CS578 HW3

November 1, 2018

Total Points: 100
Due date: 11:59pm November 13, 2018.

1 Task

1.1 Objective

The objective of this assignment is to experiment on Gradient Descent (GD), Stochastic Gradient Descent (SGD), Regularization and Logistic Function. Remember the task of classifying the images of handwritten digits using Perceptron. You will replicate the task using Gradient Descent method. Instead of training 10 perceptrons, you will train 10 classifiers using the GD algorithm, each corresponding to a different output digit. For example, let's index the 10 learners from 0 to 9. Say we are training for an image of the digit 3. Then only the learner with index 3 will have an expected output 1, all the other learners will have expected output 0. While predicting the label of an image, the learner which gives the highest probability prediction will be the winner, i.e. the predicted label of the image will be the index of that learner.

1.2 Dataset

You will use the MNIST[1] database of handwritten digits for this task. You will have to download it from the website and parse it. The training examples are in `train-images-idx3-ubyte.gz` and the test examples are in `t10k-images-idx3-ubyte.gz` which you will find on the website containing the dataset. There are $60k$ training examples and $10k$ test examples. You will use the first $10k$ images in the training set for training and all the images in the test examples for testing. Discard the rest $50k$ of the training examples. Shuffle the training examples before training.

[1] http://yann.lecun.com/exdb/mnist/

1.2.1 Feature Type-1

In the MNIST dataset each digit is represented as a 28x28 dimensional vector here, each cell having a value in range $[0, 255]$. Divide each value by 255. Don't round it to 0 or 1. Make the 2-d vector a 1-d vector of size 784 to use as features.

1.2.2 Feature Type-2

You will reduce the feature size using sliding window and max-pooling. This is called sub-sampling. Use a 2x2 sliding window to reduce the 28x28 dimensional images to 14x14 dimensional. This is how it is done: slide a 2x2 window starting from the upper-leftmost corner of the 2-d image and take the maximum of the four values. Then slide the window to the right until you reach to the upper-rightmost corner of the image. Then go down and repeat the same. In this manner you will be sub-sampling the image into 14x14 dimension. Now, use these 196 values as a 1-d feature vector.

1.3 Logistic Regression

You need to use Logistic function for predicting the output of each learner. Remember the form of logistic function discussed in the class. The formula is as follow.

$$z = w^T x$$

$$g(z) = 1/(1 + e^{-z})$$

Where, w is the weight vector and x is the input feature vector and $g(z)$ is the output. Now remember the logistic loss function,

$$Err(w) = -\sum_i y^i \log(g(w, x_i)) + (1 - y^i) \log(1 - g(w, x_i))$$

1.3.1 Open-ended Questions

1. Calculate the gradient of the loss function. **(5 Points)**
2. Prove that the logistic loss function is convex. **(5 Points)**
3. What is regularization? Why is it used? **(5 Points)**
4. What will be the gradient of the loss function if you add the L2 regularization term $(1/2)\lambda||w^2||$ with the logistic loss? **(5 Points)**
5. In GD, 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. Comment on the convergence in Stochastic Gradient Descent.**(5 Points)**
6. What will be the effect of bias term in GD/SGD learning? Justify your answer. If you think it will be useful, use that in the implementation in the same manner you used it in case of perceptron. **(5 Points)**

1.4 Batch Gradient Descent with Logistic Function (30 Points)

- Write down the batch gradient descent algorithm with logistic function in appropriate algorithmic format in LaTeX . **(5 Points)**
- Implement batch gradient descent with logistic function in the described problem. Record the training and test accuracies after every weight update. Let's define this as one epoch. Now plot training and test accuracies vs epoch in a graph. The graph should look exactly like the sample graph attached in the end of this handout. Run this experiments with and without regularization and using Features of Type-1 and Type-2 as described above. So, there will be 4 images like the example attached each having 2 plots. Includes all of the 4 images in the report. Note that the number of epochs will be different in each of the four experiments as you are setting up a stopping criteria. Also, tune the value of λ when you are using regularization.**(25 points)**
- Analyze and compare the convergence of each experiment you did. **(5 Points)**

1.4.1 Scripts

Your script's in/out format should be as follow.

Input format:

```
$ python gd.py [regularization? True/False] [Feature_Type? type1/type2]
[path to DATA_FOLDER]
```

For example, for the command "`python gd.py True type1 data`" will take 1000 training examples, will run the batch gradient descent with regularization using Feature Type-1. You can assume that "`train-images-idx3-ubyte.gz`", "`train-labels-idx1-ubyte.gz`", "`t10k-images-idx3-ubyte.gz`", "`t10k-labels-idx1-ubyte.gz`" all reside directly in the folder named "`data`".

Output format:

```
epoch 1: Training loss: 2.34, Training Accuracy: 0.23, Test Accuracy: 0.15
epoch 2: Training loss: 2.15, Training Accuracy: 0.34, Test Accuracy: 0.30
epoch 3: Training loss: 2.03, Training Accuracy: 0.37, Test Accuracy: 0.34
epoch 4: Training loss: 1.99, Training Accuracy: 0.39, Test Accuracy: 0.35
epoch 5: Training loss: 1.93, Training Accuracy: 0.45, Test Accuracy: 0.39
...
...
epoch 13: Training loss: 0.03, Training Accuracy: 0.89, Test Accuracy: 0.87
```

Also, the program should generate an image file containing the plots named `convergence.png`.

1.5 Stochastic Gradient Descent with Logistic Function (30 Points)

- Write down the stochastic gradient descent algorithm with logistic function in appropriate algorithmic format in LaTeX . **(5 Points)**
- Implement stochastic gradient descent with logistic function in the described problem. Record the training and test accuracies after every weight update. Let's define this as one epoch. Now plot training and test accuracies vs epoch in a graph. The graph should look exactly like the sample graph attached in the end of this handout. Run this experiments with and without regularization and using Features of Type-1 and Type-2 as described above. So, there will be 4 images like the example attached each having 2 plots. Includes all of the 4 images in the report. Note that the number of epochs will be different in each of the four experiments as you are setting up a stopping criteria. Also, tune the value of λ when you are using regularization.**(25 points)**
- Analyze and compare the convergence of each experiment you did. **(5 Points)**

1.5.1 Scripts

Your script's in/out format should be as follow.

Input format:

```
$ python sgd.py [regularization? True/False] [Feature_Type? type1/type2]
[path to DATA_FOLDER]
```

For example, for the command `"python sgd.py True type1 data"` will take 1000 training examples, will run the stochastic gradient descent with regularization using Feature Type-1. You can assume that `"train-images-idx3-ubyte.gz"`, `"train-labels-idx1-ubyte.gz"`, `"t10k-images-idx3-ubyte.gz"`, `"t10k-labels-idx1-ubyte.gz"` all reside directly in the folder named `"data"`.

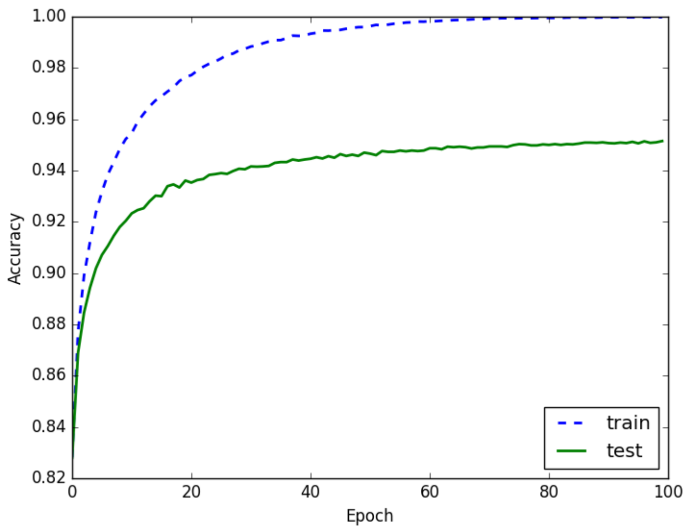
Output format:

```
epoch 1: Training loss: 2.34, Training Accuracy: 0.23, Test Accuracy: 0.15
epoch 2: Training loss: 2.15, Training Accuracy: 0.34, Test Accuracy: 0.30
epoch 3: Training loss: 2.03, Training Accuracy: 0.37, Test Accuracy: 0.34
epoch 4: Training loss: 1.99, Training Accuracy: 0.39, Test Accuracy: 0.35
epoch 5: Training loss: 1.93, Training Accuracy: 0.45, Test Accuracy: 0.39
...
...
epoch 13: Training loss: 0.03, Training Accuracy: 0.89, Test Accuracy: 0.87
```

Also, the program should generate an image file containing the plots named `convergence.png`.

1.6 Important Notes

- You need to implement this assignment from scratch in `Python 2.7`.
- 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.
- For pre-processing and loading the dataset you can use any package installed in `data.cs`. So check in `data.cs` before using any package.
- Use `gnuplot` for generating the graphs.
- Your graphs should look like the following.



2 Submission Procedure

You are required to use LaTeX to type your solutions to questions, and report of your programming as well. `https://www.overleaf.com/` is a website you can use freely as a Purdue student. Other formats of submission will **not** be accepted. A template named "homework_template.tex" is also provided for your convenience. Create a file named `"README"` and write down anything you want to let the TAs know about your program. Especially, when the program does not run without any bug.

Your code will be tested on `data.cs.purdue.edu`, where you submit your homework as well. **Make sure that your program runs properly on `data.cs.purdue.edu`.** After logging into `data.cs.purdue.edu` (physically go to the lab or use ssh remotely, as you are all granted the accounts to CS data machines during this class), please follow these steps to submit your assignment:

1. Make a directory named *'yourname_yoursurname'* (all letters in lower case) and copy all of your files there: `report.pdf`, `gd.py`, `sgd.py` and `README` (optional) shall reside directly in this directory. **Don't put the dataset files.**
2. While in the upper level directory (if the files are in /homes/dan/dan_goldwasser, go to/homes/dan), execute the following command:

   ```
   turnin -c cs578 -p HW3 *your_folder_name*
   ```

   (e.g. your instructor would use: `turnin -c cs578 -p HW3 dan_goldwasser` to submit his homework)

   Keep in mind that old submissions are overwritten with new ones whenever you execute this command.

3. You can verify the contents of your submission by executing the following command:

   ```
   turnin -v -c cs578 -p HW3
   ```

   Do **not** forget the -v flag here, as otherwise your submission would be replaced with an empty one.

Failure to follow the above instructions will incur the penalty when your homework is being graded.

2.1 Late Policies

As declared in the class.

2.2 Plagiarism Policies

Seriuosly, no cheating. If plagiarism was found, both you and the one whose homework you "referred to" receive 0 point on that homework. If you were found to have plagiarised more than once, we will report to the instructor and the department (and your home department if you are not a CS student).