Problem Set 3: Machine Learning

15-381 - Artificial Intelligence - Spring 2000

Handed out on March 14, 2000
Due at the beginning of class on April 4, 2000
Optional early hand in date (for 10% extra credit) is March 24, 2000

1 Introduction

1.1 Overview

This assignment gives you an opportunity to apply a number of different machine learning techniques to the same data set so you can understand their respective strengths and weaknesses. The assignment involves programming questions and some written questions which involve analyzing the results obtained from running your code. For the programming questions, please copy your files as specified in the assignment to your hand in directory:

`/afs/andrew/scs/cs/15-381/handin/assn3/andrew_userid/`

For the written questions, you must hand in a hard copy (either hand written or typed) of your answers. (Please write your andrew login ID on your homework.) If you have any questions about this assignment please contact Chuck Rosenberg, `chuck@cs.cmu.edu`.

1.2 A three hour tour

It's spring break and you are on vacation at some exotic locale. You decide to take a short boat tour (just three hours long) of the surrounding islands. A freak storm appears during your tour, the boat is smashed, and you and your group ends up on a deserted island. You decide to head off and scout the island for something edible. Suddenly, after many hours of hiking, you come across a woodsy area with a patch of wild mushrooms you have never seen before. After perusing your trusted copy of *The Audubon Society Field Guide to North American Mushrooms*, you realize that this mushroom species has not yet been cataloged. Flush with the excitement of discovering a new species, you are quickly sobered by the fact that besides your two-pack of Twinkies, these mushrooms are the only food source on the island, and you can't even tell if they are poisonous or not. Unfortunately, the Guide clearly states that there is no simple rule for determining the edibility of a mushroom; no rule like "leaflets three, let it be" for Poison Oak and Ivy. Fortunately, you have packed away in your backpack a waterproof solar-powered laptop, which is filled with some machine learning code you've recently been experimenting with. Refusing to give up hope, you start entering the features and edibility of each mushroom in your book into a text file called `mushroom.ssv`.

WARNING: Do not use your learned models or this dataset to make real decisions about the edibility of wild mushrooms. Some of the data labels have been simplified and the dataset has been shortened.

1.3 Getting started

We are supplying the data files and C code for the machine learning algorithms you will be experimenting with in this assignment. Some of the code is complete and in other cases you will have to fill in some small sections of code.

The tar file for this assignment is:

`/afs/andrew.cmu.edu/scs/cs/15-381/381/public/hw3.tar.gz`

Copy this file to your own directory, decompress and untar it, and type `gmake` to compile the code. When the compilation is complete, you should have five executable programs: `nettrain`, `dt`, `ssvsplit`, `jointbayes`, and `naivebayes`. Again, some of these programs are complete, others you will need to complete.

2 Neural Networks

2.1 A simple network

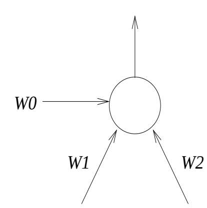
To test that the neural network code is working properly, train a neural network to learn the equivalence function by issuing the following command:

```
./nettrain -n 0.2 -u 2 -e 15000 -o equiv.net -t equiv.ssv -F
```

This command will train a network with a learning rate of 0.2 with two hidden units for 15000 epochs. The `-F` option causes `nettrain` to print out the data set along with the target values and what the network actually outputs after training completes.

Examine the output of the program and verify that the learned network correctly implements the equivalence function. Examine the learned network weights in the file `equiv.net`. (Note that the first weight for each unit is the weight to the constant "1" unit.)

Problem 1a: This is a written question. Draw a diagram of the learned network and hand it in. The network should be drawn as a directed graph with each unit drawn as in the following diagram, with the learned weights on the edges. Note that *W0* represents the weight to a constant one input, the bias term.



Problem 1b: This is a written question. Write out a truth table for each of the units in the network. Write out the truth table for the entire network. Specifically, you should generate binary truth tables, i.e. using the values 1 or 0 for the inputs to each node.

2.2 The mushroom dataset

We have provided the file `mushroom.ssv` which contains the mushroom data set. For each of the 161 mushrooms in the dataset, the mushroom is designated as edible (target value = 1) or poisonous (target value = 0) and the following fourteen discrete valued attributes are specified:

cap_shape: bell, conical, convex, flat, knobbed, sunken
cap_surface: fibrous, grooves, scaly, smooth

cap_color: brown, buff, cinnamon, gray, green, pink, purple, red, white, yellow
bruises: yes, no
stalk_shape: enlarging, tapering
stalk_root: bulbous, club, cup, equal, rhizomorphs, rooted, missing
surface_abv_ring: fibrous, scaly, silky, smooth
surface_blw_ring: fibrous, scaly, silky, smooth
color_abv_ring: brown, buff, cinnamon, gray, orange, pink, red, white, yellow
color_blw_ring: brown, buff, cinnamon, gray, orange, pink, red, white, yellow
ring_number: none, one, two
ring_type: cobwebby, evanescent, flaring, large, none, pendant, sheathing, zone
spore_print_color: black, brown, buff, chocolate, green, orange, purple, white, yellow
population: abundant, clustered, numerous, scattered, several, solitary

One note on terminology, we use the word "attribute" and "feature" interchangeably in this assignment.

2.3 Training the mushroom network

Train a neural network using the mushroom dataset by executing the following command:

```
./nettrain -u 5 -e 1000 -n 0.2 -o mush.net -t mush.ssv -p mush.psv
```

The file `mush.psv` specifies a partition of the mushroom data set into three disjoint parts: a training set, a validation set, and a test set. Only the data in the training set is used to adjust the weights in the network, the other two partitions are used for evaluating the network performance. The following questions should help you understand why this is necessary.

Problem 2a: This is a results question. Copy the trained network file `mush.net` into your hand in directory.

Problem 2b: This is a written question. At what epoch does the network achieve the highest accuracy on the training set? At what epoch does the network achieve the lowest mean squared error on the training set? Answer these same two questions for the validation set (test set 1) and the test set (test set 2).

Problem 2c: This is a written question. Plot the training set mean squared error, the validation set mean squared error, and the test set mean squared error, all on the same graph. Clearly label which curve belongs to which data set. Use an expanded scale for the y-axis on your graph, such as $y = [0.00, 0.08]$, so that the structure of the graph is easily observed.

Problem 2d: This is a written question. By inspecting your graph, does it appear that the mean squared error monotonically decreases for all three datasets? Describe the trend you observe in each of the curves as the number of training epochs increases.

Problem 2e: This is a written question. Given your answers to the previous questions, explain why it makes sense to use three data sets to ensure that we end up with a network which will likely have high accuracy on data we have not yet seen. Specifically explain the purpose of each data set and what problems would arise if all three were not used.

3 Decision Trees

3.1 Entropy required

The decision tree code we provided you is not complete. The code is missing the function which calculates the entropy of a data set. Without this function the code won't produce the correct decision tree, the entropy calculation is needed to calculate information gain which decides which attribute will be used to split on first in the tree.

Problem 3: This is a programming question. Fill in the missing parts of the function `DataEntropy` in the file `student-dt.c` and copy your completed version to your hand in directory. Hints: The code to calculate the entropy is quite simple. Do not modify any other code except the code in the file `student-dt.c`. Your modified version of `student-dt.c` must still work with the other code as we provided it to you.

3.2 Making decisions about mushrooms

Build a decision tree model using the mushroom dataset by executing the following command:

```
dt mush.psv mush.ssv
```

The decision tree code we provided you also uses three disjoint partitions of the data set, it uses the training data to build the tree, the validation data to prune parts of the tree if accuracy can be improved, and the test data to evaluate the performance of the tree.

Problem 4a: This is a results question. Capture the output of the decision tree run into a file named `mush.dt` and copy it to your hand in directory.

Problem 4b: This is a written question. Draw a diagram representing the decision tree before pruning. (For an example see Figure 18.8 on page 537 of Russell and Norvig.) Clearly label the attributes in the nodes and the attribute values in the branches.

Problem 4c: This is a written question. On your drawing of the decision tree, circle and label the section(s) of the tree which were pruned. Did the accuracy on the test set improve after pruning? What were the accuracies on the test and training sets before and after pruning?

3.3 But are we confident?

We should be pretty happy right now, it looks like our decision tree is doing an OK job. But slurping down unidentified fungi is a pretty risky business. We notice that there are a large number of leaves in our decision tree, but our training data set size is pretty small. We may wonder how reliable the decision tree's prediction are. The decision tree code can generate confidence interval estimates for its predictions at each leaf. Run the decision tree code with the confidence calculation option:

```
dt -c mush.psv mush.ssv
```

The decision tree code calculates an approximate two sided 95% confidence interval for the probability that each leaf is either yes or no. One interpretation is that there is a 95% chance that the true probability value lies +/- the interval value centered on the stated probability estimate.

Problem 5a: This is a results question. Capture the output of the decision tree run into a file named `mush-conf.dt` and copy it to your hand in directory.

Problem 5b: This is a written question. It should be obvious that the tree can make certain predictions with high confidence (narrow confidence intervals) and others which have quite low confidence. List a conjunction of attribute values for which the decision tree (before pruning) has the highest confidence in its probability estimate of the target values for that leaf. List a conjunction of attribute values for which the decision tree (before pruning) has the lowest confidence in its probability estimate of the target values for that leaf. Note that if more than one conjunction of attributes has the same confidence, please list only one.

Problem 5c: This is a written question. List a conjunction of attribute values for which the decision tree (after pruning) has the highest confidence in its probability estimate of the target values for that leaf. List a conjunction of attribute values for which the decision tree (after pruning) has the lowest confidence in its probability estimate of the target values for that leaf. Note that if more than one conjunction of attributes has the same confidence, please list only one.

4 Bayes Networks

4.1 Keep it simple

This section focuses on Bayes networks. Bayes networks are an extremely powerful means of representing a probability distribution function. Full blown Bayes networks are very powerful, but can be somewhat complex to handle computationally. For this reason we will be working with two simple Bayes network structures: *Joint Bayes* and *Naive Bayes*.

In the case of *Joint Bayes* we assume full dependence between our input attributes. For every specific conjunction of attribute values we assign a probability. If we had 2 target values and n attributes each with m discrete values we would have to keep track of $2 * m^n$ values.

In the case of *Naive Bayes* we assume full conditional independence between the input attributes. For every value of each attribute, independently, we assign a probability. If we had 2 target values and n attributes each with m discrete values we would have to keep track of $2 * m * n$ values.

4.2 Joint Bayes

The Joint Bayes code we provided you is not complete. The code is missing the functions which collect a histogram from the dataset and use that histogram to make predictions.

Problem 6: This is a programming question. Fill in the missing parts of the functions `BuildJointHistogram` and `CalculateJointAccuracy` in the file `student-bayes.c` and copy your completed version to your hand in directory. (You will be modifying this file again in the next set of problems, please only copy the *final* version of the file into your hand in directory.)

Hints: All of the functions you will need to call are in the file `util-bayes.c`. Do not modify any other code except the code in the file `student-bayes.c`. Your modified version of `student-bayes.c` must still work with the other code as we provided it to you.

4.3 Joint mushroom predictions

Build and evaluate a Joint Bayes model using the mushroom dataset by executing the following command:

`jointbayes mush.psv mush.ssv`

Problem 7: This is a written question. What accuracy does the Joint Bayes model achieve on each of the training set, the validation set, and the test set?

4.4 Naive Bayes

The Naive Bayes code we provided you is not complete. The code is missing the functions which collect a histogram from the dataset and use that histogram to make predictions.

Problem 8: This is a programming question. Fill in the missing parts of the functions `BuildNaiveHistogram` and `CalculateNaiveAccuracy` in the file `student-bayes.c` and copy your completed version to your hand in directory.

Hints: All of the functions you will need to call are in the file `util-bayes.c`. You will also need to pay attention to numerical stability here, multiplying together a large set of small numbers will end up being equal to *zero* very quickly because of limited floating point precision. A fix for this was mentioned in class. Remember that we do not need to know the actual probability value to decide whether "yes" or "no" is the correct answer. In this case a monotonic function of the probability will work just as well. Also, remember, that just because we haven't seen a specific attribute value before does not mean that the probability estimate for that attribute value should be zero. Do not modify any other code except the code in the file `student-bayes.c`. Your modified version of `student-bayes.c` must still work with the other code as we provided it to you.

4.5 Naive mushroom predictions

Build and evaluate a Naive Bayes model using the mushroom dataset by executing the following command:

`naivebayes mush.psv mush.ssv`

Problem 9: This is a written question. What accuracy does the Naive Bayes model achieve on each of the training set, the validation set, and the test set?

4.6 Confidence Revisited

One would think that the Joint Bayes model would have a higher accuracy than the Naive Bayes model since the Joint Bayes model does not make any independence assumptions. However, your results should show that this is not the case. One of the primary reasons that Naive Bayes does better is because it has far fewer parameters to estimate than Joint Bayes and therefore can do a better job of estimating those parameters accurately with a limited amount of data. This effect is reflected in the confidence intervals for the estimated probabilities.

Run Joint Bayes displaying the average confidence intervals:

`jointbayes -c mush.psv mush.ssv`

Problem 10a: This is a written question. What are the average confidence interval values for each of the training, test, and validation sets?

Now run Naive Bayes displaying the average confidence intervals:

`naivebayes -c mush.psv mush.ssv`

Problem 10b: This is a written question. What are the average confidence interval values for each of the training, test, and validation sets?

Even though Naive Bayes seems to perform much better than Joint Bayes, there are still ways to improve the performance of Joint Bayes. One way to do this is to select a subset of the features which are important to deciding our target value and build a joint model on that subset of features. Because only a subset of the features are used, less parameters will need to be estimated and the estimates will be more accurate. The Joint Bayes code we provided is capable of operating on a subset of the features.

Build a Joint Bayes model on a subset of features by executing the following command:

`jointbayes -c -f mush.fsv mush.psv mush.ssv`

The file `mush.fsv` is a special text file which specifies which features to include in the model. The features to be included are listed on a single line, separated by spaces. To see a list of all the features in the data set, examine the list in the file `mush.ssv`.

Problem 10c: This is a written and a result question. Edit the file `mush.fsv` to select a small set of features that improves *both* the accuracy and confidence intervals for the Joint Bayes model over using all the dataset features. (You do not have to include the feature already listed, it is just there as an example, but the target attribute "edible" must always be included.) List which features you selected. What are the improved accuracies and average confidence interval values for each of the training, test, and validation sets? Also, copy your modified version of `mush.fsv` to your hand in directory.

5 Tasty morsels

Problem 11: This is a written question. Given your analysis of the machine learning models you've constructed in this assignment, which one would you choose to make your edibility decisions and why?

6 What to hand in...

Make sure you have copied your versions of the following files to your hand in directory:

- `mush.net`, `mush.dt`, `mush-conf.dt`, `mush.fsv`
- `student-dt.c`, `student-bayes.c`

You must also hand in hard copy of your answers to the questions in the various sections of the assignment and please write your andrew login ID on your homework.

And give my regards to the Skipper and Mary Ann. ;)