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

Instructions

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]

Recall the MNIST data set from assignment 6 which could be downloaded at `https://github.com/mnielsen/neural-networks-and-deep-learning/blob/master/data/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, 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	
linear regression	
SVM	
MLP	
CNN	

Assignment 6 provides code that 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 - Group [40/100 points]

This part of the final is a group project that takes place on Kaggle at `https://www.kaggle.com/t/b1d0feabe2b94cc19c85a5694b606fe9`. You can sign up for a new account or re-use the account for your midterm; Again, 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.

You are not allowed to use any software that you did not develop yourself. There is one exception to this: you may use homework support code and code that you wrote yourself for your homework.

Similar to the midterm, your mark for this part of the final will be based on the score from Kaggle for your test set predictions (see below and the Kaggle competition pages), a written report that explains your findings, and your code. Your report must LaTeXformatted and follow the format given in the answer template for Question 2 below.

Imagine you are an intern working at a self-driving car company and you are writing a behavior prediction algorithm that predicts the trajectory of a car of interest. Your task is to create an ego-centric predictive model of vehicle motion conditioned on past positions (represented as coordinates $(x_{j,t}, y_{j,t})$ for a time step t) of both the ego vehicle and other agents moving about the same intersection. Specifically, given one second of vehicle position data for the ego vehicle and (up to) the ten nearest agents to the ego at the point in time where prediction starts, you need to predict the future position of the ego vehicle 3 seconds into the future.

Data

This dataset we are providing you contains large amounts of road user trajectories (e.g., vehicles, pedestrians) around an unsignalized intersection in the US (right-of-way for motorists, bicyclists, and pedestrians; not controlled by a traffic signal). The data we are providing for this task is a small subset of the data from the Interpret Challenge at `http://challenge.interaction-dataset.com/prediction-challenge/intro`.

`Training/validation data:` You will be provided with 2307 + 523 pairs of `CSV` files (i.e. `X_001.csv` and `y_001.csv`) as your training and validation data respectively, where $\mathbf{X}_j$ is a snapshot of up to the ten nearest vehicles at an intersection over one second in the past, and $\mathbf{y}_j$ is the trajectory of the ego vehicle over the next three seconds. Note that each ($\mathbf{X}_j$,$\mathbf{y}_j$) pair is a training/validation example on one ego vehicle, so it must contain exactly one ego vehicle in each file.

$\mathbf{X}_j$: Each training/validation $\mathbf{X}_j$ contains a column corresponding to the time step (the first column) and 60 more columns that contain information regarding up to 10 agents in the intersection. We have included a table below that mimics the format of each $\mathbf{X}_j$ file (in the table i ranges from 0 to 9 and numbers in tuples are randomly chosen) and instructions on how to read it.

Sorry for this notational clash, but note that ($\mathbf{X}_j$,$\mathbf{y}_j$) are different from the coordinate ($x_{j,t}$,$y_{j,t}$).

time step t	id_i	role_i	type_i	x_i	y_i	present_i	**...**
-1000	**1**	**agent**	**car**	**-4.44**	**0.025**	**1**	**...**
...	**1**	**agent**	**car**	**...**	**...**	**1**	**...**
0	**1**	**agent**	**car**	**0.0**	**0.0**	**1**	**...**

Table 1: Table of $\mathbf{X}_j$ Files

Instructions:

- **time step(ms)** represents the time the agent appears in the snapshot. There are in total ten time steps in the past from -1000ms (one second ago) to 0ms (current).
- id_i represents the unique ID of the agent starting from 1.
- role_i is used to identify the ego vehicle, if role = agent: ego vehicle ; if role = others: other agents.
- type_i represents the types of tracked agents. For example, it can be a car, a pedestrian and so on.
- **(x_i, y_i)** represents the ($x_{i,t}$, $y_{i,t}$) position of the agent i at the current time step t. Each ($x_{i,t}$, $y_{i,t}$) has been transformed to lie in the ego vehicle's reference frame at time 0 with forward towards the "engine" being the positive "y-axis" and towards the passenger door (in a left hand drive vehicle) as positive "x-axis". As a result, you will find that ($x_{i,0}$, $y_{i,0}$) for the ego vehicle (role = agent) is always (0, 0) at time 0. You may find Figure 1 below to be helpful in visualizing this coordinate system.
- present_i is a binary value that represents whether an agent is in the intersection at that time step.

Note that we pick 9-nearest neighbours as other agents (role = others) around an ego vehicle. If there are less than 9 neighbours, the extra columns will be filled with 0s.

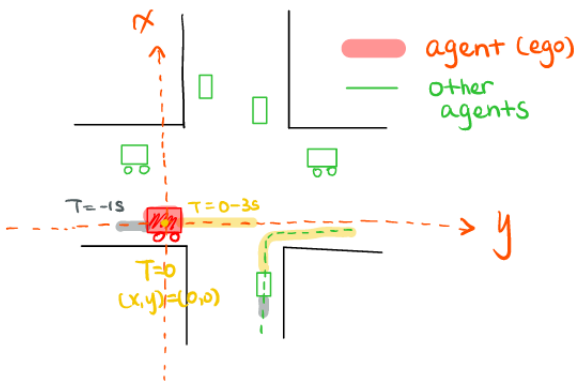

Figure 1: Coordinate System Description

$\mathbf{y}_j$: Each training/validation $\mathbf{y}_j$ is the trajectory of the ego vehicle over the next three seconds in the same coordinate system, so it only contains **x**, **y** and **time step** (0ms to 3000ms).

`Testing data:` The testing data format is exactly the same as the training/validation format, except you are not given $\mathbf{y}_j$, as that is what you need to predict. Specifically, you are given 20 test $\mathbf{X}_j$ `CSV` files, each of which contains the information of an ego vehicle and its surrounding agents over the past one second, and your task is to predict the future trajectory of the ego vehicle over the next three seconds. Note that the final result you hand in shouldn't be separated $\mathbf{y}_j$ files, but should be combined into one table according to the sample submission given on Kaggle.

Finally, there exists a python script to visuzalize the dataset at `https://github.com/interaction-dataset/interaction-dataset`, and instructions on how to use it can be found on Kaggle.

Prediction Task

For each testing data `X_###.csv`, you only need to predict the ego vehicle's future position over the next three seconds. You are not required to make predictions for other nearby agents, but you may find it helpful to do so. You are free to construct any feature set you wish and train as many models as you want in order to solve this task. Your test accuracy will be measured (and compared) against the actual positions using Root Mean Square Error (RMSE) as described in the Kaggle competition.

Submitting Your Results

Similar to the midterm Kaggle competition, your final grade for this question depends on multiple things. Rubrics will grade for test accuracy (i.e. your Kaggle ranking), a description of your pipeline (e.g. data preprocessing, feature engineering, hyper-parameter tuning, evaluation such as cross validation, ...), report writing, code readability, and reflections.

`Kaggle submission`: You will upload your predictions to the Kaggle server. Please refer to the sample submission file on Kaggle for the correct format.

`Gradescope submission`: Bundle your code along with a `.pdf` generated from the filled in LaTeXreport

skeleton 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.

Skeleton for Question 1 Answer

1 Introduction (3 *points*)

Three sentences describing the MNIST classification problem.

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.

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.

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.

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.

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.

3 Results (10 *points*)

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

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).

Skeleton for Question 2 Answer

1 Team

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

2 Introduction (3 *points*)

A few sentences describing the autonomous driving prediction problem.

3 Summary (12 *points*)

Several paragraphs describing the approach you took to address the problem.

4 Experiments (15 *points*)

Several paragraphs describing the experiments you ran in the process of developing your Kaggle competition final entry, including how you went about data prepossessing, feature engineering, model, hyper-parameter tuning, evaluation, and so forth.

5 Results (5 *points*)

Team Name	Kaggle Score
the name of your team	*your kaggle score*

6 Conclusion (5 *points*)

Several paragraphs describing what you learned in attempting to solve this problem, what you might have changed to make the solution more valuable, etc.

7 Code

Include all the code you have written for the autonomous driving prediction problem.