the beginning of the file)

[ ]:
```

```

2. Define the function for calculating the realized volatility: (2pts)

[ ]:
```

```

3. Calculate monthly realized volatility: (2pts)

[ ]:
```

```

4. Annualize the values: (2pts)

[ ]:
```

```

5. Plot the results: (2pts)

[ ]:
```

```

For this problem, we assume we already have a DataFrame called `dftmp` with three columns (`adj_close`, `simple_rtn`, and `log_rtn`) and dates set as the index.

1. Plot the simple return, log return, and the closing price on the same graph. Observe the closing price graph and choose an appropriate window to display these graphs. Hint: There

is a major change in the shape of the closing price graph. Split the data at that point of time and plot the two sides separately. (4pts)

[ ]:

2. Compare the log-return plot and simple return plot. Under what condition(s) are simple and log returns similar and why? Please provide a short mathematical explanantion. You can type your answer here or in the same document as the written portion of this assignment. (2pts)

#####PLACEHOLDER for 2: PLEASE PUT YOUR ANSWER HERE

4 Identifying outliers

In this section, we will continue to use the dftmp dataframe to identify outliers of Tesla from 2011 to 2015.

1. Download the data and calculate the rolling mean and standard deviation for a window of 15 days: (2pts)

[ ]:

2. Join the rolling metrics to the original data (window size is 15 data points): (2pts)

[ ]:

3. Define a function for detecting outliers using a nσ rule where $\sigma \in \{2, 3, 4\}$: (2pts)

[ ]:

4. Identify the outliers using $\sigma = 2$ and extract their values for later use: (2pts)

[ ]:

5. Plot the results: (2pts)

[ ]:

6. Now change the time window to 45 data points, and redo the exercise. Do we get more or fewer outliers and why? (2pts)

[ ]:

5 Programming exercise 2

While sometimes it is easy to calculate a close form solution for linear regression, such explicit solutions may not be analytically solveable for nonlinear regressions. In this exercise, we use one of the classic methods of numerical optimization–steepest decent–to approximate a solution to a logistic regression problem.

More specifically, we use logistic regression to classify two types of data. Let Y_i denote the true label of the ith data point and $\hat{Y}_i$ be the ith label predicted by the logistic regression model. The objective function is

$$J(w) = -\frac{1}{n}\sum_{i=1}^{n} Y_i \log(\hat{Y}_i) + (1 - Y_i)\log(1 - \hat{Y}_i)$$

[ ]:
```
#implement a function that returns the value of the objective function given an␣
 ↪array with true labels and another array with predicted labels (2pts)
```

[ ]:
```
#implement a function that computes the direction of the steepest decent given␣
 ↪an array of true labels and another array of predicted labels and a training␣
 ↪design matrix X.
#The design matrix X is a matrix of covariates of the training data.
#this function should return the updated weights after taking a step in the␣
 ↪steepest decent direction (2pts)
```

[ ]:
```
#implement the prediction function that returns the predicted Y given a data␣
 ↪point X  (2pts)
```

[ ]:
```
#implement the full training framework
#randomly initialize weights->get predictions->update weights->get␣
 ↪prediction->update weights->...->max iteration reached output J(w) (2pts)
```

[ ]:
```
#DO NOT CHANGE THE CODE IN THIS BLOCK PLEASE
#test your code with the following example and plot the data points with two␣
 ↪colors that represent the different classes
#it may be helpful to plot the points first. run this block to generate the␣
 ↪test example and plot.
import numpy as np
import random
import matplotlib.pyplot as plt
random.seed(123)
x1=random.choices([i for i in range(100)], k=500)
y1=random.choices([i for i in range(10,100)], k=500)
x2=random.choices([i for i in range(100)], k=500)
y2=random.choices([-i for i in range(10,100)], k=500)
X=x1+x2
Y=y1+y2
#Design matrix
M=np.array([X,Y]).T
labels=np.array([[0 for i in range(500)]+[1 for i in range(500)]]).T
theta=np.pi/4
c, s = np.cos(theta), np.sin(theta)
M=np.matmul(np.array(((c, -s), (s, c))),M.T).T
plt.scatter(M[:500,0],M[:500,1])
plt.scatter(M[500:,0],M[500:,1])
```

[1]:
```
#test your logistic regression. report the final J(w) after 50 iterations with␣
↪step size {0.05,0.01,0.02} (2pts)
```

The reason why it is useful to plot the data points first is that sometimes classification can be done with even simpler models. For example, in this case, a simple linear regression should give us good classification results. Implement OLS without using built-in models like sklearn (NumPy is allowed). Report the classification accuracy of OLS.

[ ]:
```
#OLS (4pts)
```

[ ]:
```
#Report accuracy and plot the data using the predictions of OLS (2pts)
```

Additional problems

1 Correlation and out-of-sample R^2 (15 points)

Given a training dataset $(X_{\text{train}}, Y_{\text{train}})$ and a testing dataset $(X_{\text{test}}, Y_{\text{test}})$ and a model $f(x)$ trained using the training dataset.

1. Assume the true data generation process is $Y \sim N(\mu, \sigma)$, and the trained model is a linear model $\hat{Y} = \alpha + \beta X$ where $\hat{Y}, X, \alpha$, and β are all scalars. Assuming the sample sizes are large for both training and testing, what is the out-of-sample R^2 of this model? What is the out-of-sample correlation of this model?

2. Regime change is one of the major challenges in analyzing asset prices. In particular, the underlying function of prices may change at some point in time. Here, we explore why this is a critical issue in model machine-learning models for financial data. Assume you have a model trained on data from time t to $t + N$. During this time, the true data-generating process of the price of an asset is $p_i = \alpha + \beta x_i + \epsilon_i$ where i is the time index, α is a constant scalar coefficient, β is a vector of coefficients on a set of factors at time i x_i, and $\epsilon_i \sim N(0, \sigma^2)$. Let x_i be random draws from a distribution $N(\mu, \sigma_x^2)$ which does not change over time.

   (a) First, derive a formula for a linear regression model built on the training data based on the specified true data generating process and distribution of x_i.

   (b) Assume you have enough data such that the variance on $\hat{\beta}$ and $\hat{\alpha}$ are negligible (your estimates are exactly the same as the true coefficients). However, when you test this model from $t + N + 1$ onward, the true model becomes $p_i = \alpha + \delta + \beta x_i + \epsilon_i$ (an additional constant shift δ). Compute the correlation between the predicted and realized prices in the test set, and compute the out-of-sample R^2. What happens to the two values when $|\delta|$ is large?

2 Additional coding problem (45 points)

This problem will use the dataset "$230P\_PS1_data.parquet$". This dataset contains 6 columns. 5 columns are independent variables and the outcome column

is the dependent variable. Use feature engineering (up to 3rd degree) and model selection to choose the best linear model that predicts the outcome column.

- category: a categorical variable that takes a value in the set $\{1, 2, 3\}$.
- *macro_state_1*: a continuous state variable. You can think about it like a macroeconomic variable in stock price prediction.
- *macro_state_1*: a continuous state variable. You can think about it like another macroeconomic variable in stock price prediction.
- *innovation_success*: a discrete numerical variable that takes integer values from 0 to 10.
- year: a categorical variable that takes a value in the set $\{1, 2, 3, 4\}$ (4 is only in the test set). You can think of it as a variable denoting the time.

Your goal is to achieve the highest out-of-sample r^2 when your model is applied to a test set while keeping the model simple.

Your submission file should be a python function where the input is a pandas dataframe with columns category, *macro_state_1*, *macro_state_2*, *innovation_success*, year, and outcome. Your function should return three values: the number of features used in the linear model (if you use the 5 provided columns, this number would be 5), the predicted outcome values, and the out-of-sample r^2 (true test outcomes vs. predicted outcomes).

Your grade for this problem will be 30% of the whole problem set, and you will be graded on a relative basis. A group with a higher r^2 will receive a higher grade, and when two groups are tied, the group with fewer features will receive a higher grade. A code that cannot be run from start to finish will automatically receive the lowest grade. The test dataframe will have the same columns as the provided training and validation data.