CPSC 340 Machine Learning Take-Home Final Exam (Spring 2020)

Instructions

This is a take home final with three components:

1. an individual component
2. a group component for groups of up to 5
3. and an optional/extra credit component for groups of up to 5.

You may work on the group components as an individual, but it is to your advantage to team up with others. There will be no leniency in grading for smaller groups or individual work.

If you decide to work on the optional question 3, you must do so *with the same group* as for question 2. Please take time at the start to discuss among group members on the plan of approach for this final.

Submission instructions

Typed, LaTeX-formatted solutions are due on Gradescope by **Wednesday, April 29**.

- Please use the `final.tex` file provided as a starting point for your reports.
- Each student must submit question 1 individually as a pdf file named `question1.pdf`. Include your CS ID and student ID. Upload your answer on Gradescope under **Final Exam Question 1**.
- Each group should designate one group member to submit their solution to question 2 (and optionally to question 3) to Gradescope using its group feature (`https://www.gradescope.com/help#help-center-item-student-group-members`). Please hand in your work separately for questions 2 and 3 on Gradescope. Submit a zip file for question 2 under **Final Exam Question 2** and a pdf file for question 3 under **Final Exam Question 3**. Include each group member's CS ID and student ID.

Question 1 [70/100 points]

Recall the MNIST data set from assignment 6 which could be downloaded at `http://deeplearning.net/data/mnist/mnist.pkl.gz`. Go ahead and download this dataset, since we will be using it for this question.

MNIST contains labelled handwritten digits (i.e. 0 to 9) with 60,000 training examples and 10,000 test examples. It is a widely used dataset and with known error rates for several machine learning methods encountered in class. We will be using `http://yann.lecun.com/exdb/mnist/` as a reference for test errors.

For this question, you will implement 5 machine learning methods from class and apply them to the MNIST dataset in order to do supervised classification of digits, with the goal of minimizing the test error. The approaches to be implemented and employed are one example from each of the following types:

1. k-nearest-neighbours (KNN)

2. linear regression
3. support vector machine (SVM)
4. multi-layer perceptron (MLP)
5. convolutional neural network (CNN)

This question will be answered in a report format, Â provided at the end of the exam LaTeXfile `final.tex`. You will have to provide test errors achieved using your implementations, calculated as the percentage of incorrectly labeled test examples (using the default test set provided in the MNIST dataset partition). As an example, results from `http://yann.lecun.com/exdb/mnist/` for each of the above models (with particular hyper-parameter settings) are shown below::

Model	Error (%)
KNN	0.52
linear regression	7.6
SVM	0.56
MLP	0.35
CNN	0.23

Running `python.py main.py -q 1` will load the MNIST dataset into a training set and a test set (if you stored the dataset in a separate directory called `./data/`). The rest of the code (model, training, and testing procedures) must be written by you. You are not permitted to use built-in models (e.g. from PyTorch or scikit-learn), but we encourage you to use code from your assignments. Remember that in past assignments, you have had to implement all of the models listed except for CNNs.

Bundle your code along with a `.pdf` generated from the filled in LaTeXreport into a `.zip` file and submit it to Gradescope. Marks may be taken off for very messy or hard to read code, so make sure to use descriptive variable names and include comments where appropriate. Since we are also marking based on test error, you are expected to only evaluate performance on the test set in the partition provided.

Question 2 [30/100 points]

This part of the final is a group project that takes place on Kaggle, which can be accessed from the following url: `https://www.kaggle.com/c/CPSC340FinalPart2`. You can sign up for a new account or use an existing one; however, note that the Kaggle servers may be in the US, so bear this in mind. We recommend that for data protection purposes you use a non-identifiable (but ideally hilarious) team name. You will link your group members to your team name in your submission document.

Methods that you have learned over the semester are the foundation for solving this task, but they may not be quite enough to solve it well so we recommend that you do some additional research on new methods (as one extremely relevant suggestion, consider looking into transfer learning). Your mark for this part of the final will be based on the score from Kaggle for your test set predictions, a written report that explains your findings, and your code. **Your report should follow the format outlined in** `final.tex`.

The Kaggle competition includes code that will load a dataset of lung X-rays from patients who either have COVID-19 or not (either nothing or another form of pneumonia) if you stored the dataset in a directory called `./data/`. Unlike question 1, you **are** allowed to use built-in models from libaries such as PyTorch or scikit-learn.

Bundle your code along with a `.pdf` generated from the filled in LaTeXreport into a `.zip` file and submit it to Gradescope. Again, marks may be taken off for very messy or hard to read code, so make sure to use descriptive variable names and include comments where appropriate.

It is OK to fail to solve this task "satisfactorily." If your approach is sound and the effort is appropriately high, you will still receive extra credit even if you are unsuccessful. Trying very much counts here.

Question 3 (Optional) [extra 50 points]

This part of the examination is *extra credit* and *entirely optional*. The fundamental design of this question is that we would like to encourage you to try, should you wish, to do *actual good* using your newly acquired machine learning skills.

Significant extra credit can be garnerned in either of two ways:

1. Go find another COVID-19-related machine learning task and attempt to solve it. Report on the task, explain the techniques you applied, and write-up your results.
2. Write a brief report on one of the very recent COVID-19-related research papers to come out of Dr. Wood's PLAI research group [**?**].

Your answer to this question should consist of no more than 3 pages of LaTeX-formatted writing, structured as as either (1) a research paper with Abstract, Introduction, Methods, Experiments, Results, and Conclusion sections or (2) a critical essay relating your understanding of the cited paper. In the latter case we would expect to see at least the following sections: methodological review, summary of results and findings, and next steps (all in your own words). It is OK to fail to solve the task satisfactorily. If your approach is sound and the effort is appropriately high, you will still receive extra credit even if you are unsuccessful. Trying very much counts here.

As with the other two questions, please provide any code you write in answering this question in the accompanying `.zip` file of source code.

Skeleton for Question 1 Answer

Name, Student Number: Rishabh Singal, 18802141
ugrad ID: z7b0b

1 Introduction

Three sentences describing the MNIST classification problem.

Answer: MNIST contains 28x28 pixes images of handwritten, and sometimes obscure, digits from 0-9. The MNIST dataset contains 60,000 images in total, out of which 50,000 are used in this assignment for training while the other 10,000 are used for training. The goal of machine learning models is to try and classify each test image as accurately as possible.

2 Methods

2.1 KNN

Three to four sentences describing the particulars of your KNN implementation, highlighting the hyperparameter value choices you made and why.

Answer: Since KNN (based on euclidean distance) is a non-parametric model and since MNIST dataset is massive, I ran 5-fold cross validation algorithm with k-neighbour values of 1 to 10 on 5000 examples so that no value errors are thrown during the computation of each row's euclidean distance with other ones. Lowest cross-validation error was observed for k-value of 3 and this k-value was used to train the KNN model with 20,000 randomly chosen training examples on Google colab which yielded a test error of 3.2%.

2.2 linear regression

Three to four sentences describing the particulars of your linear regression implementation, highlighting the hyperparameter value choices you made and why.

Answer: I tried several linear regression models in the following order and determined the best hyperparameters for each model using 5-fold cross-validation on various combinations of inputs:

1. Least Squares Linear Regression with L2: This was a regular least squares model which predicted the class by rounding the output value to the nearest integer. As expected, it yielded poor results with the test error being 75.54%.
2. Robust Linear Regression: The weight vector for each class was computed by solving the least squares equation with robust L0 regularization. The prediction was the max index value in each row of output matrix. It yielded test error 16.72%.
3. Least Squares Classifier: This was the same as robust linear regression model except it used L2 regularization and achieved test error of 14.6%.
4. Kernel Linear Regression Polynomial: I computed the polynomial kernel using a small sub-set of training examples (5,000) because any larger and I exceeded my computer's RAM. Using p=3 and L2 $\lambda = 1$, I achieved a test error of 4.3%. I noticed that the presence of λ didn't affect the results much.
5. Kernel Linear Regression RBF (**Best** linear regression model): Finally, I used RBF kernel to train the model and achieved a very low test error of 3.62% with $\sigma = 10$ and L2 $\lambda = 0.01$.

2.3 SVM

Three to four sentences describing the particulars of your SVM implementation, highlighting the hyperparameter value choices you made and why.

Answer: I implemented multi-class SVM with both sum and max losses with L2-regularization. Stochastic gradient descent (with momentum) with a minibatch of 1 and epoch of 5 was used to fit the final model, although epoch of 0.1 was used during cross-validation to select the hyperparameters to select the L2 λ of 0.01 for sum loss and 0.1 for max loss. Sub-gradient approach was used to compute gradients. For this application, multi-clas SVM sum loss yielded better test error of 9.46% compared to 14.22% obtained using max loss.

2.4 MLP

Three to four sentences describing the particulars of your MLP implementation, highlighting the hyperparameter value choices you made and why.

Answer: My MLP implementation in this assignment is essentially a direct carryover from assignment 6 with a small difference except the fact that I'm using Xavier initialization (with a divide by 2 term added in the denominator) to initialize the layer weights and using SGD with momentum. I ran the model with gradient descent only once which achieved a test error of 4.65%. Also I ran the model with SGD with several inputs of epochs (10, 20, 30, 40) and minibatch sizes (100, 500, 1000, 2500) and observed that epoch of 40 with minibatch size of 2500 worked quite well combined with SGD with momentum yielded a low test-error of only 3.28%. The training error was only slightly lower at 2.774%.

2.5 CNN

Three to four sentences describing the particulars of your CNN implementation, highlighting the hyperparameter value choices you made and why.

Answer: In order to create the CNN from scratch, I received significant help from Mahan Fathi's Stanford CS231N Assignment 2 repo (`https://github.com/MahanFathi/CS231/tree/master/assignment2/cs231n`). However, I do completely understand how the backpropagated gradients are derived using computation graphs and how the forward passes of each layer works. Each CNN layer's (convolutional, relu, max-pool, fully-connected and softmax) forward pass takes input matrix X, filter W and bias B (if applicable) as inputs and produces output layer. Each CNN layer's backward pass obtains the gradient flowing from upstream the network as inputs and computes gradients with respect to input matrix X, filter W and bias B (if applicable). Convolution forward pass is done by sliding the filter across the width and height of each input image matrix (1x28x28). A class is defined which constructs a CNN consisting of convolutional - relu - max-pool - fully-connected - relu - fully connected - softmax and initializes the weight and bias matrices similarly to neural_net.py except the convolutional layer filter has a 3-dimensional shape. The dimension of each convolutional filter is 5x5 and 8 filters are used. The size of hidden fully-connected layer is 128. I fit the function as is normally done in CPSC340 by flattening the weights and passing them to the stochastic gradient descent function I wrote (findMinSGD).

However, my loss term kept exploding with different combinations of SGD learning rate, filter dimensions and number of filter and and I could not achieve a result due to time constraints. Perhaps, flattening the filter matrices for SGD isn't yielding good results in this model.

3 Results

Model	Their Error	Your Error (%)
KNN	0.52	3.2
linear regression	7.6	3.62
SVM	0.56	9.46
MLP	0.35	3.28
CNN	0.23	No Result

4 Discussion

Up to half a page describing why you believe your reported test errors are different than those provided (and "detailed" on the MNIST website).

Answer: Following are potential sources of differences for each model:

1. KNN: We're using Euclidean distance to compute the distances between training examples. However, a better measure might be to combine density- and euclidean-based methods to achieve clusters with stronger relationships which would lower the test error.
2. Linear Regression: My kernal RBF linear regression actually performs better than the provided value, but the non-kernel ones did not perform as well which makes sense provided that the data is not contained in a linear sub-space.
3. SVM: They're likely using better stochastic gradient methods such as Adam to fit the training data, whereas I'm just using SGD with momentum which is not as smart as Adam.
4. MLP: Initialization, activation technique, and number of hidden layers make a big difference in the performance of MLP models. However, since the provided MLP code in assignment 6 uses sigmoid activations, adding more than 1 layer will cause problem of vanishing gradients. Hence, better error can be achieved by using smarter initialization techniques for weight matrices (area of active research), changing sigmoid activation to relu, and using Adam SGD method for fitting data.
5. CNN: I did not actually achieve a result for CNN model unfortunately due to time restrictions, but it makes sense that the provided CNN test error is lower than provided MLP test error as CNN actually learns increasingly complex features of the input image such as lines (horiontal, vertical, diagonal), combinations of lines, and patterns as the number of convolutional layers are increased. If a dog image is input into a MLP trained on MNIST digits, it will still confidently predict a digit value, but CNN won't predict anything confidently because the patterns in the dog image do not match the patterns in the digit images.

5 References

1. `https://github.com/MahanFathi/CS231/tree/master/assignment2/cs231n`

Skeleton for Question 2 Answer

Please keep the total length of your entire question 2 response to less than 2 pages. Nothing beyond three pages will be read.

1 Team

Team Members	*all team member names here*
Kaggle Team Name	*your Kaggle team name here*

2 Introduction

A few sentences describing the COVID-19 X-ray classification problem and the problems with it.

3 Method

Several paragraphs describing the approach you took to solving the problem. Highlight in particular how you worked around the small training data problem. Transfer learning is likely something you will want to read about.

4 Experiments

Several paragraphs describing the experiments you ran in the process of developing your Kaggle competition final entry.

5 Results

Model	Kaggle Score
the technical name of your approach	*your kaggle score*

6 Conclusion

Several paragraphs describing what you learned in attempting to solve this problem, why your team is ranked where it is on the leader board, how you might have changed the problem to make its solution more valuable, etc.

Skeleton for Question 3 Answer

EITHER: Write a short, no more than 3 page, research paper about the problem you chose to take on, the approach you took to solving it, the experiments you ran, their outcomes, and what anyone who reads your report should "take home from it." Use the following section labels.

1 Abstract

2 Introduction

3 Methods

4 Experiments

5 Results

6 Conclusion

OR: Write a short, no more than 3 page, report summarizing your understanding of the cited paper. Use at least the following section labels.

1 Introduction

2 Methodological review

3 Summary of results and findings

4 Next steps