CS589: Machine Learning - Spring 2020

Homework 2: Classification

Assigned: Monday, Mar 2. Due: Friday, Mar 13 at 11:00pm

In this assignment, you are given a set of RGB images (32×32 pixels) with one (and only one) of the following objects: trucks, planes, boats and cars (labels 0, 1, 2 and 3 in `run_me.py`, respectively). The goal is to train a model to recognize which of the objects is present in an image. Some samples of the training images are:

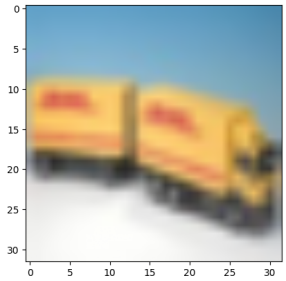

Figure 1: trucks

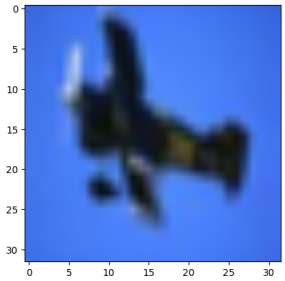

Figure 2: planes

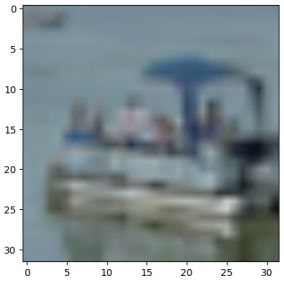

Figure 3: boats

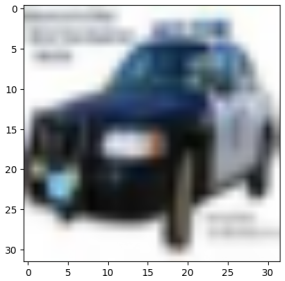

Figure 4: cars

You will train different models (Decision Trees, Nearest Neighbors, Linear models, Neural Networks), compare their performances and training time, and perform model selection using cross-validation.

Getting Started: In this assignment, you will train and evaluate different classification model on images. Please install `Python 3.7` via `Anaconda` on your personal machine. The environment you created for HW1 should have all dependencies except for `autograd` (whose details have been given below). Download the homework file `HW02.zip` via `Piazza`. Unzipping this folder will create the directory structure shown below,

```
HW2
--- Data
    |--data_train.npy
    |--data_train_knn.npy
    |--data_test.npy
    |--train_labels.npy
    |--train_labels_knn.npy
--- Submission
    |--Code
    |--Predictions
```

The data files are in `Data` directory respectively. You will write your code under the `Submission/Code` directory. Make sure to put the deliverables (explained below) into the respective directories.
Deliverables: This assignment has three types of deliverables: a report, code files, and Kaggle submissions.

- **Report:** The solution report will give your answers to the homework questions (listed below). Try to keep the maximum length of the report to 6 pages, including all figures and tables. Reports longer than five pages will only be graded up until the first six pages. You can use any software to create your report, but your report must be submitted in PDF format.

- **Code:** The second deliverable is the code that you wrote to answer the questions, which will involve implementing a regression models. Your code must be `Python 3.7` (no iPython notebooks or other formats). You may create any additional source files to structure your code.

- **Kaggle Submissions:** We will use `Kaggle`, a machine learning competition service, to evaluate the performance of your models. You will need to **register** on `Kaggle` using a `umass.edu` email address to submit to Kaggle (you can use any user name you like). You will generate test prediction files, save them in Kaggle format (helper code provided called `Code/kaggle.py`) and upload them to Kaggle for scoring.

Submitting Deliverables: When you complete the assignment, you will upload your report and your code using the `Gradescope.com` service. Here are the steps:

1. Place your final code in `Submission/Code`, and the Kaggle prediction files for your best-performing submission only for each data set in `Submission/Predictions/best.csv`

2. Create a **zip** file of your submission directory, `Submission.zip` (No rar, tar or other formats).

3. Upload this single zip file on Gradescope as your solution to the `HW02-Classification-Programming` assignment. Gradescope will run checks to determine if your submission contains the required files in the correct locations.

4. Upload your pdf report to the `HW02-Classification-Report` assignment. When you upload your report please make sure to select the correct pages for each question respectively. Failure to select the correct pages will result in point deductions.

5. The submission time for your assignment is considered to be the latest of the submission timestamps of your code, report and Kaggle submissions.

Academic Honesty Policy: You are required to list the names of anyone you discuss problems with on the first page of your solutions. This includes teaching assistants or instructors. Copying any solution materials from external sources (books, web pages, etc.) or other students is considered cheating. To emphasize: no detectable copying is acceptable, even, e.g., copying a single sentence from an outside source. Sharing your code or solutions with other students is also considered cheating. Any detected cheating will result in a grade of -100% on the assignment for all students involved (negative credit), and potentially a grade of F in the course.

Data: The dataset is already partitioned into train and test blocks. Each partition is saved in a numpy binary format (.npy) file.

- Size of training set: $20,000 \times 3072$

- Size of testing set: 4000×3072

Loading the files (helper code provided in `run_me.py`) will result in a $N \times p$ matrix, where N is the number of training/testing examples respectively and $p = 3072$ is the number of features. This loads a numpy array containing values between 0 and 256 (which are normalized, ie. divide each value by 256). The labels are not provided for the images in the test set. You need to predict the test labels, kagglize your predictions, upload to Kaggle to an accuracy score. For this assignment you **are allowed** to use `sklearn.model_selection.*` functions.

Kaggle Link: → https://www.kaggle.com/c/cs589s20hw2/
Questions:

1. (*0 points*) Collaboration statement:

Please list the names of anyone you discussed the assignment with, including TAs or instructors.

2. (*25 points*) Decision trees:

(10) a. A labeled data set D with N samples, each of which consists of F binary features, is given. Suppose that feature f_i was chosen in the root node while building a decision tree, splitting the data in two subsets, D_0 and D_1. Give an expression for the information gain in this case. More generally, what criterion can be used to choose which feature to use?

(15) b. Train 5 different decision trees using the following maximum depths $\{3, 6, 9, 12, 14\}$. Using 5-fold cross-validation, estimate the out of sample error for each model, and report them using a table. How does the maximum depth of the tree affect the estimated accuracy? Explain in at most 4 sentences. Choose the model with lowest estimated out of sample error, train it with the full training set, and predict the labels for the images in the test set. Upload your predictions to Kaggle and report the accuracy on the public leaderboard. Is the predicted out of sample error close to the real one (test set)? Make sure that your report clearly states which model was chosen and what was the predicted out of sample error for it.

3. (*25 points*) Nearest neighbors:

(10) a. A labeled dataset D with N samples, each of which consists of F features, is given. Suppose that a new sample X wants to be classified using KNN, what is the time complexity of this operation if a brute force approach is used?

(15) b. Train 5 different nearest neighbors classifiers using the following number of neighbors $\{3, 5, 7, 9, 11\}$. **Importantly** for **this question only** use the **training** data provided in the files `data_train_knn.npy` and `train_labels_knn.npy`. These can be loaded using the function `read_image_data_knn` provided in `run_me.py`. Using 5-fold cross-validation, estimate the out of sample error for each model, and report them using a table. Choose the model with lowest estimated out of sample error, train it with the full training set, and predict the labels for the images in the test set. Upload your predictions to Kaggle and report the accuracy on the public leaderboard. Make sure that your report clearly states which model was chosen and what was the predicted out of sample error for it.

4. (*10 points*) Linear model:

(10) a. Train a linear model using *L2* regularization, with the following regularization constants $\alpha = \{10^{-6}, 10^{-4}, 10^{-2}, 1, 10\}$, and with hinge and logistic regression loss (you will train 10 classifiers). Look at `sklearn.linear_model.SGDClassifier` to answer this question. Using 5-fold cross-validation, estimate the out of sample error for each model, and report them using a table. Choose the model with lowest estimated out of sample error, train it with the full training set, and predict the labels for the images in the test set. Upload your predictions to Kaggle and report the accuracy on the public leaderboard. Make sure that your report clearly states which model was chosen and what was the predicted out of sample error for it.

5. (*40 points*) **Neural Network:** You will train several neural networks (NN) with one hidden layer, as the one shown in Figure 5[1].

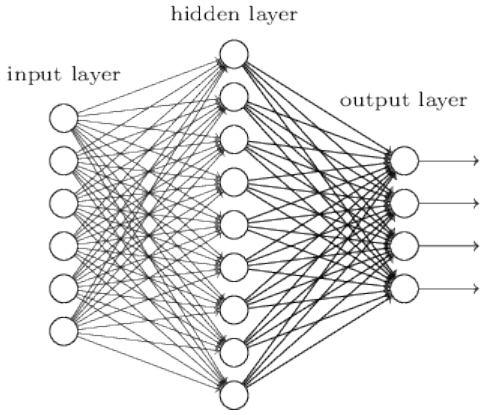

Figure 5: Sample Neural Network

- The size of the input layer is determined by the number of features. We use one feature for each of the 3 color channels of each of the pixels, so this is of size $3 \times 32 \times 32 = 3072$
- The size M of the hidden layer can be chosen freely. You will experiment with different values of $M \in \{5, 40, 70\}$
- The size of the output layer is determined by the number of classes. For this dataset this is 4

The input layer is densely connected to the hidden layer. The hidden layer is densely connected to the output layer. Given an input x the output is given by

$$f(x) = c + V\sigma(b + Wx).$$

Here, let us remind you of the meaning of each of the terms in this equation:

- x is the input, which is a vector of length 3072.
- W is a matrix of size $M \times 3072$ that maps input features to a hidden space.
- b is the "bias" term for the hidden layer. This is a vector of length M.

[1]Image extracted from http://neuralnetworksanddeeplearning.com/chap5.html

- $\sigma(s) = \tanh(s)$ is a nonlinear "sigmoid" function. You will later in this assignment need the relationship that $\sigma'(s) = 1 - \tanh(s)^2$.
- V is a matrix of size $4 \times M$ that maps the hidden features to the output space.
- c is the "bias" term for the output layer. This is a vector of length 4.

Note that $f(x) : \mathbb{R}^{3072} \rightarrow \mathbb{R}^4$ is a function that takes a vector and returns a vector. We write the i-th output of f as $f(x)_i$.
The loss function for this problem is the logistic loss, defined as

$$L(y, f(x)) = -f(x)_y + \log \sum_{i=0}^{3} \exp f(x)_j \tag{1}$$

where y is the correct label associated to the input x and $f(x)_i$ the output of the i-th neuron.

For this assignment, you will implement a simplified version of the backprop algorithm and then use it to train a neural network. In class we first described the backprop algorithm in terms of scalar partial derivatives. However, when the neural network is defined entirely in terms of matrix operations (as $f(x)$ is above) it is much easier to implement backprop using matrix operations.
Because backprop is tricky to implement correctly, you will implement it yourself and then compare the results to those of an automatic differentiation toolbox called `autograd`. You can install autograd by typing,

```
pip install autograd
```

in a terminal or Anaconda. More information on the toolbox is available here: https://github.com/HIPS/autograd/blob/master/README.md. For part (a) you will compare the outputs of your partial derivatives to that of autograd for a single data example. For part (b) you will only use the autograd toolbox to train a neural network. The autograd toolbox computes the partial derivatives for all $20,000$ train examples simultaneously hence speeding up the training of your NN. Helper code is given for both parts of this question.

(20) a. Implement backprop for one data example: The expressions of the gradient of the loss function for a single data example with respect to the four parameters (c, V, b, W, h) is given in the lecture handout. Use these partial derivatives to,

1. **Write python Code computing partial derivatives:** Write code in python implementing the four partial derivatives. Make sure the dimensions match. To get you started we have provided some helper code in function 'NN_gradient_one_sample.py'. Please write your partial derivatives in the 'partial_derivatives' function and list your code in your HW2 report as,

```
dLdc = ...
dLdV = ...
dLdb = ...
dLdW = ...
```

Note that we would like you to list only the four partial derivatives. Please do not list import, print, function definitions, comments or other irrelevant pices of code. Ideally the code listing will be between four and ten lines.

2. **Output values of partial derivatives for one data example:** Copy and paste the output of 'NN_gradient_one_sample.py' in your HW2 report. Sample output should look like,

```
Loss = 123...
dLdc, Autograd
456...
dLdc, partial derivative
456...
etc,
```

(20) b. Train NN using autograd toolbox: For this question, you will train the neural network for each of the values $M \in \{5, 40, 70\}$. For each of these, use autograd to compute the gradient of the logistic loss over the full dataset and perform gradient descent to optimize the parameters (a, b, V, W). Here are some specifics for how to solve this question:

1. Use 1000 epochs (an epoch is defined as one forward and one backward pass on all training examples)
2. Fixed learning rate of 0.0001
3. Use $tanh$ functions for the hidden units
4. Use fixed momentum of 0.1
5. Regularization of the logistic loss function with a constant fixed penalty of 10

To get you started we have provided a helper file `NN_one_layer.py` with these default settings. In this file, we have also included autograd functions to compute the gradients for the entire train dataset. Add your code to this file to,

1. Plot the mean training logistic loss (Equation 1) as a function of the number of epochs for each value of M (one figure with 3 lines, make sure to label the axes and include legends; look at `Code/plotting.py` to get started)
2. Report, using a table, the training time (in milliseconds) for each value of M.
3. Using just one train-validation split of $0.8/0.2$, estimate the validation set error for each value of M and report them using a table. How does the number of hidden units affect the estimated accuracy? Explain in at most 3 sentences. Choose the M with lowest estimated validation error, train it with the full training set, and predict the labels for the images in the test set. Upload your predictions to Kaggle and report the accuracy on the public leaderboard. Is the predicted validation error close to the real one? Make sure that your report clearly states which model was chosen and what was the predicted validation error for it.

Extra Credit: Finally, here are some extra-credit problems. These are more difficult than the above problems and have very small point values. These are also deliberately more open-ended, leaving you more space for creativity. As a result, you will need to carefully describe exactly what you did for each problem. To maximize your score with limited time, you should make sure the above problems are

done thoroughly and ignore these. We will be very stingy in giving credit for these problems– do them only for the glory, and only at your own risk!

6. (*5 points*) **Extra credit:** For the previous question you were asked to train a neural network with only one hidden layer. Derive the expression of the gradient with respect to the parameters for a Neural Network with 3 hidden layers, and train and evaluate the system using the training and testing dataset. Report the results obtained (accuracy and training time) using tables.

7. (*5 points*) **Extra credit:** Visualize the weights in the input and hidden layer and write your interpretation of the weight matrices. This is particularly interesting in image datasets to see the correspondence between the input images and what the NN is learning.