COGS 260, Spring 2018: Assignment 3

May 4, 2018

Due: May 13, 2018 11:59 PM PDT
Late policy: Every 10% of the total received points will be deducted for every extra day past due.
Bonus policy: A maximum of 10 points of bonus can be awarded.

Instructions

1. Reference materials and some useful codes can be found at the end of the document.
2. Submit the ipython notebooks you used for this project in pdf format.
3. Write the rest of the report in pdf including: a) abstract, b) method, c) experiment, d) discussion, and e) references. You can follow leading conferences like CVPR (`http://cvpr2017.thecvf.com/submission/main_conference/author_guidelines`), NIPS (`https://papers.nips.cc/`), or ICML (`http://icml.cc/2016/?page_id=151`). All of those formats are readily available through `www.sharelatex.com`. For the homework assignments, the report can be brief and you can try to summarize your experiments, results, and main findings in a concise way.
4. Submit any ipython notebooks in pdf separately and Zip and upload related code and ipython notebooks as part of your submission on TritonEd in order for the TA to reproduce your result.
5. You are supposed to provide concise quantitative details and analysis of experiments whenever asked to report the details of the experiment. Please DO NOT write long paragraphs. You are encouraged to use plots and images to explain your results. Please report the random seed you used for your experiments. You are encouraged to print out your network structure in the ipython notebook if you choose to use Keras. There is no need to include that graph in your final latex report.

In this assignment, first you are going to review least square estimation and robust analysis with L_1 and L_2 losses, then you are going to work on three generations of neural network: Perceptron learning, Feed-forward Networks and Convolutional Networks.

1 Least Square Estimation [15 pts]

We are given the data $S = \{(x_i, y_i), i = 1, \dots, n\}$. We express the data as matrices $X = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n]^\top$ and $Y = [y_1, y_2, \dots, y_n]^\top$ where $\mathbf{x}_i$ is a feature vector corresponding to the data x_i: $\mathbf{x}_i$ is either $[1, x_i]^\top$ or $[1, x_i, x_i^2]^\top$ in this problem. We wish to solve for W that minimizes the sum-of-squares error function $g(W)$.

$$g(W) = \|X \cdot W - Y\|_2^2 = (X \cdot W - Y)^\top (X \cdot W - Y). \tag{1}$$

(a) Compute the gradient of $g(W)$ with respect to W.
(b) By setting the answer of part (a) to $\mathbf{0}$, prove the following:

$$W^* = \arg\min_W g(W) = (X^\top X)^{-1} X^\top Y. \tag{2}$$

(c) Download the file `q1-least-square.npy` from the course website. Then, complete the following procedures, paste your source code and show the result in your report:

1. Import packages and load data

```
import numpy as np
import matplotlib.pyplot as plt
X_and_Y = np.load('./q1-least-square.npy')
X = X_and_Y[:, 0]  # Shape: (300,)
Y = X_and_Y[:, 1]  # Shape: (300,)
```

2. Plot the scatter graph of data.

```
plt.scatter(X, Y)
plt.xlabel('X')
plt.ylabel('Y')
plt.show()
```

3. Compute the least square line over the given data.

```
# Assume Y = w0 + w1*X = (w0, w1).(1, X) = W.X1
# X1 contains 1 and X.
X1 = np.matrix(np.hstack((np.ones((len(X),1)),
                          X.reshape(-1,1))))
W = X1.T.dot(X1).I.dot(X1.T).dot(Y)
w0, w1 = np.array(W).reshape(-1)
print('Y = {:.2f} + {:.2f}*X'.format(w0, w1))
```

4. Plot the scatter graph of data and estimated line.

```
X_line = np.linspace(0,10,300)
Y_line = w0 + w1 * X_line
plt.scatter(X, Y)
plt.plot(X_line, Y_line, color='orange')
plt.xlabel('X')
plt.ylabel('Y')
plt.show()
```

5. Compute the least square parabola over the given data.
   (please **fill the blank** for `X2` and `W`)

```
# Assume Y = w0 + w1*X + w2*X^2 = (w0, w1, w2).(1, X, X^2) = W.X2
# X2 contains 1, X and X^2.
X2 = # Fill the blank here.
W =  # Fill the blank here.
w0, w1, w2 = np.array(W).reshape(-1)
print('Y = {:.2f} + {:.2f}*X + {:.2f}*X^2'.format(w0, w1, w2))
```

6. Plot the scatter graph of data and estimated parabola.

```
X_line = np.linspace(0,10,300)
Y_line = w0 + w1 * X_line + w2 * (X_line**2)
plt.scatter(X, Y)
plt.plot(X_line, Y_line, color='orange')
plt.xlabel('X')
plt.ylabel('Y')
plt.show()
```

7. Copy and paste the figures in your report.

2 Parabola Estimation [15 pts]

Similar to previous question, you will estimate a parabola function. As before, you are given the data $S = \{(x_i, y_i), i = 1, \ldots, n\}$. The data are expressed as matrices $X = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]^\top$ and $Y = [y_1, y_2, \ldots, y_n]^\top$ where $\mathbf{x}_i$ is $[1, x_i, x_i^2]^\top$. The parabola function is defined as: $f(x; W) = w_0 + w_1 x + w_2 x^2 = \mathbf{x}^\top W$ where $\mathbf{x}$ is $[1, x, x^2]^\top$ and W is $[w_0, w_1, w_2]^\top$. Please download `q2-parabola.npy` from website as data source.

(a) Consider L_2 norm as your loss function:

$$g(W) = \|XW - Y\|_2^2 = \sum_{i=1}^{n} \left(y_i - f(x_i; W)\right)^2$$

please use the **closed form solution** to compute W and plot the scatter graph of data and estimated parabola. Report the parabola function and the figures.

(b) Consider L_1 norm as your loss function:

$$g(W) = \|XW - Y\|_1 = \sum_{i=1}^{n} |y_i - f(x_i; W)|$$

Derive the gradient:

$$\frac{\partial g(W)}{\partial W} = \left(\left(\operatorname{sign}(XW - Y)\right)^\top X\right)^\top$$

where

$$\operatorname{sign}(x) = \begin{cases} 1, & x > 0 \\ 0, & x = 0 \\ -1, & x < 0. \end{cases}$$

and if X is a matrix, $\operatorname{sign}(X)$ means performing element-wise $\operatorname{sign}(x_{ij})$ over all element x_{ij} in X.

Use the **gradient descent method** to compute W and plot the scatter graph of data and estimated parabola. Report the parabola function and the figures.

Hint 1: In NumPy, you can use `np.sign(x)` to compute the sign of matrix `x`.

Hint 2: You may need to change the number of iterations to 300000, learning rate to 0.000001 and error threshold for W to 0.00001.

3 Perceptron Learning [30 pts]

We apply perceptron learning algorithm to learn a linear classifier. In Perceptron Learning Algorithm, the activation rule is defined as:

$$f(\mathbf{x}) = \begin{cases} 1, & \text{if } \mathbf{w}^T\mathbf{x} + b \geq 0 \\ -1, & \text{otherwise} \end{cases}$$

and the optimal solution of $\mathbf{w}^*$ is $\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_{i=1}^{n} \mathbf{1}(y_i \neq \text{sign}(\mathbf{w}^T\mathbf{x} + b))$, where $\mathbf{1}(\cdot)$ is the indicate function, and $\mathbf{sign}(\cdot)$ is the sign function.

Dataset Description

The data is in iris.zip under resources on piazza. In this problem, we will attempt to make a flower classifier. We shall use a modified version of the popular UCI Iris dataset provided in the file within the iris folder. "iris_train.data" and "iris_test.data" are csv files. They can be read using code provided in example.ipynb. Each file describes a data set with the following characteristics:

- Number of Attributes: 4 numeric attributes and the class label.
- Attribute Information: sepal length in cm, sepal width in cm, petal length in cm, petal width in cm
- The last column is the class label (Iris-setosa, Iris-versicolor)

In data file, each line stands for an instance. The examples are shown below:

sepal length/cm	sepal width/cm	petal length/cm	petal width/cm	plant type
5.0	3.3	1.4	0.2	*Iris-setosa*
7.0	3.2	4.7	1.4	*Iris-versicolor*

Pseudocode for the Perceptron Learning Algorithm

We provide a piece of pseucode for the perceptron learning algorithm, as is shown in Algorithm 1.

Algorithm 1 Perceptron Learning Algorithm

```
Data: training data points $\mathbf{X}$, and training labels $\mathbf{Y}$;
Randomly initialize parameters $\mathbf{w}$ and $b$; pick a constant $\lambda \in (0, 1]$, which is similar to the step size
  in the standard gradient descent algorithm (by default, you can set $\lambda = 1$).
while not every data point is correctly classified do
    randomly select a data point $\mathbf{x}_i$ and its label $y_i$;
    compute the model prediction $f(\mathbf{x}_i)$ for $\mathbf{x}_i$;
    if $y_i == f(\mathbf{x}_i)$ then
        continue;
    else
        update the parameters $\mathbf{w}$ and $\mathbf{b}$:
          $\mathbf{w}_{t+1} = \mathbf{w}_t + \lambda(y_i - f(\mathbf{x}_i))\mathbf{x}_i$
          $\mathbf{b}_{t+1} = \mathbf{b}_t + \lambda(y_i - f(\mathbf{x}_i))$
    end
end
```

3.1 Dataset [5 pts]

Using the **pairplot** function in **seaborn** library on the raw panda data structure to plot each of the six 2 dimensional feature spaces (i.e. sepal length vs. sepal width, sepal width vs. petal width, etc.) using a different symbol for each class. Put these graphs in your report. Are the classes linearly separable in each of the feature spaces?

3.2 Programming [15 pts]

Implement your own single layer perceptron learning algorithm, and train a linear classifier on the training set. Plot the error rate on the training set during training. Report the learning rate used and the number of iterations for convergence. **HINT:** You might need to set the number of iterations to be large enough.

3.3 Decision Boundary [5 pts]

Derive the decision boundary of your trained classifier.

3.4 Test [5 pts]

Classify the data in the test set and report these error metrics shown below:

1. Accuracy (The percentage of correctly classified data points in the test set).

   (For the following 3 error metrics, you can think of one of the classes as positive, and the other on as negative.)

2. Precision ($= \frac{\sum \text{True positive}}{\sum \text{Test outcome positive}}$)
3. Recall ($= \frac{\sum \text{True positive}}{\sum \text{Condition positive}}$)
4. F-value ($= \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$)

3.5 [bonus +4]

Z-score each attribute (feature) of data. Make sure that you are not using test set in any way for computing the parameters for Z-scoring, i.e. use only the training set parameters to normalize both training and test set. Now repeat part 3.1, 3.3 on the transformed data. What did you observe?

4 Feed Forward Neural Network [20 pts]

In this problem, you are going to train a feed forward neural network to classify MNIST dataset [2]. Represent each image as a 784-dimensional feature vector. Randomly split the dataset into training and test sets with sizes: 50000 and 10000 respectively. Your tasks are:

4.1 [10 pts]

Train a feed forward network with 1 hidden layer and write the weight update rules in vector notation (matrix notation). Report the training and test error with number of iterations. Also report the network architecture and the hyperparameters (learning rate) used.

4.2 [10 pts]

Repeat the above part for feed forward network with 2 hidden layers. Comment on the performance.

4.3 [bonus +4]

Retrain the network in part 2.(a) with regularization and momentum and report the results. Did you observe any change in the performance?

5 Convolutional Neural Network [20 pts]

For this question you should use tools like Caffe, Theano, Tensorflow, Torch, etc [6].

In Assignment 2, you have played with basic convnets and probably some techniques to accelerate the learning. In this problem, you are going to start from where you left in the previous assignment. You have to train a Convolutional Neural Network on CIFAR-10 dataset [1] with at least 2 convolutional, 2 pooling and one fully connected layer. You can build on top of different network architectures (LeNet[11], AlexNet[7, 8], VGGNet[9], ResNet[10])You have to compare the following techniques to speed-up the learning. For each of the methods, comment on the iterations for convergence, accuracy on the test set, plot of training and test loss with respect to the iterations. Also, report the network architecture. Please refer to [5] for more details about the following methods.

1. Stochastic Gradient Descent [4]
2. Batch Normalization
3. Replace the fully connected layer by average pooling layer
4. Adaptive Gradient [bonus + 2] [3]
5. Nesterov's Accelerated Gradient [bonus + 2] [4]
6. RMSprop [bonus +2] [3]

References

[1] The CIFAR-10 Dataset, `https://www.cs.toronto.edu/~kriz/cifar.html`

[2] The MNIST Database of handwritten digits, `http://yann.lecun.com/exdb/mnist/`

[3] AdaGrad and RMSprop, `http://cs231n.github.io/neural-networks-3/#ada`

[4] Stochastic Gradient Descent and Momentum, `http://cs231n.github.io/neural-networks-3/#sgd`

[5] Neural Networks, Andrej Karpathy, http://cs231n.github.io/neural-networks-3/

[6] Deep Learning Software Links, http://deeplearning.net/software_links/

[7] Tutorial on Keras AlexNet, http://cswithjames.com/keras-9-1-buildtrain-alexnet-cifar10-dataset-keras/

[8] Git Repo for Keras AlexNet, https://github.com/jkh911208/cswithjames/blob/master/8_CIFAR10_alexnet.py

[9] Keras VGG16, https://github.com/keras-team/keras/blob/master/keras/applications/vgg16.py

[10] Keras cifar10 resnet Links, https://github.com/keras-team/keras/blob/master/examples/cifar10_resnet.py

[11] Keras examples, https://github.com/keras-team/keras/tree/master/examples