[CS 376] Machine Learning

Assignment #1: Regression, PCA and Clustering

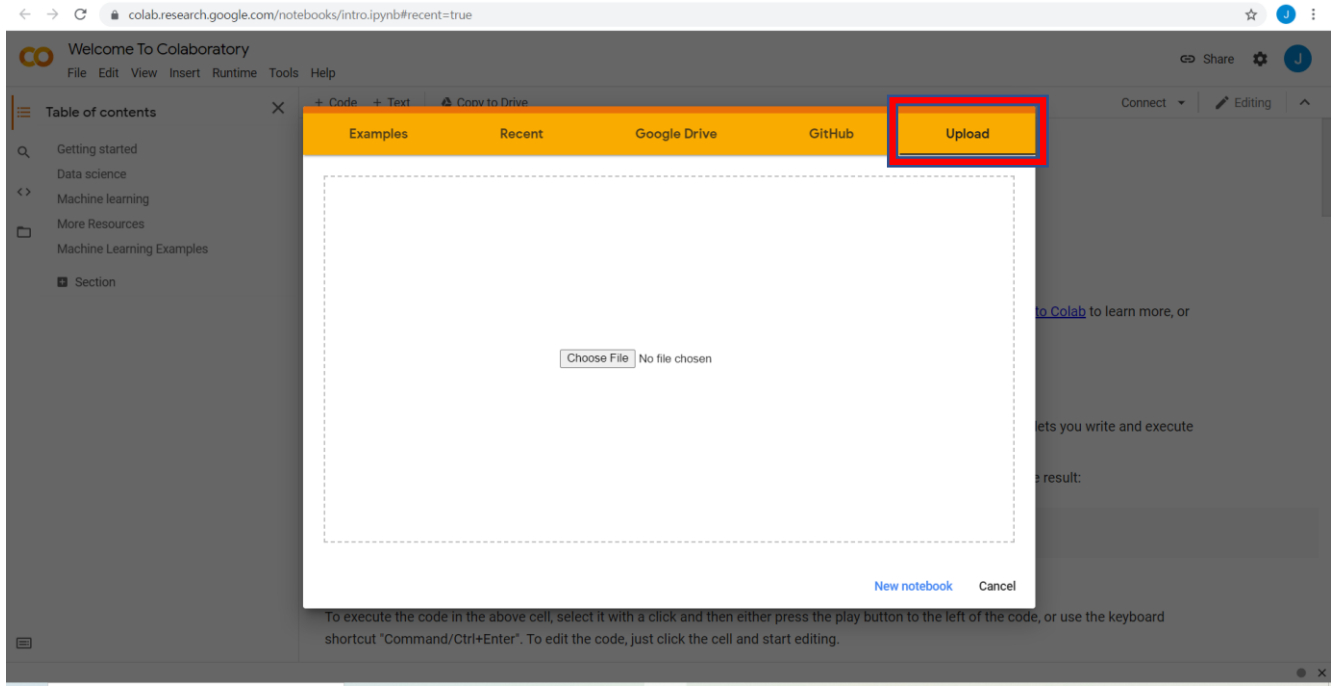
Assignment 1

Google Colab

In this assignment, we will provide you files in ipynb (Jupyter Notebook) format. You can open this file with Google Colab, which you can use freely with a Google account.

First, go to https://colab.research.google.com. Then, press the "Upload" tab and upload the provided ipynb file. Now, you can read and make changes to it.

A folder named Colab Notebooks will be created in your drive, so you can access uploaded notebooks here.



Assignment 1

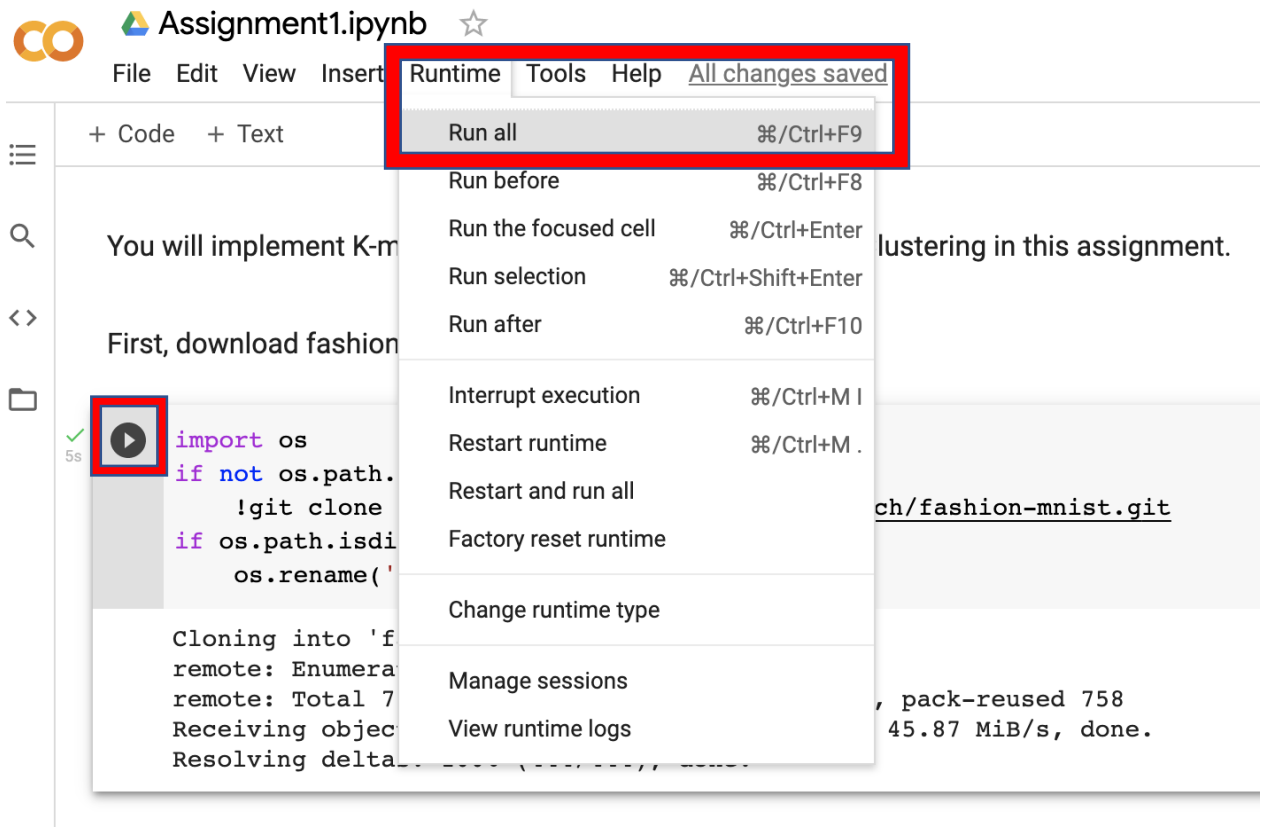
Google Colab

Here are some basic tips for running notebooks in Google Colab.

If you want to run the entire code in the notebook, you can either press Ctrl+F9, or go through the tabs "Runtime"->"Run all".

Blocks in notebooks are called cells. If you want to run a single cell, press Ctrl+Enter, or press the Play button, which is located at the top-left corner of each cell.

Variables in previous cells are preserved once you run them, so you do not have to rerun all cells every time you change and run a single cell.



Assignment 1

Numpy
In this assignment, you will need to handle multi-dimensional data. Numpy is a Python library for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

Here are some basic functions in Numpy that you may find useful for this assignment. For more details on these functions, check the official Numpy documentation.

Notations
a, b: Numpy arrays. n, k: integers. np: numpy (after running the line "import numpy as np").

np.sum(a, axis = None): add all elements of a. If axis is specified, sum is performed along that axis.
np.shape(a): returns the shape (dimensions) of a.
np.random.randint(n): returns random integer in range [0,n).
np.expand_dims(a, axis = k): expand the shape of a. Insert a new axis that will appear at position k in the expanded shape.
np.linalg.norm(a, axis = None): get matrix/vector norm of a. If axis is specified, norm will be performed along that axis.
np.argmin(a, axis = None): get index of minimum element in a. If axis is specified, argmin will be performed along that axis.
np.array_equal(a, b): checks whether a, b are arrays of the same values.
np.square(a): perform elementwise square to a.
np.exp(a): perform elementwise exponentials to a.
np.diag(a): make a new array with only the diagonal values of a.

Assignment 1 Part 1: Regression and PCA

For part 1, you are given a csv file containing a dataset. The dataset provided has 506 examples, with each example having 104 features. Your task is to predict the value in the last column "Target" using a ridge regression model.

Here are the steps you should implement for the assignment:

1. Implement the ridge regression model

2. Train a model on the given dataset, and evaluate the model

3. Train three models with different gradient descent methods, and compare their loss graphs: (Full) Batch, Mini-Batch, Stochastic

4. Train four models each with different regularization factors: 10, 1, 0.1, 0. Compare their weight distributions.

5. Use PCA to reduce the number of features in the dataset, and show that reducing the complexity of the models can lead to improved performance

Assignment 1 Part 1: Regression and PCA

[Step 1] Implement the ridge regression model

Objective function:

$$J_W = \frac{1}{2m}\sum_{i=1}^{m}(y_i - h(x_i))^2 + \lambda\sum_{j=1}^{n}{w_j}^2$$

We need to divide this term by the number of examples to stabilize the results.

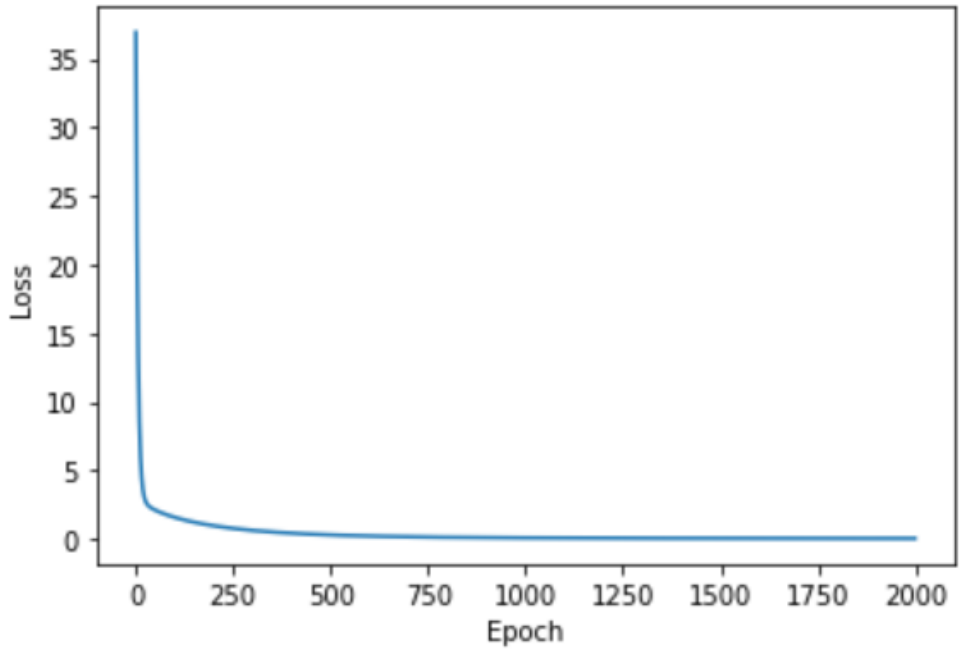
Notations

- m: $the\ number\ of\ examples$
- n: $the\ number\ of\ features$
- W: $weights$
- b: $bias$
- λ: $regularization\ factor$
- $y_W(x_i)$: $predicted\ value\ of\ the\ i-th\ data\ point$

$$y_W(x_i) = \sum_j^n x_{ij} * w_j + b$$

Assignment 1 Part 1: Regression and PCA

[Step 2] Train a model on the given dataset and evaluate the model



Root Mean Square Error (Test): 0.22607383361488978

Assignment 1 Part 1: Regression and PCA

[Step 3] Train three models with different gradient descent methods, and compare their loss graphs

- For the gradient descent algorithm:
  - Calculate gradients and update weights for N training samples
    - If N is equal to 1 → Stochastic Gradient Descent (SGD)
    - If N is equal to m (where m is the number of examples) → Full-batch Gradient Descent
    - If N is equal to k (where $1 < k < m$) → Mini-batch Gradient Descent
      - Has intermediate characteristics between the Full-batch and Stochastic Gradient Descent methods
      - Mini-batch gets the optimal solution faster than full-batch but slower than SGD
      - Mini-batch method requires less memory less than full-batch but more than SGD
      - Mini-batch method avoids falling into a local minimum better than full-batch but worse than SGD
      - Loop {

        for $i = 1$ to m/k {

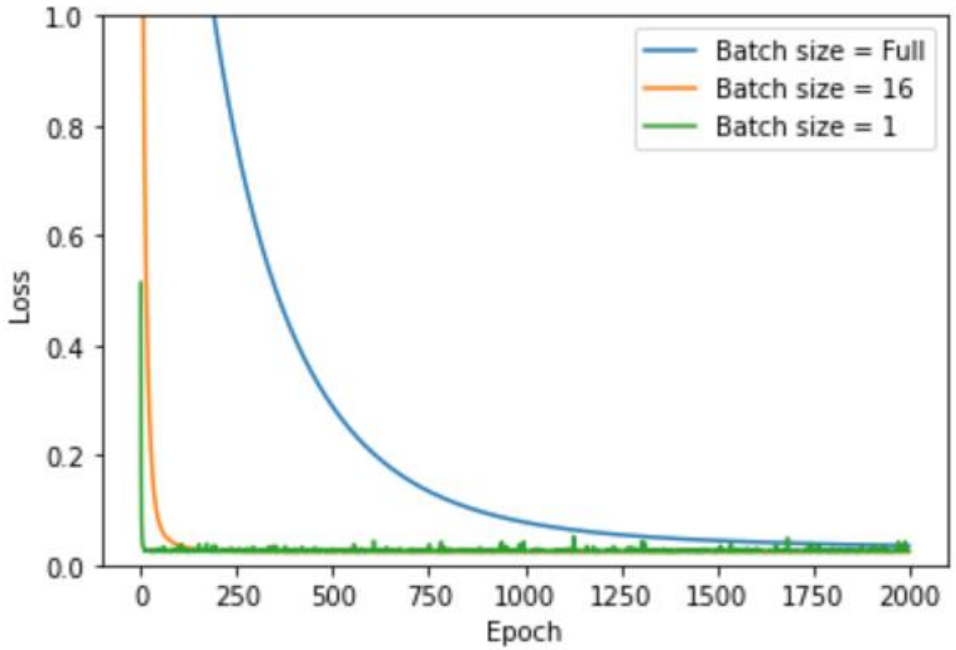
        $$w_j := w_j + \alpha \sum_{i=1}^{k} \left(y^{(i)} - y_w\left(x^{(i)}\right) \right) x_j^{(i)} \ (for\ j = 0, \dots, d, where\ d\ is\ the\ number\ of\ features)$$

        }

        }
  - For details on Full-batch and Stochastic Gradient Descent, refer to Lecture 3 (PPT p.33~p.38)

Assignment 1 Part 1: Regression and PCA

[Step 3] Train three models with different gradient descent methods, and compare their loss graphs

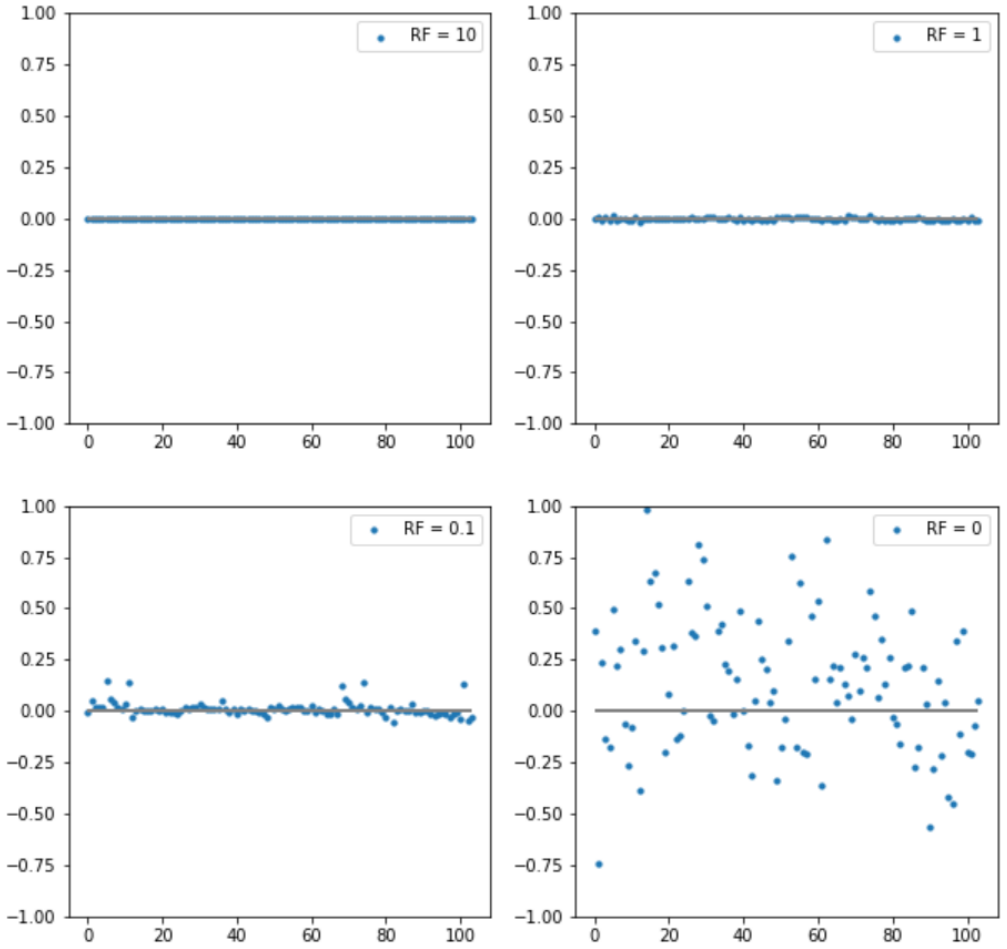


(0.22607383361488978, 0.2147921585242702, 0.22218498088575542)

RMSE (Full-batch) | RMSE (Mini(16)-batch) | RMSE (Stochastic)

Assignment 1 Part 1: Regression and PCA

[Step 4] Train four models with different gradient regularization factors and compare the distributions of weights



RMSE

RF 10 : 0.32785687211659154
RF 1 : 0.2863526682551781
RF 0.1 : 0.22782633648849757
RF 0 (Linear): 0.3036531934278245

Assignment 1 Part 1: Regression and PCA

[Step 5] Use PCA to reduce the number of features in the dataset, and show that reducing the complexity of the models can lead to improved performance

```
Cumulative explained variances
# PCs = 1: 0.55
# PCs = 2: 0.66
# PCs = 3: 0.73
# PCs = 4: 0.8
# PCs = 5: 0.84
# PCs = 6: 0.88
# PCs = 7: 0.9
# PCs = 8: 0.93
# PCs = 9: 0.94
# PCs = 10: 0.95
# PCs = 11: 0.96
# PCs = 12: 0.97
# PCs = 13: 0.97
# PCs = 14: 0.98
# PCs = 15: 0.98
# PCs = 16: 0.98
# PCs = 17: 0.98
# PCs = 18: 0.99
# PCs = 19: 0.99
# PCs = 20: 0.99
# PCs = 21: 0.99
# PCs = 22: 0.99
# PCs = 23: 0.99
# PCs = 24: 0.99
# PCs = 25: 0.99
# PCs = 26: 0.99
# PCs = 27: 0.99
# PCs = 28: 0.99
# PCs = 29: 1.0
# PCs = 30: 1.0
# PCs = 31: 1.0
      ⋮
```

The cumulative explained variance only increases by 0.01 when adding another component.
We therefore set the number of principal components to 8.

Cumulative explained variances indicate how much data information the projection preserves.

$$Cumulative\ explained\ variances\ (i) = \frac{\sum_{j=1}^{i} \lambda_j}{\sum_{j=1}^{n} \lambda_j},$$

$$where\ \lambda_i\ is\ the\ ith\ eigenvalue, and\ n\ is\ the\ number\ of\ features$$

RMSE

```
Ridge :  0.22015771241141036
Linear :  0.18687985377989405
```

Assignment 1 Part 2: Clustering

For part 2 of this assignment, you will be implementing K-means clustering and kernel K-means clustering.
Here are the steps you should implement for the assignment.

1. Implement K-means clustering.
2. Implement kernel K-means clustering.
3. Run K-means clustering on the given example and print the NMI score.
4. Visualize the K-means clustering result.
5. Run kernel K-means clustering on the given example and print the NMI score.
6. Visualize the kernel K-means clustering result.

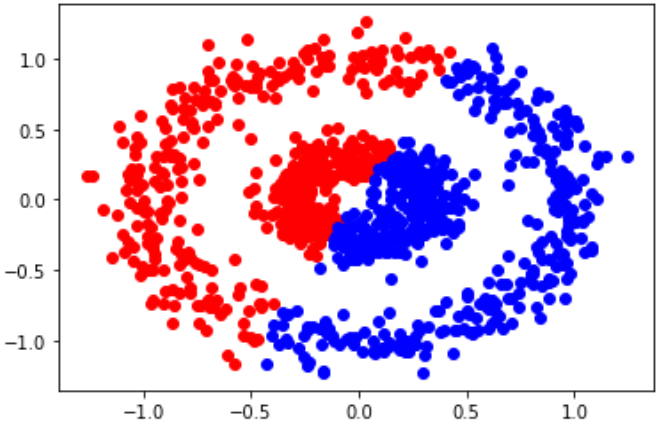
You are provided with skeleton code that will guide you through these steps.

For steps 3 and 5:

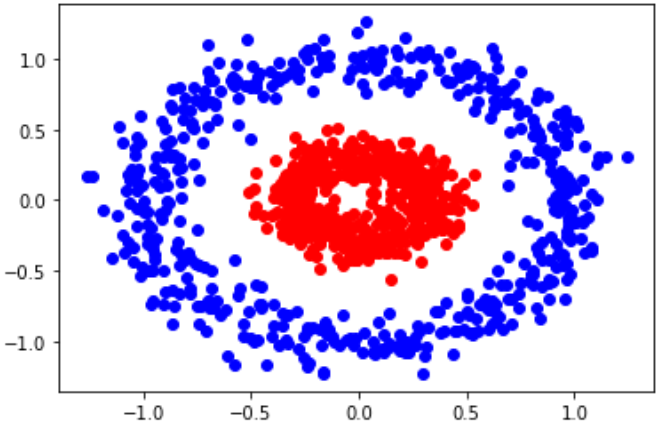
1. Fix the seed value of np.random.
    - You should use the same seed value for both steps.
        - This allows for us to make a comparison between k-means clustering and kernel k-means clustering
2. Run k-means clustering 10 times and record the objective function values.
3. Pick the clustering result with the smallest objective function value.
4. Use that result for the final outputs (NMI scores and visualization).

The NMI score for k-means clustering should be close to 0.
The NMI score for kernel k-means clustering should be close to 1.

K-means clustering



Kernel K-means clustering



Submission

- For part 1, take screenshots that show the completion of each step.
- For part 2, take screenshots of the output of the **last 4 cells (the NMI scores and visualization of each clustering method).**
- **Deadline: Oct. 4th (Monday) PM 11:59. We do not accept late submissions.**
- If you have any questions, please use the KLMS Q&A Board!
- Good luck and have fun!