CS373: Homework 4

Due Date: 11:59 PM, April 17, 2019

Instructions for submission: In this programming assignment you will implement Batch Gradient Descent and Stochastic Gradient Algorithm. You will use two loss functions viz. 'log' and 'hinge'. You will also perform regularization and compare the effects it has on learning and generalization. Evaluation will be on the Review Polarity Dataset. Instructions below detail how to turn in your code and assignment on `data.cs.purdue.edu`

You are given a skeleton code with an existing folder structure. Do not modify the folder structure. You may add any extra files you want to add in the `cs373-hw4/src/` folder. For all other parts write your answers and place plots in a single PDF. Name this `report.pdf`. Place your report in the `cs373-hw4/report/` folder. Label all the plots with the question number. Your homework must contain your name and Purdue ID at the start of the file. If you omit your name or the question numbers for the plots, you will be penalized.

To submit your assignment, log into `data.cs.purdue.edu` (physically go to the lab or use ssh remotely) and follow these steps:

1. Place all of your code in the `cs373-hw4/src/` folder and your report in `cs373-hw4/report/` folder.
2. Change directory to outside of `cs373-hw4/` folder (run `cd ..` from inside `cs373-hw4/` folder)
3. Execute the following command to turnin your code: `turnin -c cs373 -p hw4 cs373-hw4`
4. To overwrite an old submission, simply execute this command again.
5. To verify the contents of your submission, execute this command: `turnin -v -c cs373 -p hw4`. Do not forget the `-v` option, else your submission will be overwritten with an empty submission.

1 Specification

1.1 Dataset

In this homework, you will be working with the same 'Review Polarity' dataset that was introduced for the previous homework. You are given movie reviews and the task is to classify them as `positive` or `negative`. The entire dataset has 1000 movie reviews, out of which 500 are labeled `positive` and the other 500 are labeled as `negative`. The dataset is split into training (`train.tsv`) and test (`test.tsv`) sets. Training set consists of 699 instances and the test set has 301 instances. Look into the training set file and try to gauge the nature of the data and the task.

Your submission will be tested on a hidden dataset which is different from given dataset. The hidden set would have similar number of instances and data splits.

Input: Movie review consisting of a few sentences of text.
Label: Positive (+1)/Negative (−1)

The features you will use for the classification task are bag-of-words. Provided skeleton code has 2 feature extractors already implemented. The first feature extractor extracts binary word presence/absence and the second feature extractor extracts word frequency. You are free to implement and add any features that you believe would help you get a better performance on the task, but you can get full credit without using any extra features. Top-10% submissions according to the performance on the hidden test set will be given extra credit. You might need to use additional features to get extra credit. You are allowed to use libraries such

as `NLTK` or `Spacy` to extract those features.

1.2 Skeleton Code

You are provided a skeleton code with this homework. The code has the following folder structure:

```
cs373-hw4/
├── handout.pdf
├── data/
│   └── given/
│       ├── train.tsv
│       └── test.tsv
├── src/
│   ├── __init__.py
│   ├── main_sgd.py
│   ├── main_bgd.py
│   ├── classifier.py
│   ├── utils.py
│   ├── config.py
│   ├── bgd.py
│   └── sgd.py
└── report/
    └── report.pdf
```

Do not modify the given folder structure. You should not need to, but you may, add any extra files of code in `cs373-hw4/src/` folder. You should place your report (`report.pdf`) in the `cs373-hw4/report/` folder. You **must not** modify `__init__.py`, `main_sgd.py`, `main_bgd.py` or `classifier.py`. They may be replaced while grading. All your coding work for this homework must be done inside `cs373-hw4/src/` folder. You are not allowed to use any external libraries for classifier implementations such as `scikit-learn` etc. Make sure that you understand the given code fully, before you start coding.

Your code will be tested using the command `python3 main_bgd.py` and `python3 main_sgd.py` from inside the `cs373-hw4/src/` folder for SGD and BGD evaluation. Make sure that you test your code on `data.cs.purdue.edu` before submitting.

1.3 Log Loss

The log loss function is defined as,

$$L(x_i, y_i; w) = -\sum_i y_i log(g(z_i)) + (1 - y_i) log(1 - g(z_i))$$

$$g(z_i) = 1/(1 + e^{-z_i}), \quad z_i = w^T x_i + b$$

Note that here y can take 1 or 0

1.4 Hinge Loss

The hinge loss function is defined as,

$$L(x_i, y_i; w) = max(0, 1 - y_i f(x_i))$$

$$f(x_i) = w^T x_i + b$$

Note that here y can take +1 or -1

2 Theory (30 points)

1. Briefly explain the difference between batch gradient descent and stochastic gradient descent. When would you prefer one over the other? **(5 points)**
2. In gradient descent, you need to stop training when the model converges. How will you know that your model has converged? State the stopping criteria and apply the criteria throughout your implementation. **(4 points)**
3. What will be the effect of bias term in BGD/SGD learning? **(2 points)**
4. True/False. Stochastic gradient descent performs less computation per update than batch gradient descent. Give brief reasoning? **(2 points)**.
5. Why do we randomly shuffle the training examples before using SGD optimization? **(2 points)**
6. Hinge Loss which was introduced in class is not differentiable everywhere unlike log loss. Then show how we can use hinge loss function in the gradient descent optimization algorithm. What will be the gradient of hinge loss function without regularization term? **(5 points)**
7. What will be the gradient of the log and hinge loss function if you add the L_2 regularization term $(1/2)\lambda||w^2||$? **(5 points)**
8. Why is regularization used? What's a potential issue with L_2 regularization, if the λ hyperparameter can take negative values? **(5 points)**

3 Batch Gradient Descent(35 points)

3.1 Algorithm (5 points)

Write down the batch gradient descent algorithm for **log loss** in appropriate algorithmic format.

3.2 Implementation (20 points)

- Implement batch gradient descent algorithm for 'log' and 'hinge' loss functions.
- Apply L_2 regularization for both loss functions. Note that a hyperparameter λ is used which controls the amount of regularization. Also, tune the value of λ when you are using regularization.
- Both implementations will have the 'bias term'.

This part could be completed by editing only `bgd.py`, unless you plan to extract any new features for the task. You need to initialize the parameters (`__init__()` method), learn them from training data (`fit()` method) and use the learned parameters to classify new instances (`predict()` method) for each of the loss functions. Take note that `__init__.py`, `main.py` and `classifier.py` may be replaced while grading. Do not make any modifications to these files. Report the results obtained on the given test set for both the loss functions in your report. You should submit your code with the hyperparameter settings (`config.py`) that produce the best performance.

Output format is given below. Your final numbers might differ. **The given numbers are just a guideline, meant for sanity checking**.

```
$ python3 main_bgd.py

BGD Log Loss (No Regularization) Results:
Accuracy: 74.09, Precision: 80.17, Recall: 62.83, F1: 70.45

BGD Log Loss (With Regularization) Results:
```

```
Accuracy: 76.41, Precision: 80.31, Recall: 68.91, F1: 74.18

BGD Hinge Loss (No Regularization) Results:
Accuracy: 77.41, Precision: 79.84, Recall: 72.29, F1: 75.88

BGD Hinge Loss (With Regularization) Results:
Accuracy: 78.07, Precision: 79.70, Recall: 74.32, F1: 76.92
```

3.3 BGD Analysis (10 points)

You need to perform additional experiments to answer the following questions. You don't need to submit your code for this part. You only need to include your plots and discussions in your report. Make sure that the code you submit doesn't include any changes you don't want to be included, as that might affect your chances of getting extra credit.

You need to record the training and test accuracy after every weight update. Let's define this as one epoch.

1. Plot training and test accuracy vs epoch without regularization in a graph. We will have one graph for each type of loss function. A sample graph attached in the end of this handout. For this question 2 graphs are expected. **(5 points)**
2. Run the same experiments with regularization and plot the training and test accuracy for each epoch again for each type of loss function. For this question 2 graphs are expected. **(5 points)**

4 Stochastic Gradient Descent Implementation (35 points)

4.1 Algorithm (5 points)

Write down the stochastic gradient descent algorithm for **hinge loss** in appropriate algorithmic format.

4.2 Implementation (20 points)

- Implement stochastic gradient descent algorithm for 'log' and 'hinge' loss functions.
- Apply L_2 regularization for both loss functions. Note that a hyperparameter λ is used which controls the amount of regularization. Also, tune the value of λ when you are using regularization.
- Both implementations will have the 'bias term'.

This part could be completed by editing only `sgd.py`, unless you plan to extract any new features for the task. You need to initialize the parameters (`__init__()` method), learn them from training data (`fit()` method) and use the learned parameters to classify new instances (`predict()` method) for each of the loss functions. Take note that `__init__.py`, `main.py` and `classifier.py` may be replaced while grading. Do not make any modifications to these files. Report the results obtained on the given test set for both the loss functions in your report. You should submit your code with the hyperparameter settings (`config.py`) that produce the best performance.

Output format is given below. Your final numbers might differ. **The given numbers are just a guideline, meant for sanity checking**.

```
$ python3 main_sgd.py

SGD Log Loss (No Regularization) Results:
Accuracy: 75.42, Precision: 80.83, Recall: 65.54, F1: 72.38
```

```
SGD Log Loss (With Regularization) Results:
Accuracy: 76.41, Precision: 80.31, Recall: 68.91, F1: 74.18

SGD Hinge Loss (No Regularization) Results:
Accuracy: 77.41, Precision: 79.84, Recall: 72.29, F1: 75.88

SGD Hinge Loss (With Regularization) Results:
Accuracy: 78.07, Precision: 79.70, Recall: 74.32, F1: 76.92
```

4.3 SGD Analysis (10 points)

You need to perform additional experiments to answer the following questions. You don't need to submit your code for this part. You only need to include your plots and discussions in your report. Make sure that the code you submit doesn't include any changes you don't want to be included, as that might affect your chances of getting extra credit.

You need to record the training and test accuracy after every weight update. Let's define this as one epoch.

1. Plot training and test accuracy vs epoch without regularization in a graph. We will have one graph for each type of loss function. A sample graph attached in the end of this handout. For this question 2 graphs are expected. **(5 points)**
2. Run the same experiments with regularization and plot the training and test accuracy for each epoch again for each type of loss function. For this question 2 graphs are expected. **(5 points)**

5 Time Limit

Your code must terminate with in $5 - 10$ minutes for all models combined together. If it doesn't terminate with in $5 - 10$ minutes, you will be graded for the output your code generates at the end of the 10 minute time limit.

6 Important Notes

- For all hyper-parameter tuning Accuracy will be the performance measure in this assignment.
- Place all the graphs in your report.
- Initialize all the weights randomly.
- Use gnuplot / matplotlib for generating the graphs.
- Your graphs should look like similar to Figure 1

7 Extra Credit (10 points)

7.1 Top - 10% on Leaderboard (5 points)

Your submission will be evaluated on a hidden dataset of similar nature. The hidden dataset is of similar size and splits. Top-10% (17 in number) of the students in the class would be awarded 5 extra credit points. You are not allowed to use any optimization libraries but you are allowed to use feature extraction software. If in doubt, ask a question on piazza and the instructors would respond. Remember that the extra credit only depends on the results on the hidden dataset. Overfitting the given dataset might prove counter-productive.

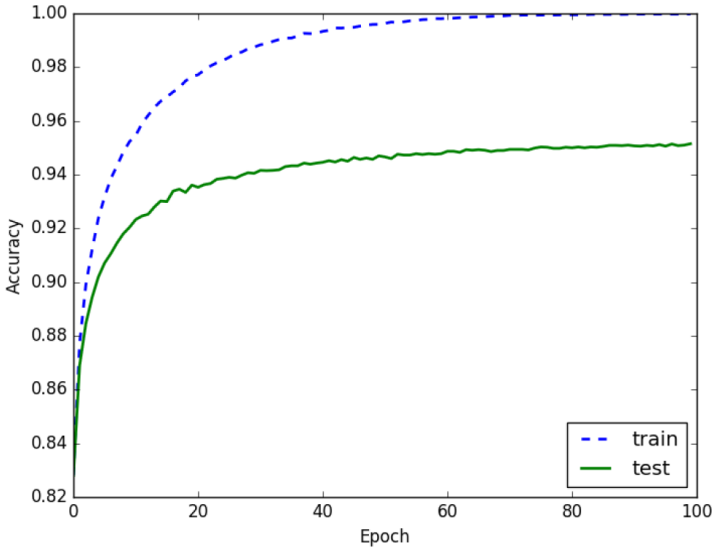

Figure 1: Sample graph. Curves are randomly drawn.

7.2 K-Fold Cross Validation (5 points)

Perform K-fold cross validation (with K = 5) using the train set for evaluating any one of the models (GD or SGD). Any loss function can be used. Include a file called `cv.py` which has the implementation and report results in the homework report.

1. Randomly split your entire dataset into K - 'folds'
2. For each K - fold in your dataset, build your model on K – 1 folds of the dataset. Then, test the model to check the effectiveness for K^{th} fold
3. Record the accuracy you see on each of the predictions
4. Repeat this until each of the K - folds has served as the test set
5. The average of your K recorded accuracy is called the cross-validation accuracy and will serve as your performance metric for the model.