CS 178: Machine Learning: Winter 2021

Homework 4

Due Date: **Thursday, February 18, 2021**

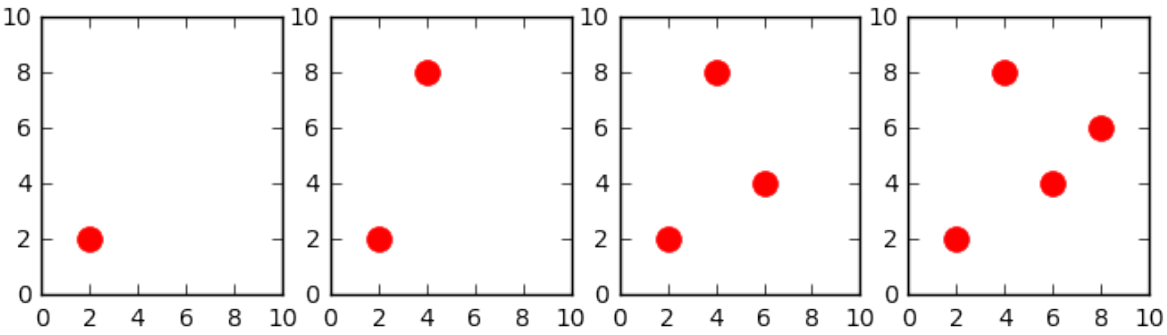
The submission for this homework should be ***a single PDF file*** containing all of the relevant code, figures, and any text explaining your results. Be sure to include copies of any code you write as answers to the appropriate question, so that it may be graded.

Problem 1: Shattering and VC Dimension (15 points)

Consider the data points in Figure 1, each of which is represented by two real-valued features x_1, x_2. For each of classifiers given below, $T(z)$ is the sign threshold function: $T(z) = +1$ for $z \geq 0$, and $T(z) = -1$ for $z < 0$. The classifier parameters a, b, c are real-valued scalars.

Which of the four datasets can be shattered by each classifier? Give a brief explanation justifying your answer, and use your analysis to guess the VC dimension of the classifier. (You do not have to give a formal proof.)

1. $\hat{y}(x) = T(a + bx_1)$. *(5 points)*
2. $\hat{y}(x) = T(a + bx_1 + cx_2)$. *(5 points)*
3. $\hat{y}(x) = T((x_1 - a)^2 + (x_2 - b)^2 + c)$. *(5 points)*



(a) (b) (c) (d)

Figure 1: Four datasets where we seek to determine if they can be *shattered* by different binary classifiers. **No three data points are on a line.**

Problem 2: Decision Trees for Spam Classification (30 points)

We'll use the same data as in our earlier homework: In order to reduce my email load, I decide to implement a machine learning algorithm to decide whether or not I should read an email, or simply file it away instead. To train my model, I obtain the following data set of binary-valued features about each email, including whether I know the author or not, whether the email is long or short, and whether it has any of several key words, along with my final decision about whether to read it ($y = +1$ for "read", $y = -1$ for "discard").

x_1 know author?	x_2 is long?	x_3 has 'research'	x_4 has 'grade'	x_5 has 'lottery'	y $\Rightarrow$ read?
0	0	1	1	0	-1
1	1	0	1	0	-1
0	1	1	1	1	-1
1	1	1	1	0	-1
0	1	0	0	0	-1
1	0	1	1	1	1
0	0	1	0	0	1
1	0	0	0	0	1
1	0	1	1	0	1
1	1	1	1	1	-1

In the case of any ties where both classes have equal probability, we will prefer to predict class +1.

1. Calculate the entropy $H(y)$, using base-2 logarithms, of the binary class variable y. *Hint:* Your answer should be a number between 0 and 1. *(5 points)*

2. Calculate the information gain for each feature x_i. Which feature should I split on for the root node of the decision tree? *(10 points)*

3. Determine the complete decision tree that will be learned from these data. (The tree should perfectly classify all training data.) Specify the tree by drawing it, or with a set of nested if-then-else statements. *(15 points)*

Problem 3: Decision Trees on Kaggle (50 points)

In this problem, we will use our Kaggle in-class competition data to test decision trees on real data. Kaggle is a website designed to host data prediction competitions; we will use it to gain some experience with more realistic machine learning problems, and have an opportunity to compare methods and ideas amongst ourselves. Follow the instructions on the CS178 Canvas and Piazza pages to create an appropriate Kaggle account (if necessary), join our in-class competition, and download the competition data.

1. The following code may be used to load the training features X and class labels Y:

```
X = np.genfromtxt('data/X_train.txt', delimiter=None)
Y = np.genfromtxt('data/Y_train.txt', delimiter=None)
X,Y = ml.shuffleData(X,Y)
```

   Print the minimum, maximum, mean, and variance of each of the 14 features. *(5 points)*

2. To allow faster experimentation, select the first 10,000 (shuffled) data points as training data `Xtr,Ytr`, and the next 10,000 data points as validation data `Xva,Yva`. Learn a decision tree classifier from the training data using the method implemented in the `mltools` package (this may take a few minutes):

```
learner = ml.dtree.treeClassify(Xtr, Ytr, maxDepth=50)
```

   Here, we set the maximum tree depth to 50 to avoid potential recursion limits or memory issues. Compute and report your decision tree's training and validation error rates. *(5 points)*

3. Now try varying the `maxDepth` parameter, which forces the tree learning algorithm to stop after at most that many levels. Test `maxDepth` values in the range `0, 1, 2, ..., 15`, and plot the training and validation error rates versus `maxDepth`. Do models with higher `maxDepth` have higher or lower complexity? What choice of `maxDepth` provides the best decision tree model? *(10 points)*

4. The `minParent` parameter controls the complexity of decision trees by lower bounding the amount of data required to split nodes when learning. Fixing `maxDepth=50`, compute and plot the training and validation error rates for `minParent` values in the range `2**[2:12]=[4,8,16,...,4096]`. Do models with higher `minParent` have higher or lower complexity? What choice of `minParent` provides the best decision tree model? *(10 points)*

5. Our Kaggle competition measures performance using the ROC curve, specifically the *area under curve* (AUC) score. For the best decision tree model trained in the previous parts, use the `roc` function to plot an ROC curve summarizing your classifier performance on the 10,000 training points, and another ROC curve summarizing your performance on the 10,000 validation points. Then using the `auc` function, compute and report the AUC scores for the training and validation data. *(10 points)*

6. Based on your results in the previous parts, pick the `maxDepth` and `minParent` values that you think will perform best. Retrain your decision tree model using as much of the data in `X_train.txt` as possible (preferably, all of it). Then using code like the following, make predictions on all 200,000 test points, and export your predictions in the format required by Kaggle:

```
learner = .. # train one using X,Y
Xte = np.genfromtxt('data/X_test.txt', delimiter=None)
Yte = np.vstack((np.arange(Xte.shape[0]), learner.predictSoft(Xte)[:,1])).T
# Output a file with two columns, a row ID and a confidence in class 1:
np.savetxt('Y_submit.txt',Yte,'%d,%.2f',header='ID,Prob1',comments='',delimiter=',')
```

   Submit your predictions on all of the test data to Kaggle, and include your Kaggle username and leaderboard AUC in your homework solutions. *(10 points)*

Note that we use `predictSoft` to output probabilistic predictions (that test examples are members of class 1) for upload to Kaggle. While you may also upload "hard" predictions (class values), accounting for the learned model's confidence in each prediction will produce a smoother ROC curve and (usually) a better AUC score.

Problem 4: Statement of Collaboration (5 points)

It is **mandatory** to include a *Statement of Collaboration* in each submission, that follows the guidelines below. Include the names of everyone involved in the discussions (especially in-person ones), and what was discussed.

All students are required to follow the academic honesty guidelines posted on the course website. For programming assignments in particular, I encourage students to organize (perhaps using Piazza) to discuss the task descriptions, requirements, possible bugs in the support code, and the relevant technical content *before* they start working on it. However, you should not discuss the specific solutions, and as a guiding principle, you are not allowed to take anything written or drawn away from these discussions (no photographs of the blackboard, written notes, referring to Piazza, etc.). Especially *after* you have started working on the assignment, try to restrict the discussion to Piazza as much as possible, so that there is no doubt as to the extent of your collaboration.