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

Instructions

Brian Yeung, Student Number: 39534433, CSID: e0y2b

This is a take home final with two components:

1. an individual component
2. a group component for groups of up to 5. Note that your final and midterm groups will not be allowed to have any overlap in membership besides you.

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.

Submission instructions

Typed, LaTeX-formatted solutions are due on Gradescope by **Wednesday, December 16 at 11:59pm PST**.

- 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 to Gradescope using its group feature (`https://www.gradescope.com/help#help-center-item-student-group-members`). Submit a zip file for question 2 under **Final Exam Question 2**. Include each group member's CS ID and student ID.

Question 1 - Individual [60/100 points]

1 Introduction (3 *points*)

Three sentences describing the MNIST classification problem.

Answer: MNIST is a database of handwritten digits from 0-9, which can be used to train machine learning algorithms. It is a relatively small dataset with 70,000 examples overall, each example being an array of pixel values and the corresponding digit classification. Given the pixel values corresponding to the written digit, we will try to correctly classify the digit.

2 Methods (40 *points*)

2.1 KNN (8 *points*)

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

Answer: I used the KNN implementation from the midterm with cosine distance. I then ran a hyperparameter search from k = 1 to k = 20 and found that the elbow is at k = 4 and the validation error increases from k = 5 onward. I also compared cosine distance to Euclidean distance and found cosine distance outperformed Euclidean distance for every value of k. My model using k = 4 and cosine distance has comparable performance to the reference KNN models using L3 or Euclidean (L2) distance.

2.2 linear regression (8 *points*)

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

Answer: Using the SoftmaxClassifier from a4, initially I had overflow issues, but they were resolved using the hint from Piazza @804. Running a optimization parameter search I found that gamma didn't matter but an optTol=1 gave best results for optimization. The model attains a smaller error than the reference model.

2.3 SVM (8 *points*)

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

Answer: This SVM uses hinge loss sub-gradient descent with a decaying learning rate. Hyperparameter search across C values from 0.5 to 1000 showed that optimal C value is between 100 to 1000. Further fine-tuning resulted in C value of 400. I then tried multiple learning rates and found that an initial lr=1e-4 worked best for 1000 epochs.

2.4 MLP (8 *points*)

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

Answer: This MLP adapted from a6 uses SGD with ReLU activation, decaying learning rate and batchsize of 500. For a single hidden layer, I found that increasing number of hidden units improved performance, from a 3.08% validation error for 32 hidden units to 2.46% for 1024. Manually tuning of learning rate between 1e-5 and 1e-2 showed a learning rate of 3e-3 was best since the loss converges in a reasonable amount of time. I then ran a 2-layer model with 1024 hidden units per layer and found extra layers improved performance.

2.5 CNN (8 *points*)

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

Answer: For the CNN model, I adapted the functions from https://towardsdatascience.com/convolutional-neural-networks-from-the-ground-up-c67bb41454e1 to make my model. I adapted the convolutions to use a single 5x5 filter with stride 1, and replaced the maxpool layer with a third convolutional layer. I then tried various learning rates and found that lr=0.01 worked best.

3 Results (10 *points*)

Model	Their Error	Your Error (%)
KNN	0.52	2.87
linear regression	7.6	7.39
SVM	0.56	8.64
MLP	0.35	2.02
CNN	0.23	11.3

4 Discussion (7 *points*)

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: My KNN model has a 2.87% error, which is comparable with the KNN models provided that use Euclidean (L2) distance. I would probably be able to achieve better performance by preprocessing.

My softmax classifier linear model has an error of 7.39% which is comparable with the linear regression models provided.

My SVM model with hinge loss results in a error of 8.64% which is significantly higher than the error from the reference SVM. This is likely because the reference models use more sophisticated methods such as the Gaussian kernel and higher degree polynomials. As well, the reference models include preprocessing steps such as deskewing.

For the MLP model, increasing size and number of hidden layers seem to improve performance, however, at the expense of training time. My current model has comparable performance with the reference models that use similar amount of hidden units and layers. I think if I used enough computing power and had more layers and hidden units, I would be able to achieve better performance.

As for the CNN model, I adapted the code given at https://towardsdatascience.com/convolutional-neural-networks-from-the-ground-up-c67bb41454e1 to create my model. By replacing the maxpool layer with a convolutional layer, my model has 3 convolutional layers that feed into a MLP to predict the class. This model is very simple and only has a single filter per layer. The error would likely improve if I used multiple filters per layer and included other transformations such as maxpooling or used a deeper model.