CS446: Machine Learning	**Fall 2014**
Problem Set 2 Solutions	
Handed Out: September 16^{th}, 2014	*Handed In: September* 25^{th}, 2014

1 Learning Decision Trees

1.1

There are several decision trees that can represent the data because the information gain is often equal for multiple attributes. Here is one tree, obtained by randomly choosing an attribute upon a tie:

```
if Color = Yellow :
  if Size = Small :
    class = T
  if Size = Large :
    if Age = Child :
      class = F
    if Age = Adult :
      if Act = Dip :
        class = F
      if Act = Stretch :
        class = T
if Color = Purple :
  if Act = Dip :
    class = F
  if Act = Stretch :
    if Age = Adult :
      class = T
    if Age = Child :
      class = F
```

1.2

The misclassification heuristic generates similar trees as the previous case. Here is a slightly different tree generated using this heuristic.

```
if Color = Yellow :
  if Size = Small :
    class = T
  if Size = Large :
    if Age = Child :
      class = F
```

```
    if Age = Adult :
      if Act = Dip :
        class = F
      if Act = Stretch :
        class = T
if Color = Purple :
  if Age = Child :
    class = F
  if Age = Adult :
    if Act = Stretch :
      class = T
    if Act = Dip :
      class = F
```

The primary reason for the trees generated by the two methods being similar to each other is the small size of the data set. However, even in this example, we can see how the two measures of impurity are different from each other. Consider the case of examples with `Color = Purple`. Using entropy, we see that the information gain by splitting on `Age` and `Act` are both 0.3112, while `Size` has no gain, indicating that splitting on `Size` does not change the fraction of `T` and `F` labels. We see from the data that choosing either of the other two attributes will create one subtree that is labeled `F`.

On the other hand, using the majority error, we see that all three attributes are scored zero. Choosing the next split at random, you may end up with a tree that you won't get using entropy-based information gain. In general, suppose a data set has two labels with n^+ and n^- examples for each label with $n^+ > n^-$. If, after splitting on an attribute, we find that for every child node, $n^+ > n^-$, then the information gain measured using the majority error will be zero. (Why?) However, information gain using the entropy can provide a better estimate of impurity in these cases.

1.3

It is possible that by limiting the depth of the decision tree, the leaf nodes are not all labeled with one class. If this happens, we can assign to the leaf the label that is more frequent in the examples corresponding to that leaf.

Using the standard ID3 approach, one of the trees we can get by training on the first twelve examples is:

```
if Color = Yellow :
  if Size = Small :
    class = T
  if Size = Large :
    class = F
if Color = Purple :
  if Act = Dip :
    class = F
```

```
if Act = Stretch :
  class = F
```

This gets an error rate of 0.25. The majority error based approach can also produce the same tree.

2 Decision Trees as Features

We provide a brief description of the target concept and report the results of the different algorithms compared in this problem set.

If you are an Illini fan or an ardent Scrabble player, you will enjoy the target concept! The target concept of the modified Badges data is the following: If the first five characters of the first name contains any consonant from *"fighting"* (i.e. an f, g, h, t, or n) and the first five characters of the last name contains any consonant from *"illini"* (i.e. an l or n), the label is `+`. Also, if the first five characters of both the first name and the last name contain any of the five highest score (or least frequent) tiles in Scrabble (i.e. a z, q, x, j, or k) and the last name is at least as long as the first name, then the label is `+`. For all other names, the label is `-`.

While this is a relatively simple concept to describe, in terms of the features we gave you, it is not easy, though possible, to represent as a tree. The reason is that, while the concept seems simple with the descriptive features given above, the features you actually use do not all appear in the training data provided (but may appear in the test data).

2.1 Experimental Results

The goal of this experiment was to illustrate two concepts:

- The expressive power of decision trees, and the impact of the tree depth on performance.
- The power of transforming from one feature space to a more expressive feature space.

It was easy to illustrate the first concept, as shown in the results below, and in the results of most students. The second concept was more difficult to illustrate. The key reason was that the target concept was "too easy" as reflected in the results of the linear classifier (SGD) over the original space. Consequently, moving the a more expressive features space did not help a lot.

2.1.1 Exemplary Results

In order to give a representative picture of possible results, we have summarize the *test* data results from the top ten[1] student submissions, by performance on part (2.e). To protect the privacy of those involved, individual results are not given, but only the mean values of P_A. Several of these submissions included additional features in their `FeatureGenerator.java` .

[1]With some who did not provide all results removed and others disqualified for irregularities, replaced with the next down the list.

Algorithm	**Mean Acc. P_A (%)**	**Std. Dev.**	99% Interval	**T**	**p-value**
Full decision tree	90.10	3.20	[82.70, 97.60]	1.4932	0.1695
100xDepth 4 + SGD	88.50	3.10	[81.20, 95.70]	1.0552	0.3187
Depth 8 tree	87.50	3.90	[78.40, 96.60]	1.7286	0.1179
Simple SGD (LMS)	85.98	3.17	[78.68, 93.28]	3.0637	0.0134
Depth 4 tree	81.90	5.70	[68.70, 95.00]		

Table 1: Performance of various algorithms on the `badges.test` data from the top ten student submissions (by part (2.e))

From these results we can begin to see that the SGD with decision stubs as features outperforms SGD with the original features and that it also outperforms its constituent parts, with performance significantly above that of a depth-4 tree trained on all the data ($T = 5.0349$, $p = 0.0007$). We note that some of the results included in this table used additional features (for all of their algorithms). By having such features as "contains letter x", and more features to represent name length and relative length, some of these students had generalization accuracies approaching 100%. Improvements in accuracy can also be explained by better selections of learning rate, convergence threshold, and batch-size.

3 Notes about hypothesis testing

The goal in this part of the assignment was to have you think about what it really means for one algorithm to be "better" than another. Conclusions must be backed with a meaningful analysis of data. We are using here the "standard" machine learning methodology for comparing learning algorithms. Namely, we consider the difference in their performance over multiple samples, and assume that these differences are distributed according to the t-distribution. Sec. 3.3 below does just that. In 3.2 we abuse this assumption and use it for a single algorithm. In this case we can have also used the normal distribution.

This note provides the necessary information about t-tests. See also the last few slides in the lecture notes at **http://l2r.cs.uiuc.edu/~danr/Teaching/CS446-14/Lectures/02-LecDT.pdf**

3.1 Hypothesis testing

In hypothesis testing, we form a *null-hypothesis*, H_0 , which explains the data without the effect being sought. Example null-hypotheses are "cancer patients taking drug X live no longer than those taking a placebo" and "algorithm X performs no better than algorithm Y" .

We then calculate the likelihood (*p-value*) of seeing the data (or more extreme data) under the assumption that the H_0 is true. If this likelihood is below a certain *significance threshold*, α , we see that it was unlikely that the model in the null-hypothesis produced the data, and we "reject the null-hypothesis". (Otherwise we "fail to reject the null-hypothesis".)

Rejecting null-hypotheses can add credence to an *alternate hypothesis*, that is, the thing one was trying to show in the first-place.

Significance thresholds of $\alpha = 0.05$ and $\alpha = 0.01$ are typical of science.

3.2 One-sample test

In the basic t-test[1], we assume as our null-hypothesis that a random variable X has mean μ_0.

Given that a sample is drawn from this distribution, we want to test the probability of seeing a particular sample, summarized by its mean ($\bar{x}$), sample standard deviation ($\hat{\sigma}$), and sample size (n).

If the p-value is too small, we will see that it is unlikely that the random process of the null-hypothesis ($\bar{x} = \mu_0$) produced the data, and reject the null-hypothesis.

We find the t-value for our sample:[2]

$$\begin{aligned} T &= \frac{\bar{x} - \mu_0}{\hat{\sigma}/\sqrt{n}} \\ \text{d.f.} &= n - 1 \end{aligned} \tag{1}$$

Which we can use to look up a p-value in our t-table.

3.3 Paired t-test

In some cases, one experiment may produce multiple random variables. For example, "before" (X) and "after" (Y) cases on the same patient. To determine if "after" is any better than "before", you can form a new random variable $Z = Y - X$, and perform the single-sample t-test on the Z's .

Here the null-hypothesis is that the differences are all zero; $\mu_0 = 0$.

Since in every experiment we used identical folds for cross-validation, we can compare the scores between them.

Student	**SGD**	**Full Tree**	**Difference**
1	87.10	90.85	3.75
2	86.60	91.10	4.50
3	87.30	91.80	4.50
4	85.85	88.20	2.35
5	92.35	98.40	6.05
6	86.10	88.20	2.10
7	79.85	88.20	8.35
8	86.65	88.20	1.55
9	83.40	88.20	4.80
10	84.60	88.20	3.60

Table 2: **badges.test** accuracies from the top ten performers by part (2.e).

$\bar{Z} = 4.155$

[2]You may have seen this with $n - 1$ replacing n, this will be accepted for grading.

$\hat{\sigma} = 1.913$

$T = (\bar{Z})(\sqrt{n-1}/\hat{\sigma}) = 6.517$

From this we can look up the *p-value* in the table of t-distribution integrals. (t-table). If you have a statistics package, you may be able to get a more exact *p-value* for your t-statistic.

Here:

$p = 0.00010925$

(Here we have applied the small-sample correction, substituting $n - 1$ for n when n is "small", generally defined as $n \leq 25$.)

References

[1] A. Student, "The Probable Error of a Mean," *Biometrica*, vol. 6, no. 1, pp. 1–25, 1908. [Online]. Available: http://www.jstor.org/pss/2331554