CIS 520, Machine Learning, Fall 2020
Homework 1
Due: Monday, September 21st, 11:59pm
Submit to Gradescope

Instructions. Please write up your responses to the following problems clearly and concisely. We require you to write up your responses using LaTeX; we have provided a LaTeX template, available on Canvas, to make this easier. **Submit your answers in PDF form to Gradescope. We will not accept paper copies of the homework.**

Collaboration. You are allowed and encouraged to work together. You may discuss the **written homework** to understand the problem and reach a solution in groups. However, **it is recommended that each student also write down the solution independently and without referring to written notes from the joint session.** You must understand the solution well enough to reconstruct it by yourself. (This is for your own benefit: you have to take the exams alone.)

Assignment Policies

Instructions. Please write up your responses to the following problems clearly and concisely. We require you to write up your responses using LaTeX; we have provided a LaTeX template, available on Canvas, to make this easier. **Submit your answers in PDF form to Gradescope. We will not accept paper copies of the homework.**

Learning Objectives

After completing this assignment, you will be able to:

- Compute entropy and information gain and describe them intuitively
- Recognize tradeoffs in performance between K-NNs and decision trees
- Recognize and address overfitting in nonparametric models

Deliverables

This homework can be completed individually or in groups of 2. You need to make one submission per group. Make sure to add your team member's name on Gradescope when submitting the homework's written and coding part.

1. **A PDF compilation of `hw1.tex` with team member's names in the agreement**
2. **A `.ipynb` file with the functions implemented**

Homework Submission Instructions

Written Homeworks

All written homework **must** be submitted as a PDF to Gradescope. **Handwritten assignments (scanned or otherwise) will not be accepted.** We require the use of LaTeXto generate your final PDF. We will be posting the homeworks in both PDF and LaTeX source form, and we encourage you to use this source as a template for your submission. **We recommend using Overleaf,** a free online LaTeX editor, though you are welcome to edit assignments however you like.

Coding Homeworks

All coding assignments will be done in Jupyter Notebooks. We will provide a `.ipynb` template for each assignment as well as function stubs for you to implement. Your final submission will be a `.ipynb` file compiled from your notebook submitted to Gradescope. Though you are free to use your own installation of Jupyter for the assignments, **we recommend using Google Colab**, which provides a Jupyter environment connected to Google Drive along with a hosted runtime containing a CPU and GPU.

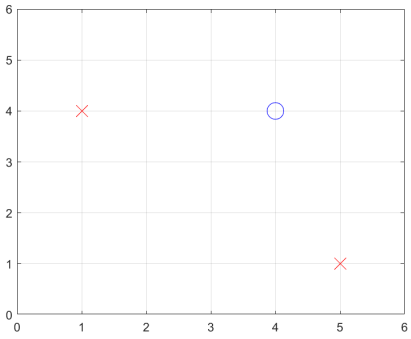
1 Non-Normal Norms [10 points]

1. Given the following four vectors,
   x_1=[6.7, 1.3, 2.7, 0.8]
   x_2=[4.0, 1.6, 5.2, 1.3]
   x_3=[2.9, 2.3, 0.6, 0.1]
   x_4=[3.1, 4.0, 2.7, 2.0]

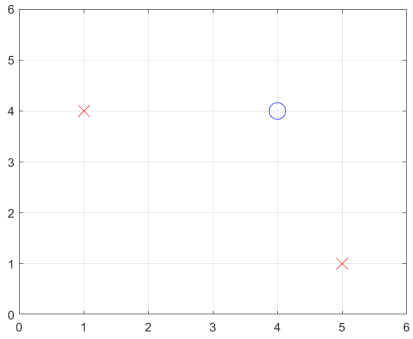
   Which point (other than x_1) is closest to x_1 under each of the following norms?
   a) L_0
   b) L_1
   c) L_2
   d) L_{inf}

2. Draw the 1-Nearest Neighbor decision boundaries with the given norms and lightly shade the region in which points will be classified as circles:

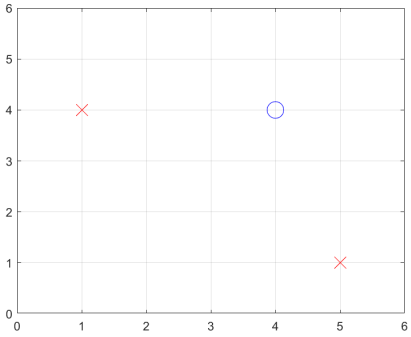
a) L_0



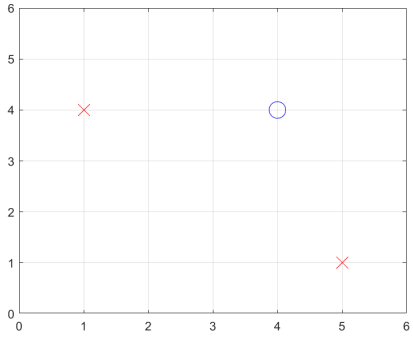
b) L_1



c) L_2



d) L_{inf}



(If there are any classification regions of zero area, draw the line or point of that classification region and label it clearly as belonging to either x or o. In the event of region that is tied, leave it unmarked.)

Hint: you can

- *scan: use your phone or a scanner to take an image of the graph with your solution drawn in and include it in the .pdf you hand in or*
- *use a pdf tool like adobe acrobat to write directly on the pdf or*
- *use any editor you like (e.g. powerpoint)*

2 Decision Trees [25 points]

1. Consider the following set of training examples for the unknown target function $< X_1, X_2 > \rightarrow Y$. Each row indicates the values observed, and how many times that set of values was observed. For example, $(+, T, T)$ was observed three times, while $(-, T, T)$ was observed six times.

Y	X_1	X_2	Count
+	T	T	3
+	T	F	6
+	F	T	4
+	F	F	9
-	T	T	6
-	T	F	4
-	F	T	3
-	F	F	5

Table 1

(a) What is the sample entropy $H(Y)$ for this training data (with logarithms base 2)?

(b) What are the information gains $IG(X_1) \equiv H(Y) - H(Y|X_1)$ and $IG(X_2) \equiv H(Y) - H(Y|X_2)$ for this sample of training data?

(c) Draw the decision tree that would be learned by ID3 (without postpruning) from this sample of training data. (Hints: Entropy is zero when one outcome is certain. Be sure to specify the predicted values for the leaves; This is done by a simple majority vote of the labels of examples classified by each leaf in the training data. Feel free to include a photo instead of using latex.)

2. When we discussed learning decision trees in class, we chose the next attribute to split on by choosing the one with maximum information gain, which was defined in terms of entropy. To further our understanding of information gain, we will explore its connection to Kullback-Leibler (KL) divergence. KL divergence (also called relative entropy) is a measure of how one probability distribution $q(x)$ is different from a second, reference probability distribution $p(x)$. In general,

$$KL\left(p(x)||q(x)\right) = -\sum_x p(x)\ \log\left(\frac{q(y)}{p(x)}\right)$$

We can define information gain as the KL-divergence from the observed joint distribution of X and Y to the product of their observed marginals.[1]

$$IG(x,y) \equiv KL\left(p(x,y)||p(x)p(y)\right) = -\sum_x \sum_y p(x,y)\ \log\left(\frac{p(x)p(y)}{p(x,y)}\right)$$

When the information gain is high, it indicates that adding a split to the decision tree will give a more accurate model.

(a) If variables X and Y are independent, is $IG(x,y) = 0$? If yes, prove it. If no, give a counter example.

(b) Show that this definition of information gain is equivalent to the one given in class. That is, show that $IG(x,y) = H[x] - H[x|y] = H[y] - H[y|x]$, starting from the definition in terms of KL-divergence.

3 High Dimensional Hi-Jinx [25 points]

Nearest neighbor classifiers can be very flexible and accurate, but what if we have many irrelevant features? Suppose that we have two classes, $y = 1, 2$ and $P(\mathbf{x}|y)$ is Gaussian. Let's consider what happens to distances

[1] The negative sign is introduced in this definition because $\log(p/q) = -\log(q/p)$; flipping the fraction will give us KL as defined previously.

between points in the same class and in different classes as the dimension of $\mathbf{x}$ grows but only one of the dimensions is informative. We'll assume for simplicity that for all Gaussians below the variance is the same, σ^2. Reminder: If $X \sim N(\mu, \sigma^2)$, $\mathbf{E}[X] = \mu$, $Var[X] = \sigma^2$, and $\mathbf{E}[X^2] = \mu^2 + \sigma^2$. Also, recall that expectation is linear, so it obeys the following three properties:

$$\begin{aligned} \mathbf{E}[X + c] &= E[X] + c \quad \text{for any constant } c \\ \mathbf{E}[X + Y] &= E[X] + E[Y] \\ \mathbf{E}[aX] &= a\mathbf{E}[X] \quad \text{for any constant } a \end{aligned}$$

Also, note that if X and X' are independent, then $\mathbf{E}[XX'] = \mathbf{E}[X]\mathbf{E}[X']$.

Finally, you may leave summations in your answers below. There's no need to convert to matrix notation.

1. (Intra-class distance) Consider two points from class 1 in one dimension: $X \sim N(\mu_1, \sigma^2)$ and $X' \sim N(\mu_1, \sigma^2)$. What is the expected squared distance between them? ($\mathbf{E}[(X - X')^2] = \ldots$)
2. (Inter-class distance) Now consider two points from different classes in one dimension. Now X and X' have different means: $X \sim N(\mu_1, \sigma^2)$ and $X' \sim N(\mu_2, \sigma^2)$. What is the expected squared distance between them? ($\mathbf{E}[(X - X')^2] = \ldots$)
3. (Intra-class distance, m-dimensions) Again, consider two points from class 1 but in m dimensions. For each dimension j, we have a different mean μ_{1j}: jth dimension of X is $X_j \sim N(\mu_{1j}, \sigma^2)$ and jth dimension of X' is $X'_j \sim N(\mu_{1j}, \sigma^2)$. What is the expected squared distance between them? ($\mathbf{E}[\sum_{j=1}^{m}(X_j - X'_j)^2] = \ldots$)
4. (Inter-class distance, m-dimensions) Finally, consider two points from different classes but in m dimensions. That is, jth dimension of X is $X_j \sim N(\mu_{1j}, \sigma^2)$ and jth dimension of X' is $X'_j \sim N(\mu_{2j}, \sigma^2)$. What is the expected squared distance between them? ($\mathbf{E}[\sum_{j=1}^{m}(X_j - X'_j)^2] = \ldots$)
5. Suppose that only one dimension is informative about class values, that is $\mu_{11} \neq \mu_{21}$, but all others have the same mean $\mu_{1j} = \mu_{2j}$ for $j = 2, \ldots, m$. Write down the ratio of expected intra-class distance divided by inter-class distance under this assumption. Briefly explain the significance of this ratio. As m increases towards ∞, what value does this ratio approach? What does this limit imply about the performance of a NN classifier in this case?

4 K-nearest neighbors Classification (Programming) [15 points]

In this coding exercise, we will implement the K-nearest Neighbors (KNN) algorithm. You are provided with a Jupyter Notebook in which you will have to fill in the functions as instructed therein. Be sure to read the notebook thoroughly for the instructions and also comment your code appropriately.

This is a classification problem and we will use the Diabetes dataset(link given in the Jupyter notebook). We have loaded the dataset for you in the Jupyter notebook.

A training data (X_train) is provided which has several datapoints, and each datapoint is a p-dimensional vector (i.e., p features). You are also provided with separate validation (X_val) and test sets (X_test). Your task is to implement the K-nearest neighbors algorithm to determine the ideal combination of the value of K and the metric norm. For this you have to play with different combinations of metric norm and different values of K.

K	Norm	Accuracy (%)
3	L1	—
3	L2	—
3	L-inf	—
5	L1	—
5	L2	—
5	L-inf	—
7	L1	—
7	L2	—
7	L-inf	—

Table 2: Accuracy for the KNN classification problem on the validation set

Compute the accuracy for the X_val set for every combination of K and metric norm. Once, you have decided the ideal value of K and a relevant metric norm using the validation set, use those values to report the accuracy for the test set X_test. You have to use the following values of K = 3, 5 and 7. The different metrics norms to be implemented are: L1, L2 and L-inf. Do not use any library to implement the norms.

Now answer the following questions:

1. How could having a larger dataset influence the performance of KNN?
2. Tabulate your results in Table 2 for the **validation set** as shown below and include that in your latex file.
3. Finally, mention the best K and the norm combination you have settled upon from the above table and report the accuracy on the test set using that combination.
4. The Autograder for your code submission will grade on the correctness of your implementation for the following functions as given in the Jupyter notebook: **distanceFunc**, **computeDistancesNeighbors**, **Majority** and **KNN**. Each of the functions are worth three points.

5 Decision Trees (Programming) [20 points]

In this coding exercise, we will tune decision tree hyperparameters to reduce overfitting. Refer to the corresponding section in the Jupyter Notebook in which you will have to fill in the functions as indicated. Be sure to comment your code properly. Write the associated answers to the questions in the sections below as follows.

You are given the full dataset and a smaller version of the same dataset for this task. Your task is to compare the performance of the decision tree algorithm on the two datasets and determine the ideal value of tree depth and the ideal value of max leaf count using grid search. For this you have to play with different combinations of these two hyperparameters.

At a high level, first, compute and compare the training and testing accuracies of both datasets given a vanilla decision tree (default hyperparameter settings). Then, perform grid search (iterate over all possible hyperparameter settings) to identify the optimal hyperparameter combination. Make sure that you're using the accuracy for the validation set to measure the performance of the pairs of hyperparameter values. Recalculate new training and testing accuracies with the new hyperparameters and compare them to the accuracies with the vanilla decision tree that used default settings. Also plot a graph of training and validation scores when using a decision tree with different values for the leaf count hyperparameter, holding tree depth constant at the value chosen with grid search. Refer to the notebook for more detailed instructions.

Answer the questions below as you fill out the Jupyter notebook.

5.1 Part 1: Effects of Dataset Size on Performance

1. Report the training, validation, and test accuracies on the full and partial datasets below.

Accuracy Scores		
	Full Dataset	Small Dataset
Training Accuracy	?	?
Validation Accuracy	?	?
Test Accuracy	?	?

2. Which dataset had a higher difference between training and test accuracy? Briefly explain why.

5.2 Part 2: Effects of Dataset Size on Performance

1. Report the chosen hyperparameters for the complete and partial set below.

Grid Search Chosen Hyperparameters		
	Full Dataset	Small Dataset
Tree Depth	?	?
Max Leaf Nodes	?	?

2. Did the small dataset have higher or lower chosen hyperparameter values than the full dataset? Briefly explain why.

5.3 Part 3: Retrain Decision Tree and Plot Hyperparameter Search

1. Report the train, validation, and test accuracies after retraining the decision tree with the new hyperparameters. Also paste in the values for the training and validation scores lists when varying the max leaf node count hyperparameter.

Retrained Decision Tree Performance for Small Dataset	
	Score
Training Accuracy	?
Validation Accuracy	?
Test Accuracy	?

Training and Validation List Values	
	List
Training	[?...?]
Validation	[?...?]

2. How did the training accuracy and testing accuracy change after tuning compared to before? Briefly explain why.

3. Paste the plot of training and validation scores with different leaf count values on the small dataset below. Explain any trends or patterns with the plot within validation and training scores and briefly explain why.

6 Feature Scaling Effects (Programming) [5 points]

Up until now, we have not been using standardized features on the coding practices. Let's observe the effects of standardized features with decision trees and KNNs. Standardization, or feature scaling, is a common preprocessing step for data within machine learning. Let us see whether it's necessary.

Different features have different ranges and scales. For example, in a housing dataset, one feature might be price, which will have values ranging in the several thousands, while another feature may be number of household residents, which will often be only a handful. Standardizing data often entails removing the mean value of each feature, then scale it by dividing non-constant features by their standard deviation so that each feature represents its standard deviation from the feature mean.

Do the coding exercise and answer the following questions briefly:

1. Report the training and testing accuracies for unstandardized and standardized data for both Decision Trees and KNNs using their default hyperparameter values. Note that this portion will be graded by the autograder.

Scores for Unstandardized and Standardized Data				
	KNN Unscaled	KNN Scaled	DT Unscaled	DT Scaled
Training Accuracy	?	?	?	?
Test Accuracy	?	?	?	?

2. **[3 points]** What happens to performance when we use standardization for data with decision trees? What about KNN? Briefly explain why each happened.