CP - Midterm - 2020

October 6, 2020

1 CP - Midterm - 2020

1.1 Instruction

- Modify this file to be Midterm-<Your FirstName-[First Letter of Last Name]>, e.g., Midterm-Chaklam-S.ipynb
- This exam accounts for 25% of the overall course assessment.
- This exam is open-booked; open-internet.
- You ARE NOT allowed to use sklearn or any libraries, unless stated.
- The completed exams shall be submitted at the Google Classroom
- All code should be **complemented with comments**, unless it's really obvious. **I and Joe reserve the privilege to give you zero for any part of the question where the benefit of doubt is not justified**

1.2 Examination Rules:

- For **offline** students, you may leave the room temporarily with the approval and supervision of the proctors. No extra time will be added to the exam in such cases.
- For **online** students, you are required to turn on your webcam during the entire period of the exam time
- Students will be allowed to leave at the **earliest 45 minutes** after the exam has started
- **All work should belong to you**. A student should NOT engage in the following activities which proctors reserve the right to interpret any of such act as academic dishonesty without questioning:
  - Chatting with any human beings physically or via online methods
  - Plagiarism of any sort, i.e., copying from internet sources or friends. **Both copee and copier shall be given a minimum penalty of zero mark for that particular question or the whole exam.**
- No make-up exams are allowed. Special considerations may be given upon a valid reason on unpredictable events such as accidents or serious sickness.

1.3 Question 1 (21 pts)

1). **The rabbit**: (5pts)

Once upon a time, there is a father rabbit lives in a far away jungle. Everyday, the father rabbit has to go out and find some carrots for his family. In his family there are mother rabbit, grampa rabbit, sister rabbit, and his son. In total there are 5 rabbits to feed. In one day, the adult rabbits (himself, mother rabbit and sister rabbit) will eat 3 carrots while the elderly eat 2 carrots and baby rabbit eat 1 carrot.

Unfortunately, the carrots are not easy to find. The father rabbit has to travel into the scary jungle and find some carrot then bring them back to the family before the sunset at 6PM.

- Every 1 km, the rabbit will find 3 carrots.
- The rabbit will use 1 hour to travel 1 km.

In summary, in order to find the least number of carrot for each day, the rabbit will have to use (3 + 3 + 3 + 2 + 1)/3 = 4 hours. This mean that he has to leave the house at the latest 10AM (4 hours for go out and another 4 for comming back).

This daily work has to be done exactly on time, leaving to late will cause whether his life or his family life. Would you like to help the rabbit?

[ ]:
```
# print 'yes' to help the rabbit or 'no' to refuse the challenge. (if yes -> 1␣
↪pt)
```

Good to hear that young programmer!!

What I have in mind is to build a clock that when the rabbit puts the number of (adult, elderly, young) rabbit, it will calculate how many hours is required for a travel. Of cause we have to make it as a function because the number of each rabbit type will change over the time.

- Write a function carrot that takes three integers as an input in follow this format (adult, elderly, young) (1pt)
- The function will calculate number of hour required for travelling a day. (1pt)
- The function will also calcualte the time to leave. Think of it as an alarm clock for leaving the house (1pt)
- The function will return a tuple (#hours, #time) (1pt)

[ ]:
```
# Your code here
```

2). **Print the shape**: (16pts)

- Write a function square that takes integer as an input. (1pt)
- The function will return a string of * in the shape of a square with both width and height equal to the input interger. (2pts)

[ ]:
```
'''
Level: 3
***
***
***

Level: 5
*****
*****
*****
*****
*****
'''
```

```
[ ]: # Your code here
```

- Write a function triangle that takes integer as an input. (1pt)
- The function will return a string of * in the shape of a triangle with level equal to the input integer. (2pts)

```
[ ]: '''
     Level: 3
       *
      **
     ***

     Level: 5
         *
        **
       ***
      ****
     *****
     '''
```

```
[ ]: # Your code here
```

- Write a function pyramid that takes integer as an input. (1pt)
- The function will return the string of * in the shape of a pyramid with level equal to the input interger. (2pts)

```
[ ]: '''
     Level: 3
       *
      ***
     *****

     Level: 5
         *
        ***
       *****
      *******
     *********
     '''
```

```
[ ]: # Your code here
```

Now, let's combine the three algorithms into one single class. - Create a class named MyShape that can do the followings - Take two arguments during the class construction. The first one is an integer and the second one is a string. The names are level and shape (1pt) - Check the input arguments whether the interger is in the range of [1,10] and string is in the set of {'squ','tri','pyr'}. Raise a ValueError. (2pts) - Both attributes should be able to change via a set method **only**. set[attrName] (1pt) - Of cause, the set method should check the out of range too. (1pt) - To check the current setting, write a get method. get[attrName] (1pt) - Print the shape with method show. It should return the string of the current shape with the correct level (1pt)

```
[ ]: '''
     Example 1

     >>> ms = MyShape(2,'tri')
     >>> ms.show()
      *
     **
     >>> ms.setLevel(3)
     >>> ms.setShape('squ')
     >>> ms.show()
     ***
     ***
     ***
     >>> ms.setShape('a')
     Traceback (most recent call last):
       File "<stdin>", line 1, in <module>
     ValueError: ..........
     >>>
     '''
```

```
[5]: # Your code here
     class MyShape:
         def a(self,param):
             square(param)
     #         print(param)

     b = MyShape()
     b.a('a')
```

```
a
```

1.4 Question 2 (10 pts)

2). **ML Skill**

$$y = ax + b$$

The above equation is your favorite linear equation where a,b are the constant value indicate the slope and the offset of the line in the graph.

We all know given and two points.

$$(x_1, y_1)(x_2, y_2)$$

you can find a,b very easy using Geometry

$$a = \frac{y_2 - y_1}{x_2 - x_1}$$
$$b = y_i - ax_i$$

Since we have learnt that using LinearRegression can find the value of the a,b too.

Now, do the followings.

- Write a function drawLine that takes two tuples as inputs. (1pt)
- Calculate a,b using Geometry. (1pt)
- Draw the first graph with scatter on the given two points and a line. (1pt)
- Calculate a,b using LinearRegression with Batch Gradient Descent.
  - Generate 1000 sample data along the line. (1pt)
  - Regress on the data using LinearRegression-BatchGradientDescent (2pts)
- Draw the second graph with scatter on the given two points and a line. (1pt)
- Does both method yeild the same outcome? Which method runs faster? (use timeit) (1pt)
- What will happen if the data is normalize first? (draw another graph and timeit) (1pt)
- What will happen if the data is standardize first? (draw another graph and timeit) (1pt)

[3]:
```
# Your code here
```

1.5 Question 3 (69 pts)

1). **Exploratory Data Analysis**: - Load the data "howlongwelive.csv" to pandas and print the first 5 and last 5 rows of data (1 or 0pt)

- Print the shape, feature names, and summary (describe) of the data (1 or 0pt)
- Check whether there is missing data. (1 or 0pt)
- Fix all missing data using means or mode (1 or 0pt)
- Since Hepatatis B has a lot of nans, and highly correlate with Diptheria, simply drop column Hepatatis. Also drop column Population since there are way too many nans (1 or 0pt)
- If there are any features which are string and you want to use them as features, we need to convert them to int or float. For now, convert Status to 0 or 1 (1 or 0pt)
- Rename column thinness_1-19_years to thinness_10-19_years (1 or 0pt)
- Perform a groupby country and plot their life expectancy. Which country has the lowest/highest life expectancy? (1 or 0pt)
- Plot average life expectancy of developed country vs. developing country. (1 or 0pt)
- Perform a t-test of life expectancy between developed and developing countries. Is the result significant? (1 or 0pt)
- Perform a pairplot to see which features are likely to have strong predictive power for life expectancy. Identify the most 3 important features. (1 or 0pt)
- Perform a histogram of life expectancy. Is it normal? (1 or 0pt)

2). **Regression** - Prepare your X and y into Numpy array (you have to map from Pandas to numpy). For X, prepare two versions of them. For first X_selected, you have to choose the most 3 important features from above, and for second X_all, simply use all features (you may want to omit Country since they are categorical). Set y to life expectancy. (1 or 0pt)

- Perform standardization using Numpy way (NOT sklearn way). (1 or 0pt)

- Perform train-test split by using Numpy way (NOT sklearn way). Use test size of 0.3. (1 or 0pt)
- Perform assertion whether your splitting is correct accordingly (1 or 0pt)
- Write a class Regression(X, y, grad_method, max_iter, alpha, tol, decay, decay_iter, decay_rate, stop_delay_counter, verbose, lam, poly, poly_deg) that can perform the followings:
  - Mini-batch, Stochastic, and Batch Gradient Descent (each 2pts)
  - Polynomial of degree k (2 or 0pt)
  - Decay learning rate (1 or 0pt)
    * Decay learning rate is a learning rate that becomes smaller after certian iteration. For example, after 5 iterations, the learning rate will reduce to 95% of the current learning rate.
    * To implement it, simply multiply current learning rate with some constant decay_rate. For now, set it to 0.9
  - Regularization with ridge (2 or 0pt)
  - Must have at least four methods for fit() (i.e., for finding weights) predict() (i.e., for predicting X_test data), score() (i.e., for returning r^2 score), and mse() (return mse) (each 1pt)
  - Accepts X, y, grad_method (default set to "batch"), alpha (learning rate), max_iter, tol, decay (whether to use decay learning rate; default set to False), decay_iter (after how many iterations will the decay apply), stop_delay_counter (this is the maximum number of times that decay the learning rate), verbose (default is set to False, whether model will display the Cost for each iteration), lam (this is the ridge regularization parameter), poly (default is set to False), and poly_deg (default is set to 2) (each 1/13pt)
- Create the following 3 models **from your class** (For any unspecified parameters, feel free to use any :D)
  1. For the first model, transform your feature using polynomial degree 3, then perform linear regression with batch gradient descent with early stopping of tol 1e-3 (1 or 0pt)
  2. For the second model, perform linear regression with mini-batch gradient desent with early stopping of tol 1e-3 (1 or 0pt)
  3. For the third model, perform ridge regression with stochastic gradient desent with early stopping of tol 1e-3 and decay set to True and lam to 1e-4 (1 or 0pt)
- Create Lasso model from Sklearn with default parameters (1 or 0pt)
- For these four models, using two different versions of X, perform a cross validation of 10 folds, comparing the four models * two versions of X. Here you should implement cross validation. Report which one is the best candidate model (3pts for implement from scratch or 1pt for using sklearn)
  - Recall that in a 10 folds cross validation, you split your data into 10 even pieces. Then you run 10 iterations, where in each iteration, you pick 1 of this piece as the validation set, and the rest as training set. Once you reach the 10th iteration, you would have already exhaust all the 10 pieces as validation set.
- Using the best model, fit again with the training data. Plot the weights using bar charts along the feature names. Before you actually plot the weights, we need to

multiply these weights by their feature standard deviation, so to reduce these weights to same unit of measure. Interpret these weights and what they imply. (For those who are curious why we need to multiply with std, you may read this > https://scikit-learn.org/stable/auto_examples/inspection/plot_linear_model_coefficient_interpretation.html#interpreti coefficients-scale-matters (2 or 0pt)

- Perform predictions on testing data. Print adjusted r^2 and mse. (1 or 0pt)
- Plot the predicted values against actual values (1 or 0pt)

3). **Classification**

- Change your y to discrete value. Here split y into three class, {0, 1, 2}, where 0 belongs to low life expectancy group, and 2 for the high life expectancy group. (1 or 0pt)
- Write a class for multinomial logistic regression with stochastic gradient descent. Must have at least six methods for fit() (i.e., for finding weights) predict() (i.e., for predicting X_test data), accuracy() (i.e., for returning accuracy score), recall(), precision(), and f1() (each 1pt)
- Using the best X_train of the two suggested by the cross validation step, fit the data with your class. (1 or 0pt)
- Perform predictions on testing data. Print accuracy, recall, precision, and f1_score from your class. (1 or 0pt)
- Plot the decision boundary with the X_test data. To plot this, you may want to choose only 2 features. (1 or 0pt)

4). **Final verdict**

- Attempt to do whatever ways - including sklearn or scratch - or change your features, or do feature enginnering such that your mse is lowest possible. (0 to 5pts - following class normal distributions)

```
[2]: # Your code here
```