Lab11

Lab11

EE 599 Lab11
Spring 2019 Nazarian

100+Extra100 Points

Assigned: Monday April 8th
Due: ~~April 24th, Wedneday~~
I was asked to extend for a few days. The new deadline: Sun. April 28 at 11:59pm. Those who submit by orig. (April 24) will receive 15% extra credit.
Late submissions will be accepted for two days after the deadline with a maximum penalty of 15% per day. For each day, submissions between 12am and 1am: 2% penalty, between 1 and 2am: 4%, 2-3am: 8%, and after 3am: 15%.

Notes:
This assignment is based on individual work. No collaboration is allowed (Discussing the technical details of each problem with other students before submitting your work, copying from students of current or previous semesters is not permitted in our lab and HW assignments). You may post your questions on the discussion forums and/or use the office hours. We may pick some students in random to demonstrate their design and simulations. Please refer to the syllabus and the first lecture regarding the USC and our policies, including the penalties for any violation. If you have any doubts about what is allowed or prohibited in this course, please contact the instructor.

What You Will Practice
In this lab, you will get familiar the standard machine learning pipeline, practice implementing several algorithms from scratch, solving regression and multi-class classification problem using Python3.

Part 0 - Preliminaries

Attached Files: requirements.txt (81 B)

Part 0 Preliminaries

1. Set up a Python 3 virtual environment

Please log in the StudentVM as CS103 Student User to work on this Lab.

Some packages are pre-installed but might be outdated, please use the provided *requirements.txt* to update the packages, by entering the following command in your terminal:

```
$ sudo pip install virtualenv # Install if you didn't
install it
$ virtualenv -p python3 .env # create a virtual
environment
$ source .env/bin/activate # activate the virtual
environment
$ pip install -r requirements.txt # install necessary
packages inside of the virtual environment
# Work on your lab...
$ deactivate
```

2. Get Familiar with the required Python packages

In this lab, we will intensively use functions from Python libraries including **numpy, scipy, pandas, sklearn, matplotlib**. Please read the documentation of each library to understand the usage.

Part 1 - Regression (30%)

Attached Files: candyshop_data.txt (1.327 KB)

Motivation

Regression analysis is a set of useful statistical modeling methods that estimates the relationships among variables. Specifically, in Machine Learning, regression is widely used a as a supervised learning for prediction (e.g. Linear Regression) and classification (e.g. Logistic Regression) problems. In this part, you will implement Linear Regression with one variable to help predict profits for a candy shop.

Problem Description

Suppose you are an intern in the sales and marketing department in a candy and chocolate company. The company is considering different cities for opening a new retail store. The factory has already have stores in various cities such as LA, New York, Boston etc. and you are given a dataset containing profits and populations from those cities. You are asked to train a model based on the collected data to decide which city should be chosen next.

Here are more details about this problem.

(1) Dataset:

The file **candyshop_data.txt** contains the dataset for this linear regression problem. The first column is the population (in 10,000 s) of a city and the second column is the profit (in 10,000 $ s) of a candy shop in that city. A negative value for profit indicates a loss.

(2) Model:

You are told that the model is expected to be linear (affine). You need to augment the given input feature with a dimension containing ones.

Q0. Visualization of Data

Before you start to train a model, it is good to have a rough understanding of the distribution of your data. In this problem, it will be a 2D plot, where x-axis represents the population, while y-axis represents the corresponding profits. You don't need to turn in this plot.

Q1. Linear Least Squares Solution (30%)

The goal of your model is to minimize the cost function *J(w)*:

$$J(w_0, w_1, \dots, w_d) = \frac{1}{2N} \sum_{i=1}^{N} (h_w(x^{(i)}) - y^{(i)})^2$$

where x is the population (input feature), y is the corresponding profit (output value), and N is the number of your pairs of data. The hypothesis function *h(.)* is given as below for this problem:

$$h_w(x) = w_0 + w_1 x_1$$

We can get a closed-form solution for a linear regression problem, as known as the Normal Equation. Suppose we notate the data matrix as X, the closed-form parameter $\underline{w}$ can then be solved by:

$$\underline{w} = (X^T X)^{-1} X^T \underline{y}$$

Your resulting $\underline{w}$ should be a 2x1 vector in this problem setting.

What you need to do:

- Write a Python3 program named "LLSR_<STUDENT_ID>.py" to get the model parameter $\underline{w}$. Report $\underline{w}$ in your readme file.
- Draw the linear fit using your final $\underline{w}$ on your plot of data in Q0.
- Predict on new data: what are the expected profits in the cities of 20,000 and 50,000 population. Report your results.

Part 2 - Regression (Extra Credit 40%)

Q2. Gradient Descent (Extra Credit 40%)

Here, you will learn the linear regression parameter $\underline{w}$ using Gradient Descent.

To get the best $\underline{w}$ that minimizes your cost function, you update each j dimension (j = 0, ..., d) of $\underline{w}$ based on the gradient of the cost function until converge, where η is the learning rate that you define.

$$w_j{}^{(k+1)} = w_j{}^{(k)} - \eta \frac{1}{N} \sum_{i=1}^{N} (h_w(x^{(i)}) - y^{(i)}) x_j{}^{(i)}$$

In this problem, you can initialize $\underline{w}$ as a zero vector. You can try different learning rate η and iteration number. Please remember to augment data x with a dimension of all-ones to get the affine model.

To get full extra credits, you need to do:

- Write a Python3 program named "GD_<STUDENT_ID>.py" to use gradient descent to solve this linear regression problem.
- Monitor the cost: During your implementation, record the cost *J(w)* in each iteration to monitor whether it is converging or not. Plot the curve of cost with respect to the iteration index.
- Report your final $\underline{w}$.
- Discuss about the effects of different choices of the hyperparameters in GD.
- Discuss about the pros and cons for GD and least-square solution

Part 3 - Image Classification with PCA and KNN (70%)

Part 3 Image Classification with PCA and KNN (70%)

1. Dataset: CIFAR-10

The CIFAR-10 dataset has labeled tiny images (32x32 each) with 10 classes. There are in total 50000 images for training and 10000 images for testing. It is a widely-used dataset for benchmarking image classification models.

- Download the python version of CIFAR-10 dataset from the link below: https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
- Look into the description on the web page(https://www.cs.toronto.edu/~kriz/cifar.html) to figure out how to extract the data from the downloaded pickle files.
- Extract the first 1000 images from the data batch 1 for the following problems.
- Split the images into training and testing sets. The first N images are used as testing images which are queries for KNN classifier. The rest

of (1000-N) images are used for training. (N is specified as an input argument.)

2. Preprocessing and PCA (20%)

Instead of classifying images in the pixel domain, we usually first project them into a
feature space since raw input data is often too large, noisy and redundant for analysis. Here is where dimensionality reduction techniques come into play. Dimensionality reduction is the process of reducing the number of dimensions of each data point while preserving as much essential information as possible. PCA is one of the main techniques of dimensionality reduction. It performs a linear mapping of the data to a lower-dimensional space in such a way that the variance of the data in the lower-dimensional representation is maximized. In this problem, the objective is to use the sklearn PCA package to perform dimensionality reduction on images extracted from the CIFAR-10 dataset.

- Convert the RGB images to grayscale with the following formula (you should do it manually without using any package):

  $$Y = 0.299R + 0.587G + 0.114B$$

- Read the documentation of the PCA package provided by sklearn (http://sklearn.org/stable/modules/generated/sklearn.decomposition.PCA.html Compute the PCA transformation by using only the training set with the sklearn PCA package. Perform dimensionality reduction on both training and testing sets to reduce the dimension of data from 1024 to D. (D is specified as an input argument.)
- Note that you should specify a full SVD solver to build the PCA embeddings (i.e. svd solver="full"). (The results may not be consistent if the randomized truncated SVD solver is used in the sklearn PCA package.

3. K-Nearest Neighbors (KNN) Classifier (50%)

K-nearest neighbors algorithm (KNN) is a nonparametric method used for classification. A query object is classified by a majority vote of the K closest training examples (i.e. its neighbors) in the feature space. In this problem, the objective is to implement a KNN classifier to perform image classification given the image features obtained by PCA.

- Implement a K-Nearest Neighbors classifier to predict the class labels of testing images. Make sure you use the inverse of Euclidean distance as the metric for the voting. In other words, each neighbor n_i, is represented as a vector, where i = 1, ..., K, contributes to the voting with the weight of $1/\left\|x-n_i\right\|_2$, where x is a queried vector. (K is specified as an input argument.)

- For this part, you are NOT allowed to use any library providing KNN classifier (e.g. sklearn).

4. Verify Your implementation with Sklearn

Try using KNN implemented in sklearn package (https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html) to predict the class labels of testing images.

5. Program Description

a. Script

Write a Python3 program named “knn_<STUDENT_ID>.py" to solve the aforementioned problems. Your program should use sys.argv[] for taking four argument inputs including K, N, D, and PATH_TO_DATA. We will evaluate your code with the following command:

```
$ python knn_<STUDENT_ID>.py K D N PATH_TO_DATA
```

b. Output

Your program should output two text files named “knn_results.txt" and “knn_results_sklearn.txt" after execution. The output file “knn_results.txt" should contain the KNN results of N testing images in order. Each line of the output should contain a predicted label and a ground truth label, separated by a single space (i.e. i-th line contains the predicted label and the ground truth label of the i-th testing image). The output file “knn_results_sklearn.txt" should contain the KNN results by using the function from sklearn package, following the same format as “knn_results.txt". **Ideally, the content of two output text files should be identical.**

c. Sample command

```
$ python knn_<STUDENT_ID>.py 5 100 10 ./cifar-10-batches-py/data_batch_1
```

d. Sample output (sample_knn_results.txt)

```
3 6
8 9
9 9
6 4
0 1
```

1 1

2 2

0 7

6 8

0 3

Part 4 - Seeds Classification with Gaussian Naive Bayes (Extra Credit 60%)

1. Dataset Manipulation (6%)

We will work with the UCI Seeds dataset, which uses measurements of seven geometric parameters of wheat kernels to classify three different types of wheat seeds. The detailed description of the dataset can be found at https://archive.ics.uci.edu/ml/datasets/seeds. The full dataset comprises of 270 samples, you need to randomly shuffle the samples and split the dataset as a training set and test set. The training set should include 85% of the samples, while the rest 15% are test samples.

2. Implementing Gaussian Naive Bayes from Scratch (36%)

Gaussian Naive Bayes is an simple and effective probabilistic model for classification when the attributes are of continuous values.

The naive Bayes model defines the joint distribution:

$p(\mathbf{X}, Y) = p(\mathbf{X}|Y)p(Y)$

In the training process of a Bayes classification problem, we need to do the following with the sample training data:

- Estimate log-likelihood distributions of X for each value of Y:
  $\log p(\mathbf{X} = \mathbf{x}|Y = y)$
  - X is a multi-dimensional feature vector, in Naive Bayes, we assume the features are conditional independent, thus
  $$\log p(\mathbf{X}|Y) = \sum_{i=1}^{n} \log p(X_i|Y)$$
  - In Gaussian Naive Bayes, we additionally assume each conditional distribution follows Gaussian Distribution, thus
  $$p(X_i = x_i|Y = y) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp -\frac{(x_i - \mu_i)^2}{\sigma^2}$$

  - Use MLE to estimate the model parameters, i.e. calculating the conditional mean μ_i and variance σ_i^2 for each feature.

- Estimate prior distribution $p(Y)$

In the testing process, we calculate the log-likelihood $\log p(\mathbf{X} = \mathbf{x}, Y = y)$ for each class y, and assign the class that has the maximum probability as our prediction.

$$\hat{y} = \underset{y}{\operatorname{argmax}} \log p(\mathbf{X} = \mathbf{x}, Y = y)$$

Report the training accuracy, testing accuracy, training time and testing time in seconds in your readme file.

3. Comparison with the Implementation in Sklearn (12%)

Now you are required to compare your implementation with the Gaussian Naive Bayes implemented in Sklearn (`sklearn.naive_bayes.GaussianNB`). Report the training accuracy, testing accuracy, training time and testing time in seconds in your readme file.

4. Playing with Other Classifiers (6%)

Try applying other classifiers implemented in Sklearn, such as SVM, Decision Trees, Random Forests, Multi-layer Perceptron (MLP) classifier to seeds dataset. Report the hyperparameters you used, training accuracy, testing accuracy, training time and testing time in seconds for each of the method in your readme file.

5. Program Description

a. Script

Write a Python3 program named “naive_bayes_<STUDENT_ID>.py" to solve the aforementioned problems. We will evaluate your code with the following command:

```
$ python naive_bayes_<STUDENT_ID>.py
```

b. Output

Your program should not generate any output file. Instead, you should print the accuracy and time information in the console with the following format:

```
My Naive Bayes:
Training acc: XX.XX% Training time: XXXX s
Testing acc: XX.XX% Testing time: XXXX s
Sklearn Naive Bayes:
Training acc: XX.XX% Training time: XXXX s
Testing acc: XX.XX% Testing time: XXXX s
```

```
<Other Classifier>:
Training acc: XX.XX% Training time: XXXX s
Testing acc: XX.XX% Testing time: XXXX s
......
```

Note that printing other unnecessary information in the console will result in losing points. Also, you should report the generated results in your readme file.

Notes and FAQs

Notes

- Please make sure your program does not download the dataset during execution.
- Python packages other than **numpy, scipy, pandas, sklearn, matplotlib** are not required to finish your lab. If you additionally use other packages, please clearly state the package name and version in your readme file.

Submission

Submission

Github Link to accept the assignment invitation:
https://classroom.github.com/a/N1ChyTut
Please push all .py files you wrote into the GitHub repo.
Don't forget to submit a readme file:

- Your readme file should be named as readme_<STUDENT_ID>.pdf, please replace <STUDENT_ID> with your own 10-digit USC-ID. All the four parts should go in one single readme file.
- Any non-working part should be clearly stated.
- The citations should be done carefully and clearly, e.g.: "to write my code, lines 27 to 65, I used the Djkstra's shortest path algorithm C++ code from the following website: www.SampleWebsite.com/..."

You'll lose points if you:

- don't submit a readme file
- don't name your files properly
- submit a .zip file (you need to submit individual files instead)

Discussion Slides