EECS 349 Machine Learning

Exam 1

April 27, 2018

Name: ____

1. (0.5 points) You're learning a decision tree to predict whether recipes are delicious, based on ingredients. You train a decision tree on a data set of 100 example recipes and achieve 100% accuracy on the training examples. If you then test your tree on a separate set of 50 different recipes, you would generally expect the accuracy to be:
   (a) 100%
   (b) Less than 100%.

2. (0.25 points) Yoshi is debating between two different techniques for model selection. He has a training set of 1000 examples. He is trying to choose between 10-fold Cross Validation (*10-fold CV*) versus using 70% of the examples for training and 30% of the examples for testing (*70/30 split*). His concerns are (a) which method will return a better estimate of the accuracy he can expect his deployed model (trained on the full data) to have on future examples, and (b) which method takes longer to run. Which of the following is true?
   (a) 10-fold CV will generally give a more reliable accuracy estimate, but it takes longer to run
   (b) 10-fold CV will generally give a more reliable accuracy estimate, and it runs faster
   (c) 70/30 split will generally give a more reliable accuracy estimate, but it takes longer to run
   (d) 70/30 split will generally give a more reliable accuracy estimate, and it runs faster

3. (0.25 points) Yejin is working on a binary classification task with 10,000 training examples and 200 binary attributes. She thinks the target function can be expressed as a small disjunction of conjunctions, like $(x_7 \land \neg x_{40} \land x_{50}) \lor (\neg x_7 \land x_{12} \land x_{27}) \lor (x_5 \land x_{15})$, where $\land$ indicates logical "and" and $\lor$ indicates logical "or." She is not sure which attributes are relevant or what the exact formula is. Which classifier makes more sense to try first?
   (a) Decision Trees
   (b) Nearest Neighbor
   (c) Gradient Descenders

4. (1 point) Sarah is modeling a regression task with two inputs x_1 and x_2. Using a linear regression package on a data set of 700 training examples and 300 validation examples, she is getting 0.2 sum squared error on average, but she thinks the real function is not linear but instead something like $f(x_1, x_2) = w_1 x_1^2 + w_2 x_1 x_2 + w_3 x_2 + w_4$ for some unknown parameters w_i. Which of the following approaches do you think is the best method for Sarah to use to learn her new function?
   (a) She can just switch to using 10-fold CV within the linear regression package.
   (b) She can just pre-process her data in a certain way and keep using the linear regression package on that processed data.
   (c) She can switch to using simulated annealing to find the weights of her new function.

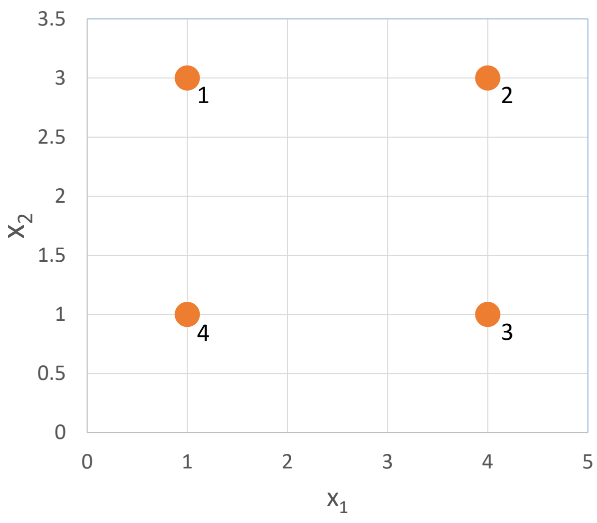

Figure 1: Training Set.

5. (3 points) Consider the figure above of four examples defined over two numeric attributes x_1 and x_2. For each of the following three functions f_i, assign a label of $+$ or $-$ to each example such that the following holds:

   (a) For f_1, 1-nearest-neighbor achieves 100% accuracy in leave-one-out CV (LOOCV), and 3-nearest neighbor achieves 0% LOOCV accuracy.

   (b) For f_2, 3-nearest-neighbor achieves 75% LOOCV accuracy.

   (c) for f_3, k-nearest-neighbor achieves 100% LOOCV accuracy for both $k = 1$ and $k = 3$.

   In all cases, assume a "vanilla" nearest neighbor algorithm using the L2 distance metric, with no attribute weighting or distance weighting. Note, there are multiple correct answers for this question.

example#	x_1	x_2	$f_1(x_1, x_2)$	$f_2(x_1, x_2)$	$f_3(x_1, x_2)$
1	1	3			
2	4	3			
3	4	1			
4	1	1			

```
function DTL(examples, default) returns a tree
  if examples is empty then return tree(default)
  else if all examples have the same classification then return tree(classification)
  else if all splits are trivial then return tree(mode(examples))
  else
        best <- Choose-attribute(attributes, examples)
        new_tree <- a new decision tree with root test best
        for each value v of best:
              $examples_i$ = {elements of examples with best = v}
              subtree = DTL($examples_i$ , mode(examples))
              Add branch to new_tree with label v and subtree subtree
        return new_tree
Note: choose-attribute selects the highest information-gain attribute, and in the case
of ties chooses the first attribute that results in a non-trivial split.
```

Figure 2: Decision tree algorithm.

6. (3.5 points) Consider the following training set for a binary classification task with four categorical (binary) attributes A, B, C, and D. Draw the decision tree learned over these examples, using the algorithm in Figure 2 (which is identical to the algorithm you were asked to implement in Problem Set 1). You should be able to estimate which split has highest information gain without computing the formula.

A	B	C	D	$f(A,B,C,D)$
0	0	0	1	0
0	0	1	1	1
0	1	0	1	0
1	0	0	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	1	1

7. (0.5 points) Now consider the following validation data set. Using this validation set and your tree above, apply a pruning algorithm that repeatedly, greedily chooses the leaf node to prune that most increases validation set accuracy, terminating when there is not a node to prune that will increase accuracy on the validation set. Draw the tree that results after pruning.

A	B	C	D	$f(A,B,C,D)$
0	1	1	1	0
1	0	0	0	1

8. (1 point) In the tennis playing domain, there are three categorical attributes, each with three possible values. The weather x_1 (rainy, cloudy, or sunny), your tennis racket's color x_2 (red, chartreuse, or silver), and your opponent x_3 (Rafael, Conners, or Serena). Consider trying to learn a binary-valued output for whether you win (+) or lose (−). Fill in the below data set of four examples such that 3 nearest neighbor achieves 75% accuracy in LOOCV on the data, whereas 1-nearest-neighbor achieves 25% LOOCV accuracy. Assume that the nearest neighbor converts the categorical attributes into one-hot representations, and uses Hamming distance (which is equivalent to L1 distance in this case) as the distance metric. The best answer will *not* require assumptions about how the algorithm breaks ties in 1-nearest-neighbor when there are two nearest neighbors with differing labels.

x_1	x_2	x_3	$f(x_1, x_2, x_3)$