Homework 3: Decision Trees, Linear Classifiers, Neural Networks

CSE 5522, Spring 19

Your responses to these questions should be submitted electronically; please see the submission instructions below.

Answers to the questions in 1–4 should either be written in an ASCII text file or a PDF file. PLEASE CONVERT WORD DOCUMENTS TO PDF FILES. You may also write it out by hand and scan/photograph and convert to PDF.

PLEASE NOTE: YOU NEED TO SUBMIT RUNNING INSTRUCTIONS FOR THE TA FOR PROBLEM 4!

Submission instructions: Place the answers for the questions and the code/instructions for the fourth question into a zip file, and submit the zip file via Carmen.

1. (15 points) A survey is taken of people in the class of favorite flavor of ice cream among a forced choice of chocolate, vanilla, or strawberry. 70% prefer chocolate, 20% prefer vanilla, and 10% prefer strawberry.

   (a) (5 points) What is the entropy of this distribution? (Show how you derived the entropy.)

   (b) (5 points) How far is this from the maximum amount of entropy that you could observe for a variable that has three values?

   (c) (5 points) Consider the following probability distributions:

   - $\mathbb{P}(Flavor = \langle c, v, s \rangle | Gender = male) = \langle .80, .12, .08 \rangle$
   - $\mathbb{P}(Flavor = \langle c, v, s \rangle | Gender = female) = \langle .67, .22, .11 \rangle$
   - $\mathbb{P}(Flavor = \langle c, v, s \rangle | StudentType = grad) = \langle .76, .16, .08 \rangle$
   - $\mathbb{P}(Flavor = \langle c, v, s \rangle | StudentType = undergrad) = \langle .64, .24, .12 \rangle$
   - $\mathbb{P}(Gender = \langle f, m \rangle) = \langle .75, .25 \rangle$
   - $\mathbb{P}(StudentType = \langle ug, gr \rangle) = \langle .5, .5 \rangle$

   What would an appropriate first decision be for the decision tree algorithm?.

2. (30 points) For the following functions, determine whether the function can be represented by a single perceptron, or if a multi-layer perceptron is needed. Assuming threshold units, provide a set of weights for either the single perceptron or the multi-layer perceptron that represents the function.

   (a) (10 points) The majority function: assuming 7 binary inputs $x_1 \ldots x_7$ (either 0 or 1), 4 or more inputs are 1.

   (b) (10 points) The odd parity function: is 1 if and only if an odd number of binary inputs are 1 (demonstrate with 4 inputs $x_1 \ldots x_4$). (0 otherwise)

   (c) (10 points) x_1 and x_2 are real valued, and is only 1 when x_1 and x_2 are non negative but $x_1 + x_2 < 10$.

3. (5 points) [Exercise 18.11 from AMIA] Suppose you are running a learning experiment on a new algorithm for Boolean classification. You have a data set consisting of 100 positive and 100 negative examples. You plan to use leave-one-out cross-validation and compare your algorithm to a baseline function, a simple majority classifier. (A majority classifier is given a set of training data and then always outputs the class that is in the majority in the training set, regardless of the input.) You expect the majority classifier to score about 50% on leave-one-out cross-validation, but to your surprise, it scores zero every time. Can you explain why?

4. (50 points, plus 20 bonus) My family likes to play games, but considers a wide variety of factors when considering which game to play. I've created a dataset of instances on deciding whether to play "Apples To Apples" or "Settlers of Catan" based on the following attributes:

   - dayOfWeek: Weekday, Saturday, Sunday
   - timeOfDay: morning, afternoon, evening
   - timeToPlay: < 30, $30 - 60$, > 60
   - mood: silly, happy, tired
   - friendsVisiting: no, yes
   - kidsPlaying: no, yes
   - atHome: no, yes
   - snacks: no, yes
   - game (predicted value): SettersOfCatan, ApplesToApples

The dataset comprises the following files:

- `game_attributes.txt`: The list of attributes and their possible values
- `game_attrdata_train.dat`: A comma-separated list of values for each attribute (in the same order as in) for each training instance, with one instance per line
- `game_attrdata_test.dat`: The test instances in the same format

Your job is to build and analyze a linear classifier using an enhanced (but still very simple) version of the perceptron learning algorithm.

(a) (10 points) The first step is to vectorize the data. Conceptually, each training instance pairs a dictionary of attribute-value pairs with a class label, i.e. Train $= \left\{\left(A^{(i)}, C^{(i)}\right)\right\}_{i=1}^{n}$ where $A^{(i)} = \{\text{dayOfWeek} = v_1, \ldots, \text{snacks} = v_8\}$ has an appropriate value $v_1 \ldots v_8$ for each attribute, and where $C^{(i)}$ is either SettersOfCatan or ApplesToApples (and likewise for the test data). You'll need to convert the data to a bunch of numbers, i.e. to Train $= \left\{\left(\mathbf{x}^{(i)}, y^{(i)}\right)\right\}_{i=1}^{n}$ where $\mathbf{x}^{(i)}$ is a vector of numbers representing $A^{(i)}$ and where $y^{(i)}$ is a boolean (and likewise for the test data); for concreteness, we'll let 1 represent SettersOfCatan and use 0 for ApplesToApples. As in the book, we'll let $\mathbf{x}_0$ always have the value 1 to avoid the need for a separate bias weight. You'll need to decide how to map the attribute-value pairs to the rest of the elements of $\mathbf{x}$, and to keep track of the mapping for interpreting your results later.

One question that arises in vectorizing the data is to decide whether to represent values as a continuum (using integers or reals) or as unrelated. For example, for timeOfDay, you could have a single feature $\mathbf{x}_j$ (for some index j) where you represent morning by 1, afternoon by 2 and evening by 3, or you could have three separate boolean features (called a *one-hot* coding). In general, when would it make sense to code values using a continuum of numbers? For this dataset, explain which attributes you decided to represent using one-hots and why.

(b) (20 points) Implement the averaged perceptron algorithm shown in Figure 1 and run it on this dataset, augmenting it so that at the end of each training epoch, it calculates the accuracy of the current model and the averaged model on both the training and

testing data. Next, plot training curves for each of these models. You should run enough training epochs that accuracy on the test data stops improving. Does the averaged model work better than the final model?

(c) (10 points) For the model that performs best on the test data, give a mathematical description of the decision function using the coefficients from the model. Based on the learned weights, which attribute appears to have the greatest impact on the classification decision?

(d) (10 points) Rerun the training and testing steps where, for each attribute, you train a model using all the attributes except that one (this is called an *ablation* test). Comparing the accuracy of these different models, which attribute appears to be the most important one for the classification decision?

(e) (10 **bonus**) In general, which method would you say is the best way to determine the most influential attribute, examining the weights or running an ablation test? Why?

(f) (10 **bonus**) Averaging perceptron models is generally expected to work better than just using the final trained model. Why?

You may use any programming language, but make sure that it is runnable on stdlinux. Give the TA instructions on how to run your program. YOU MAY NOT USE OFF THE SHELF SOLUTIONS — CODE THIS YOURSELF.

Input:

- Vectorized training data $\left\{\left(\mathbf{x}^{(i)}, y^{(i)}\right)\right\}_{i=1}^{n}$ where $\mathbf{x}_0$ is always 1
- The number of training epochs L

Initialization:

- $\mathbf{w} \leftarrow 0$, where $\mathbf{w}$ is a weight vector with the same size as each $\mathbf{x}$, representing the current model
- $\mathbf{t} \leftarrow 0$, where $\mathbf{t}$ is a vector representing the sum of all the intermediate models

Decision function:

- let $h_{\mathbf{w}}(\mathbf{x})$ be 1 if $\mathbf{w} \cdot \mathbf{x} \geq 0$, otherwise 0

Training:

- for $l \leftarrow 1 \ldots L$, $i \leftarrow 1 \ldots n$
  1. $\mathbf{w} \leftarrow \mathbf{w} + (y^{(i)} - h_{\mathbf{w}}(\mathbf{x}^{(i)}))\mathbf{x}^{(i)}$
  2. $\mathbf{t} \leftarrow \mathbf{t} + \mathbf{w}$

Output:

- the averaged model $\frac{1}{Ln}\mathbf{t}$

Figure 1: Averaged Perceptron Algorithm