CS446: Machine Learning **Spring 2017**

Problem Set 2

Haoran Wang *Handed In: February 15, 2017*

1. **Answer to problem 1**

(a) In order to determine the attribute to split at the decision tree, we could use the information gain of each specific attribute as heuristic to help make the decision. And we need to first calculate the entropy of the dataset before the calculation of information gain of a specific attribute of the dataset. The basic formulas used in this case are shown as the followings:

$$Entropy(S) = -p_+ \cdot log(p_+) - p_- \cdot log(p_-) \tag{1}$$

$$Gain(S,a) = Entropy(S) - \sum_{v \in Value(a)} \frac{|S_v|}{|S|} \cdot Entropy(S_v) \tag{2}$$

Where S is the dataset, a represents attributes.

The calculation process is implemented as:

$$E(S) = -\frac{35}{50} \cdot log(\frac{35}{50}) - \frac{15}{50} \cdot log(\frac{15}{50})$$
$$\approx 0.8813$$

1) For the attribute Holiday:

$$E(H = yes) = -\frac{20}{21} \cdot log(\frac{20}{21}) - \frac{1}{21} \cdot log(\frac{1}{21}) \approx 0.2762$$

$$E(H = no) = -\frac{15}{29} \cdot log(\frac{15}{29}) - \frac{14}{29} \cdot log(\frac{14}{29}) \approx 0.999$$

2) For the attribute ExamTomorrow:

$$E(ET = yes) = -\frac{10}{15} \cdot log(\frac{10}{15}) - \frac{5}{15} \cdot log(\frac{5}{15}) \approx 0.9183$$

$$E(ET = no) = -\frac{25}{35} \cdot log(\frac{25}{35}) - \frac{10}{35} \cdot log(\frac{10}{35}) \approx 0.86312$$

The information gain of two attributes are calculated as followings:

$$Gain(S,H) = Entropy(S) - E(H = yes) - E(H = no)$$

$$= 0.8813 - \frac{21}{50} \cdot 0.2762 - \frac{29}{50} \cdot 0.999$$
$$\approx 0.1859$$

$$\begin{aligned} Gain(S, ET) &= Entropy(S) - E(ET = yes) - E(ET = no) \\ &= 0.8813 - \frac{15}{50} \cdot 0.9183 - \frac{35}{50} \cdot 0.8612 \\ &\approx 0.00163 \end{aligned}$$

So we see $G(S, H) = 0.1859 > G(S, ET) = 0.00163$, which indicates that we should split at the attribute **Holiday** instead of ExamTomorrow at the root of this decision tree.

(b) For this problem, the heuristic is changed to Majority Error. So we need to calculate the potential misclassification probability in order to select the next attribute to split on.

First, for the root of the decision tree, we notice that the misclassification rates are all the same for the four attributes:

	Inflated=F	Inflated=T
Blue	5	3
Red	2	6
Small	5	3
Large	2	6
Stretch	5	3
Dip	2	6
Adult	5	3
Child	2	6

Table 1: Misclassification Situation Table

So we choose the attribute from left to right to split on the root here. Based on the same idea, we could implement the split on the next attribute which has the minimum error probability. The final decision tree is shown as:

```
if Color = Blue:
    if Size = Small:
       class = F
          if Size = Large:
              if Act = Dip:
                  class = T
              if Act = Stretch:
                  class = F
if Color = Red:
   if Act = Dip:
       class = T
```

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

(c) It cannot guarantee globally optimal. It is a kind of greedy algorithm and it can stuck in local optima. Because it only selects the best attribute in current candidates to split for the next iteration, it will ignore the global effect when choosing an attribute to split. So the decision is made purely based on the information gain of the candidate attributes. If we want to find an optimal solution, we need to implement backtracking algorithm to find the globally optimal strategy.

2. **Answer to problem 2**

(a) SGD Implementation

The idea for Stochastic Gradient Descent algorithm is an on-line learning algorithm. So we need to update the weights every time we see a training example. The update rule is presented after the simplification of the math:

$$\Delta\vec{w_i} = R \cdot (\hat{y_i} - y_i) \cdot \vec{x_i}$$

$$\vec{w_i} = \Delta\vec{w_i} + \vec{w_i}$$

Where R is learning rate, $\hat{y_i}$ is the predicted label(for my implementation, I convert the result of the dotproduct to the predicted label, i.e. $\hat{y_i} \in \{-1, 1\}$), y_i is the actual label of the i_{th} example.

According to this on-line updating rule, we could implement the SGD algorithm by using the attributes and methods of Weka, e.g. Instances. As for the initialization of weights, I chose to initialize them randomly based on a specified seed and constant threshold θ is presented as a separate variable instead of merging it into the weights in this implementation. So we also need to update θ for each training example.

And it is largely impossible that we get a desired classifier through only one epoch training, so we need to go through lots of epochs to update our weights and we also need to set a stopping criteria that tells us that our classifier is well trained and good enough for test. The stopping criteria used here is:

$$|E(\vec{w}^{(j+1)}) - E(\vec{w}^{(j)})| < \epsilon$$

$$E(\vec{w}^{(j)}) = \frac{\sum_{i=1}^{N}(\hat{y_i} - y_i)^2}{N}$$

Cross Validation is used to tune parameters in the implementations. The epoch value is set as 100 for the training of the classifiers.

(b) Decision Tree without depth limit

For the implementation of decision tree without depth limit, we could use the Id3.java in the provided file and ignore the method setMaxDepth(int limit) in this implementation.

(c) Depth limit 4 & 8

We could use the same provided Id3.java and the simple statement: setMaxDepth(4) would help us limit the depth to 4. It is changed to setMaxDepth(8) for depth limited to 8.

(d) Decision Stumps as features

We first need to initialize 100 decision trees with depth limit 4. And then we need to train the 100 decision trees for each training example and output 100 predicted labels about each training example. Then, we use the 100 predicted as features for the SGD linear classifier to predicted one final output label. So it contains two stages, first we need to train 100 decision trees to get 100 output label, then train SGD with 100 feature label to get one single output.

(e) Results

	Average Accuracy
SGD	67.68707482993197%
Decision Tree	72.10884353741497%
Decision Tree(Depth Limit 4)	65.98639455782312 %
Decision Tree(Depth Limit 8)	70.06802721088435 %
Decision Stumps(SGD)	73.12925170068027 %

Table 2: Five Average Accuracy

Several comments on the five results: first, for SGD, I set five different values for the learning rate λ: [0.01, 0.05, 0.001, 0.005, 0.0001] and five stopping criteria ϵ: [0.1, 0.01, 0.05, 0.001, 0.005], after trying different combinations of these parameters in the five cross-validations, the best pair of values are: $\lambda = 0.05$, $\epsilon = 0.01$. Besides, I found other several parameters could impact the average accuracy. The seed of the random initialization of weights has influenced the accuracy, so in my implementation, I tried 20 different seeds and picked up the one that has the highest accuracy(67%) as the result. And the seed for the randomly shuffling the training examples could also influence the final accuracy. Since it is not the most vital parameters for us to tune in this algorithm, so I just ignore more details here about how it will influence the result. And we finally got around 67.69% average accuracy for SGD.

For the implementation of Decision Tree algorithm, we notice different depth limits will present different output. Because the depth influences the generalization of the classifier. The complexity of the classifier is higher if the depth is larger and it is highly possible we will have overfitting problem. If the depth is smaller, we may cannot fully learn the structure of the features and make decision easily and it turns out that lots of them are actually the wrong prediction.

For the implementation of Decision Stump, we found it includes two stages before making a final prediction, which means it has more complex structure. As we use the decision tree as features for the SGD algorithm, we might get better result because we actually would do a second classification based on the 100 features produced by the 100 decision trees, so it is a kind of combination of different prediction based on different angles. Intuitively speaking, it should perform better.

Based on the above results, the rank of five algorithms is:

$$Decision\ Stumps(SGD)\ >\ Decision\,Tree(Without\ depth\ limit)$$

$$>\ Decision\,Tree(Depth\,limit\,8)\ >\ Decision\,Tree(Depth\,limit\,4)\ >\ SGD$$

(f) Student-t Confidence Interval

To calculate the confidence interval, we need to clarify the degree of the freedom in our case. As we totally get five accuracy in the cross validation, so the degree of freedom is $n-1=5-1=4$. Then, we want to calculate 99% confidence interval, the confidence is $1-\alpha=0.99$, so $\alpha=0.01$. Now we are able to look up the t-table, we find the $t_{\frac{\alpha}{2}}=4.604$ for degree of freedom 4. The confidence interval can be obtained via student-t test distribution in this formula:

$$confidence\,interval=\bar{x}\pm t_{\frac{\alpha}{2}}\cdot\frac{\sigma}{\sqrt{N}} \tag{3}$$

where $\frac{\sigma}{\sqrt{N}}$ is the standard deviation of the five values, $\bar{x}$ is the mean of the five values.

The confidence intervals are:

	Average Accuracy	Standard Deviation(%)	Confidence Interval(%)
SGD	67.687%	4.318	[47.5, 87.2]
Decision Tree	72.109%	2.769	[59.5, 85.0]
Decision Tree(Depth Limit 4)	65.986 %	5.39	[40.8, 90.4]
Decision Tree(Depth Limit 8)	70.068 %	3.76	[52.4, 87.0]
Decision Stumps(SGD)	73.129 %	7.95	[35.6, 108.0]

Table 3: Five Confidence Interval

We notice that the standard deviation of the last one is higher than others, it is possible as we totally have five different train set and test set, the variation in the complex structure is possible higher than other algorithms here. As I got these average accuracy in the Decision stumps cross validation: [73.8%, 59.6%, 67.4%, 78.8%, 81.7%], it is clear that there exists higher variation in these five results. And that's why we could see the range of confidence interval is wider than others, even the upper bound is higher than 100%.

(g) Statistical Significance

For the student's t-test, we first propose the null hypothesis: the difference between the average accuracy two consecutive algorithm is 0. I assume a simple null hypothesis here as it also indicates that there is no improvement between two consecutive algorithms in the bunch of experiments. By the way, the algorithms are ranked by the average accuracy in decreasing order. As for the implementation of t-test, I used *t.test()* in *R*(Statistical language) for the output of t-value and p-value. The basic output is:

```
Command in R for t-test

> temp1= c(73.129,72.109,70.068,67.687)
> temp2= c(72.109,70.068,67.687,65.986)
> t.test(temp1,temp2, paired = TRUE, conf.level = 0.99)

    Paired t-test

data:  temp1 and temp2
t = 6.1461, df = 3, p-value = 0.008665
alternative hypothesis: true difference in means is not equal to 0
99 percent confidence interval:
 0.08867454 3.48282546
sample estimates:
mean of the differences
               1.78575
```

From the result of t-test, we could see that the t-value is 6.1461, and p-value is 0.008665. So if we compare p-value with the threshold:

$$p - value = 0.008665 < 1 - \alpha = 1 - 0.99 = 0.01$$

So we reject our null hypothesis and use the alternative hypothesis that there exists difference between the average accuracy of consecutive algorithms. Therefore, the difference is statistically significant.

(h) Conclusion

From the results, we could say that the Decision Stumps(SGD) has the best performance among the five algorithms, though it is only a little bit higher than Decision Tree without depth limit. I think the reason is that the Decision Stumps(SGD) employs more complex structured classifier to train and classify test dataset. It would go through two stages in the implementation, and we get our final prediction from the linear separator which classifier 100 prediction from 100 decision trees. So intuitively speaking, it combines the results from 100 smaller decision trees and learn them to adjust parameters for the SGD classifier. Thus, the decision made by the SGD classifier(of Decision Stump) is more convincing as it could reflect a more accurate underlying structure of the actual model.

However, Simple pure SGD cannot perform as good as SGD on the decision stumps. Because the features maybe not linear separable for the 260 features. As for the decision tree implementations, we notice the performance is better if the depth limit is larger. Because the deeper the tree is, the more features we could add in the tree to make decisions, which is certainly more accurate than stop expanding at a depth limit. So the result is that Decision Tree with depth limit 4 cannot better represent the true model than the Decision Tree with depth limit 8. Conceptually speaking, the actual model may use more than 4 characters (e.g. ‘a’,‘i’,‘r’,‘f’,‘o’,‘x’) to linearly separate the dataset. The depth of 4 decision tree cannot accurately represent this true model. So tree with deeper depth could perform better.

In a nutshell, Decision Stumps(SGD) has the best performance over other four algorithms, and it mainly depends on the complexity of the internal structure to separate dataset according to the features used. The assumption we rely on is that we use the characters on different positions to split the decision tree. Besides, the complexity of the structure of both SGD and decision trees decide the performance of the algorithm.

3. **Tree Display**

(a) Decision Tree(Without depth limit)

```
Decision Tree(Without depth limit)
ID3

lastName0=m = 1
|   firstName2=e = 1: +
|   firstName2=e = 0
|   |   lastName1=o = 1: +
|   |   lastName1=o = 0
|   |   |   firstName0=p = 1: -
|   |   |   firstName0=p = 0
|   |   |   |   firstName0=r = 1: -
|   |   |   |   firstName0=r = 0
|   |   |   |   |   firstName0=y = 1: -
|   |   |   |   |   firstName0=y = 0
|   |   |   |   |   |   firstName1=u = 1: -
|   |   |   |   |   |   firstName1=u = 0
|   |   |   |   |   |   |   lastName3=a = 1
|   |   |   |   |   |   |   |   firstName3=r = 1: +
|   |   |   |   |   |   |   |   firstName3=r = 0: -
|   |   |   |   |   |   |   lastName3=a = 0: +
lastName0=m = 0
|   lastName1=l = 1
|   |   firstName0=d = 1: -
|   |   firstName0=d = 0: +
|   lastName1=l = 0
|   |   lastName2=l = 1
|   |   |   firstName2=r = 1: -
|   |   |   firstName2=r = 0
|   |   |   |   lastName4=n = 1: -
|   |   |   |   lastName4=n = 0
|   |   |   |   |   firstName2=h = 1: -
|   |   |   |   |   firstName2=h = 0: +
|   |   lastName2=l = 0
|   |   |   lastName2=o = 1
|   |   |   |   firstName0=b = 1: -
|   |   |   |   firstName0=b = 0: +
|   |   |   lastName2=o = 0
|   |   |   |   firstName3=f = 1: +
|   |   |   |   firstName3=f = 0
|   |   |   |   |   lastName4=l = 1
|   |   |   |   |   |   firstName2=h = 1: -
|   |   |   |   |   |   firstName2=h = 0
|   |   |   |   |   |   |   lastName0=l = 1: -
|   |   |   |   |   |   |   lastName0=l = 0
|   |   |   |   |   |   |   |   lastName0=q = 1: -
|   |   |   |   |   |   |   |   lastName0=q = 0: +
|   |   |   |   |   lastName4=l = 0
|   |   |   |   |   |   firstName1=o = 1
|   |   |   |   |   |   |   lastName0=f = 1: -
|   |   |   |   |   |   |   lastName0=f = 0
|   |   |   |   |   |   |   |   firstName2=e = 1: -
|   |   |   |   |   |   |   |   firstName2=e = 0
|   |   |   |   |   |   |   |   |   firstName3=a = 1
|   |   |   |   |   |   |   |   |   |   lastName0=h = 1: +
|   |   |   |   |   |   |   |   |   |   lastName0=h = 0: -
|   |   |   |   |   |   |   |   |   firstName3=a = 0
|   |   |   |   |   |   |   |   |   |   firstName2=n = 1: +
|   |   |   |   |   |   |   |   |   |   firstName2=n = 0
|   |   |   |   |   |   |   |   |   |   |   lastName2=n = 1: +
|   |   |   |   |   |   |   |   |   |   |   lastName2=n = 0
|   |   |   |   |   |   |   |   |   |   |   |   lastName1=a = 1: -
|   |   |   |   |   |   |   |   |   |   |   |   lastName1=a = 0
|   |   |   |   |   |   |   |   |   |   |   |   |   firstName3=g = 1: -
|   |   |   |   |   |   |   |   |   |   |   |   |   firstName3=g = 0: +
|   |   |   |   |   |   firstName1=o = 0
|   |   |   |   |   |   |   lastName0=l = 1
|   |   |   |   |   |   |   |   firstName1=a = 1
|   |   |   |   |   |   |   |   |   firstName0=d = 1: +
|   |   |   |   |   |   |   |   |   firstName0=d = 0: -
|   |   |   |   |   |   |   |   firstName1=a = 0: +
|   |   |   |   |   |   |   lastName0=l = 0
|   |   |   |   |   |   |   |   lastName3=m = 1
|   |   |   |   |   |   |   |   |   firstName2=r = 1: -
|   |   |   |   |   |   |   |   |   firstName2=r = 0: +
|   |   |   |   |   |   |   |   lastName3=m = 0
|   |   |   |   |   |   |   |   |   firstName1=e = 1
|   |   |   |   |   |   |   |   |   |   firstName2=n = 1: +
|   |   |   |   |   |   |   |   |   |   firstName2=n = 0
|   |   |   |   |   |   |   |   |   |   |   firstName2=o = 1
|   |   |   |   |   |   |   |   |   |   |   |   lastName0=b = 1: -
|   |   |   |   |   |   |   |   |   |   |   |   lastName0=b = 0: +
|   |   |   |   |   |   |   |   |   |   |   firstName2=o = 0
|   |   |   |   |   |   |   |   |   |   |   |   lastName2=r = 1
|   |   |   |   |   |   |   |   |   |   |   |   |   firstName0=m = 1: -
|   |   |   |   |   |   |   |   |   |   |   |   |   firstName0=m = 0: +
|   |   |   |   |   |   |   |   |   |   |   |   lastName2=r = 0: -
|   |   |   |   |   |   |   |   |   firstName1=e = 0
|   |   |   |   |   |   |   |   |   |   firstName0=t = 1
|   |   |   |   |   |   |   |   |   |   |   lastName4=e = 1: -
|   |   |   |   |   |   |   |   |   |   |   lastName4=e = 0
|   |   |   |   |   |   |   |   |   |   |   |   lastName0=s = 1: -
|   |   |   |   |   |   |   |   |   |   |   |   lastName0=s = 0: +
|   |   |   |   |   |   |   |   |   |   firstName0=t = 0
|   |   |   |   |   |   |   |   |   |   |   firstName3=o = 1
|   |   |   |   |   |   |   |   |   |   |   |   firstName0=a = 1: +
|   |   |   |   |   |   |   |   |   |   |   |   firstName0=a = 0
|   |   |   |   |   |   |   |   |   |   |   |   |   firstName0=m = 1: +
|   |   |   |   |   |   |   |   |   |   |   |   |   firstName0=m = 0: -
|   |   |   |   |   |   |   |   |   |   |   firstName3=o = 0
|   |   |   |   |   |   |   |   |   |   |   |   firstName4=o = 1
|   |   |   |   |   |   |   |   |   |   |   |   |   firstName0=s = 1: -
|   |   |   |   |   |   |   |   |   |   |   |   |   firstName0=s = 0: +
|   |   |   |   |   |   |   |   |   |   |   |   firstName4=o = 0
|   |   |   |   |   |   |   |   |   |   |   |   |   lastName0=s = 1
|   |   |   |   |   |   |   |   |   |   |   |   |   |   firstName0=d = 1: +
|   |   |   |   |   |   |   |   |   |   |   |   |   |   firstName0=d = 0
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   lastName4=h = 1
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   firstName0=s = 1: -
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   firstName0=s = 0: +
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   lastName4=h = 0: -
|   |   |   |   |   |   |   |   |   |   |   |   |   lastName0=s = 0
|   |   |   |   |   |   |   |   |   |   |   |   |   |   lastName3=l = 1
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   firstName0=d = 1: -
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   firstName0=d = 0: +
|   |   |   |   |   |   |   |   |   |   |   |   |   |  lastName3=l = 0: -
----------------------------------------------------------------------------

Correctly Classified Instances          35               76.087  %
Incorrectly Classified Instances        11               23.913  %
Kappa statistic                          0.5199
Mean absolute error                      0.2391
Root mean squared error                  0.489
Relative absolute error                 48.1752 %
Root relative squared error             98.1732 %
Total Number of Instances               46
```

(b) Decision Tree(Depth limit 4)

```
Decision Tree(Depth limit 4)
ID3

lastName2=l = 1
|  firstName2=r = 1: -
|  firstName2=r = 0
|  |  firstName2=m = 1: -
|  |  firstName2=m = 0: +
lastName2=l = 0
|  lastName2=o = 1
|  |  firstName0=d = 1: -
|  |  firstName0=d = 0
|  |  |  firstName2=l = 1: -
|  |  |  firstName2=l = 0: +
|  lastName2=o = 0
|  |  firstName3=f = 1: +
|  |  firstName3=f = 0
|  |  |  lastName0=m = 1
|  |  |  |  firstName0=n = 1: -
|  |  |  |  firstName0=n = 0: +
|  |  |  lastName0=m = 0
|  |  |  |  lastName1=l = 1: +
|  |  |  |  lastName1=l = 0: -
----------------------------------------------------------------------------

Correctly Classified Instances          48               72.7273 %
Incorrectly Classified Instances        18               27.2727 %
Kappa statistic                          0.409
Mean absolute error                      0.3812
Root mean squared error                  0.4582
Relative absolute error                 78.7652 %
Root relative squared error             93.1911 %
Total Number of Instances               66
```

(c) Decision Tree(Depth limit 8)

```
Decision Tree(Depth limit 8)
ID3

```

```
firstName3=f = 1: +
firstName3=f = 0
|   lastName0=c = 1: −
|   lastName0=c = 0
|   |   lastName4=l = 1
|   |   |   lastName0=q = 1: −
|   |   |   lastName0=q = 0: +
|   |   lastName4=l = 0
|   |   |   firstName0=r = 1
|   |   |   |   firstName1=o = 1: +
|   |   |   |   firstName1=o = 0
|   |   |   |   |   firstName1=a = 1: +
|   |   |   |   |   firstName1=a = 0
|   |   |   |   |   |   firstName1=e = 1: +
|   |   |   |   |   |   firstName1=e = 0: −
|   |   |   firstName0=r = 0
|   |   |   |   lastName0=m = 1
|   |   |   |   |   firstName2=n = 1: −
|   |   |   |   |   firstName2=n = 0
|   |   |   |   |   |   firstName0=p = 1: −
|   |   |   |   |   |   firstName0=p = 0
|   |   |   |   |   |   |   lastName2=t = 1
|   |   |   |   |   |   |   |   firstName0=t = 1: +
|   |   |   |   |   |   |   |   firstName0=t = 0: −
|   |   |   |   |   |   |   lastName2=t = 0: +
|   |   |   |   lastName0=m = 0
|   |   |   |   |   lastName0=l = 1
|   |   |   |   |   |   firstName1=a = 1
|   |   |   |   |   |   |   firstName0=d = 1: +
|   |   |   |   |   |   |   firstName0=d = 0: −
|   |   |   |   |   |   firstName1=a = 0: +
|   |   |   |   |   lastName0=l = 0
|   |   |   |   |   |   lastName3=m = 1
|   |   |   |   |   |   |   firstName2=r = 1: −
|   |   |   |   |   |   |   firstName2=r = 0: +
|   |   |   |   |   |   lastName3=m = 0
|   |   |   |   |   |   |   lastName2=l = 1
|   |   |   |   |   |   |   |   firstName2=r = 1: −
|   |   |   |   |   |   |   |   firstName2=r = 0: +
|   |   |   |   |   |   |   lastName2=l = 0
|   |   |   |   |   |   |   |   lastName3=l = 1: +
|   |   |   |   |   |   |   |   lastName3=l = 0: −
────────────────────────────────────────────────────────────────────────

Correctly Classified Instances          48               73.8462 %
Incorrectly Classified Instances        17               26.1538 %
Kappa statistic                          0.4785
Mean absolute error                      0.3371
Root mean squared error                  0.4722
Relative absolute error                 67.4285 %
Root relative squared error             94.4514 %
Total Number of Instances               65
```

(d) SGD and Decision Stumps(SGD)

Since we have no built-in tree display function in the SGD algorithm implementations, so I just list the three tree displays of the Decision Tree classifiers implemented via Id3.java. And these trees come from the the cross-validation set which have the best performance among five cases. So I think it is enough to show the tree information here.