Statistical Machine Learning
Class Project

In this class project, we will systematically implement and examine the three major categories of machine learning techniques of this course, including supervised learning, un-supervised learning and deep learning.

Part 1: Density Estimation and Classification

Project Overview:

In this part, you need to first perform parameter estimation for a given dataset (which is a subset from the MNIST dataset). The MNIST dataset contains 70,000 images of handwritten digits, divided into 60,000 training images and 10,000 testing images. We use only images for digit “0” and digit “1” in this question.

Therefore, we have the following statistics for the given dataset:

Number of samples in the training set: "0": 5923 ;"1": 6742.
Number of samples in the testing set: "0": 980; "1": 1135

You are required to extract the following two features for each image:

1) The average brightness of the image
2) The average of the variances of each rows of the image.

We assume that these two features are independent, and that each image (represented by a 2-D features vector) is drawn from a 2-D normal distribution.

We also further assume that the prior probabilities are the same (P(Y=0) = P(Y=1) =0.5), although you may have noticed that these two digits have different numbers of samples in both the training and the testing sets.

You may go to the original MNIST dataset (available here http://yann.lecun.com/exdb/mnist/) to extract the images for digit 0 and digit 1, to form the dataset for this project. To ease your effort, we have also extracted the necessary images, and store them in “.mat” files. You may use the following piece of code to read the dataset:

```
import scipy.io
Numpyfile= scipy.io.loadmat(‘matlabfile.mat’)
```

The specific algorithmic tasks you need to perform for this part of the project include:

1) Extracting the features and then estimating the parameters for the 2-D normal distribution for each digit, using the training data. Note: You will have two distributions, one for each digit.
2) Use the estimated distributions for doing Naïve Bayes classification on the testing data. Report the classification accuracy for both “0” and “1” in the testing set.

Algorithms:

MLE Density Estimation, Naïve Bayes classification

Resources:

A subset of MNIST dataset, download either from http://yann.lecun.com/exdb/mnist/ (requiring you to extract data corresponding to digit 0 and digit 1 only), or from the .mat files provided.

Workspace:

Any Python programming environment.

Software:

Python environment.

Language(s):

Python. (MATLAB is equally fine, if you have access to it.)

Required Tasks:

1. Write code to extract features for both training set and testing set.
2. Write code to estimate/compute the parameters of the relevant distributions.
3. Write code to implement the Naïve Bayes Classifier and use it produce a predicted label for each testing sample.
4. Write code to compute the classification accuracy.
5. Submit a short report summarizing the results, including the estimated parameters of the distributions and the final classification accuracy.

Optional Tasks:

1. Repeat the experiments for different pairs of digits.
2. Consider doing multi-class classification.

Optional tasks are to be explored on your own if you are interested and have extra time for them. No submission is required on the optional tasks. No grading will be done even if you

submit any work on the optional tasks. No credit will be assigned to them even if you submit them. (So, please do not submit any work on optional tasks.)

What to Submit and Due Dates:

1. Code:
    - Acceptable file types are .py/.m or .zip.
    - If you have only one file, name the file to be main.py or main.m for matlab users, and submit it.
    - If you have multiple code files, please name the main file as main.py and name other files properly based on its content; Similarly, for matlab users, you should have only one main.m and other relevant .m files. Next, zip all the files and submit Code.zip file.
    - Documentation comment is important and required.
        - Please add comment properly to explain what you do for each task.
        - You do not need to explain every line. But for each task, a brief introduction is required before the code segment. To be specifically, in the comment part, you should explain what you do for the task, what is the input, what is the output, what is the meaning of each variable you use and what is the meaning of the functions you use. If there is no comment and the code is unreadable, no score will be given.
2. Report:
    - Acceptable file types: .pdf or .doc/docx.
    - Length of the report: 2-5 A4 pages.
    - Content: (The following must be included)
        - The formula you used to estimate the parameters, and the estimated values for the parameters.
        - The expression for the estimated normal distributions.
        - Explanation on how the distributions are used in classifying a testing sample(i.e., explaining how you implement the Naïve Bayes Classifier).
        - The final classification accuracy for both “0” and “1” for the testing set.

The code and report are due at the end of Week 3.

Evaluation criteria:

Working code; Correct final results (for both estimated parameters and the classification results).

Code

- 3 points - Correctly extracts features for both training and testing set.
- 3 points - Correctly estimates and computes the parameters of the relevant distributions

- 3 points - Correctly implements the Naive Bayes Classifier and uses it to produce a predicted label for each testing sample
- 1 point - Correctly computes the classification accuracy
- 4 points - Each of these code parts is correctly documented in comments

Report:

- 2 points - Formula for estimating the parameters and their estimated values
- 1 points - Expression for the estimated normal distributions
- 2 points - Explains how the distributions are used in classifying a testing sample
- 1 points - The final classification accuracy for both “0” and “1” for the testing set

Part 2: Unsupervised Learning (K-means)

Project Overview:

In this part, you are required to implement the k-means algorithm and apply your implementation on the given dataset, which contains a set of 2-D points. You are required to implement two different strategies for choosing the initial cluster centers.

> Strategy 1: randomly pick the initial centers from the given samples.
>
> Strategy 2: pick the first center randomly; for the i-th center (i>1), choose a sample (among all possible samples) such that the average distance of this chosen one to all previous (i-1) centers is maximal.

You need to test your implementation on the given data, with the number k of clusters ranging from 2-10. Plot the objective function value vs. the number of clusters k. Under each strategy, plot the objective function twice, each start from a different initialization.

(Referring to the course notes: When clustering the samples into k clusters/sets D_i, with respective center/mean vectors $\mu_1, \mu_2, \ldots \mu_k$, the objective function is defined as $\sum_{i=1}^{k} \sum_{\mathbf{x} \in D_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2$)

Algorithms:

k-Means Clustering

Resources:

A 2-D dataset to be provided.

Workspace:

Any Python programming environment.

Software:

Python environment.

Language(s):

Python. (MATLAB is equally fine, if you have access to it.)

Required Tasks:

1. Write code to implement the k-means algorithm with Strategy 1.
2. Use your code to do clustering on the given data; compute the objective function as a function of k (k = 2, 3, …, 10).
3. Repeat the above step with another initialization.
4. Write code to implement the k-means algorithm with Strategy 2.
5. Use your code to do clustering on the given data; compute the objective function as a function of k (k = 2, 3, …, 10).
6. Repeat the above step with another initialization.
7. Submit a short report summarizing the results, including the plots for the objective function values under different settings described above.

Optional Tasks:

1. Based on the code you developed above, design a way to choose a k that is optimal. Explain in what sense you view your k as optimal.

Optional tasks are to be explored on your own if you are interested and have extra time for them. No submission is required on the optional tasks. No grading will be done even if you submit any work on the optional tasks. No credit will be assigned to them even if you submit them. (So, please do not submit any work on optional tasks.)

What to Submit:

1. Code file with comments explaining what you do for each part as directed
2. A report that summarizes the results and includes the plots for each of the objective function values.

The code and reports are due at the end of Week 5.

Evaluation criteria:

Working code and correct final results:

Code:

- 4 points - Correctly implements the k-Means algorithm with Strategy 1
- 1 point - Code is used to correctly do clustering on the given data; computes the objective function as a function of k (k - 2, 3, …, 10).
- 4 points - Correctly implements the k-Means algorithm with Strategy 2
- 1 point - Code is used to correctly do clustering on the given data; computes the objective function as a function of k (k - 2, 3, …, 10).

- 4 points - Each of these code parts is correctly documented in comments

Report:

- 2 points - Report summarizes results
- 4 points - The plots for the objective function values under each of the settings described in the code criteria

Part 3: Classification Using Neural Networks and Deep Learning

Project Overview:

In this part, we will revisit the Handwritten Digits Recognition task in Part 1, using a convolutional neural network. The basic dataset is the same MNIST dataset from Part I, but you may choose to use only a subset for training and testing, if speed performance with the entire dataset becomes a bottleneck. For example, you may use only 6000 samples for training (each digit with 600 samples) and 1000 samples for testing (each digit with 100 samples).

The basic requirement of this part is to experiment with a convolutional neural network with the following parameter settings:

(1) The input size is the size of the image (28x28).
(2) The first hidden layer is a convolutional layer, with 6 feature maps. The convolution kernels are of 3x3 in size. Use stride 1 for convolution.
(3) The convolutional layer is followed by a max pooling layer. The pooling is 2x2 with stride 1.
(4) After max pooling, the layer is connected to the next convolutional layer, with 16 feature maps. The convolution kernels are of 3x3 in size. Use stride 1 for convolution.
(5) The second convolutional layer is followed by a max pooling layer. The pooling is 2x2 with stride 1.
(6) After max pooling, the layer is fully connected to the next hidden layer with 120 nodes and relu as the activation function.
(7) The fully connected layer is followed by another fully connected layer with 84 nodes and relu as the activation function, then connected to a softmax layer with 10 output nodes (corresponding to the 10 classes).

We will train such a network with the training set and then test it on the testing set.

You are required to plot the training error and the testing error as a function of the learning epochs. You are also required to change some of the hyper-parameters (the kernel size, the number of feature maps, etc), and then repeat the experiment and plot training and testing errors under the new setting.

These are the minimum requirements. Additional requirements may be added (like experimenting with different kernel sizes, number of feature maps, ways of doing pooling, or even introducing drop-out in training, etc.).

Algorithm:

Convolutional Neural Network

Resources:

MNIST dataset, Google CoLab

Workspace:

Google CoLab (see file intro_to_colab.docx for more details)

Software:

Google CoLab

Language(s):

Python

Getting Started:

Read this document carefully, as well as additional files included (intro_to_colab.docx and baseline.docx). For more details about Colab, please go to https://colab.research.google.com/notebooks/welcome.ipynb

Required Tasks:

1. Read intro_to_colab.docx to get familiar with the platform.
2. Run the baseline code (as provided) and report the accuracy.
3. Change the kernel size to 5*5, redo the experiment, plot the learning errors along with the epoch, and report the testing error and accuracy on the test set.
4. Change the number of the feature maps in the first and second convolutional layers, redo the experiment, plot the learning errors along with the epoch, and report the testing error and accuracy on the test set.
5. Submit a brief report summarizing the above results, along with your code.

Optional Tasks:

1. Change the kernel size to 9*9 and redo the experiment and report your results.

2. Use another optimizer (it can be chosen from https://keras.io/optimizers/) and at least 3 different learning rate to redo the experiment, and report the accuracy vs learning rate on the test set.
3. Use average pooling and redo the experiment an0d report your results.

Optional tasks are to be explored on your own if you are interested and have extra time for them. No submission is required on the optional tasks. No grading will be done even if you submit any work on the optional tasks. No credit will be assigned to them even if you submit them. (So, please do not submit any work on optional tasks.)

What to Submit and Due Dates

1. Code: please add comment properly to explain what you do.

2. A report including the plots for the learning/testing errors and the final classification accuracy.

Code and Report are due at the end of Week 7.

Evaluation criteria:

Results (accuracy numbers and learning curves)

- 4 points - Run the baseline code as provided and report the accuracy
- 2 points - Change the kernel size to 5*5, redo the experiment
- 2 points - Plot the learning errors along with the epoch
- 2 points - Report the testing error and accuracy on the test set
- 2 points - Change the number of the feature maps in the first and second convolutional layers, redo the experiment
- 2 points - Plot the learning errors along with the epoch
- 2 points - Report the testing error and accuracy on the test set

Report:

- 3 points - Report summarizes results
- 1 points - The code