Adaboost using Decision stump and Perceptron

Machine Learning (ITCS 6156) assignment 3

Train small classifiers in a sequence. Combine weak classifiers by letting them vote on prediction. This improves the effectiveness of predictors. Here, we use Decision stump and perceptron as a classifier.

Adaboost using Decision stump and Perceptron

Machine Learning (ITCS 6156) assignment 3

Training:

Perceptron is used as weak classifier and trained a sequence of perceptron and considered a weighted vote of each perceptron. There are 2 types of weights in perceptron viz. Data weights and weights of perceptron. After each perceptron is trained the data weights are re-calculated and given to train the new perceptron. Data weights calculation depends on misclassified data points.

Perceptron Training-

In the program, pseudoinverse is used to calculate the initial weights for the perceptron training. This significantly reduces the number of iteration of perceptron. Below are some important point in training the perceptron:

- Among the misclassified points, pick the one with highest weight.
- Error is calculated as sum of **misclassified data weights**.
- Divided weight vector with the sum of weight vector, to normalize the weight vector so that sums up to 1.
- Perceptron is trained based on pocket algorithm with 200 iterations.

The number of perceptron in a model is limited by **overfitting** or number 20 whichever is less. Alpha is calculated based on the error. Hence, we take the vote of each perceptron we weight it with alpha. Below example shows demonstration for model with 4 perceptron.

Decision Stump training-

Decision stump is a shortest possible decision tree. It only splits on a single attribute at a time and hence is a weak classifier. Here, we will keep on creating the decision stump until the model over fits. The upper limit is 20 for the number of decision stumps.

Note: To decide the splitting attribute I have built all the decision stumps (1 on each attribute) and checked for min error decision tree.

Algorithm works in a similar way to the above perceptron. It calculates validation error to check for overfitting.

Assumption:

Handling '?'s in the data: In the dataset given, we have answers for some questions as '?'. In such case, I have replaced '?' with the majority answer for that particular question.

Error: In-sample error is calculated as summation of misclassified weights instead of traditional error.

Perceptron Performance:

Below are the validation errors for every iteration (perceptron).

Iteration 0 - 0.0245398773006

Iteration 1 - 0.0184049079755

Iteration 2 - 0.0184049079755

Iteration 3 - 0.0122699386503

Hence, there are 4 perceptron created before overfitting. These 4 perceptron are nothing but **weak classifier**. We weight each of these weak classifier to make a strong classifier using below alpha (weight) values.

[2.3387454237838585, 2.682988007510925, 3.375550734468382, 3.721831841557795]

Three hypothesis – perceptron weights are as below:

[array([-5.07340401, 2.97770323, -0.99294285, -5.14979627, 13.7815266 , 0.95241482, -3.02516679, 1.05487437,1.01874946, -7.18180236, 1.0184771 , -5.07021073, 3.000736 , -7.07046937, 1.01304244, -3.04971745, 1.01576666]),

array([-6.07340401e+00, -2.22967708e-02, 7.05715196e-03, -6.14979627e+00, 1.07815266e+01, 1.95241482e+00, -4.02516679e+00, 6.05487437e+00, -1.98125054e+00, -6.18180236e+00, 1.84771023e-02, -4.07021073e+00, 7.35998709e-04, -6.07046937e+00, 2.01304244e+00, -4.04971745e+00, 2.01576666e+00]),

array([-5.07340401, 2.97770323, 1.00705715, -5.14979627, 13.7815266 , 2.95241482, -3.02516679, 5.05487437, -0.98125054, -5.18180236, 3.0184771 , -5.07021073, 1.000736 , -7.07046937, 1.01304244, -1.04971745, -0.98423334]),

array([-4.07340401e+00, 1.97770323e+00, 2.00705715e+00, -1.01497963e+01, 1.67815266e+01, 1.95241482e+00, -6.02516679e+00, 2.05487437e+00, -3.98125054e+00, -8.18180236e+00, 4.01847710e+00, -8.07021073e+00, 7.35998709e-04, -1.00704694e+01, 1.30424381e-02, -2.04971745e+00, 1.57666616e-02])]

Perceptron Performance: Below are the error and corresponding iterations of weak learner. This is calculated for 10 iterations. The algorithm is ran 10 times for the analysis.

```
Iteration  0 - 0.037037037037
Iteration  0 - 0.101851851852
Iteration  0 - 0.0648148148148
Iteration  1 - 0.0648148148148
Iteration  2 - 0.0648148148148
Iteration  3 - 0.0648148148148
Iteration  4 - 0.0648148148148
Iteration  5 - 0.0648148148148
Iteration  0 - 0.0740740740741
Iteration  1 - 0.0740740740741
Iteration  0 - 0.0555555555556
Iteration  0 - 0.0555555555556
Iteration  1 - 0.0462962962963
Iteration  0 - 0.0648148148148
Iteration  0 - 0.0555555555556
Iteration  1 - 0.0555555555556
Iteration  0 - 0.0555555555556
Iteration  1 - 0.0462962962963
Iteration  2 - 0.0462962962963
Iteration  0 - 0.0740740740741
Iteration  1 - 0.037037037037
```

Accuracy- [97.247706422018339, 95.412844036697251, 93.577981651376135, 93.577981651376135, 96.330275229357795, 97.247706422018339, 97.247706422018339, 93.577981651376135, 93.577981651376135, 93.577981651376135]

Decision Stump Performance

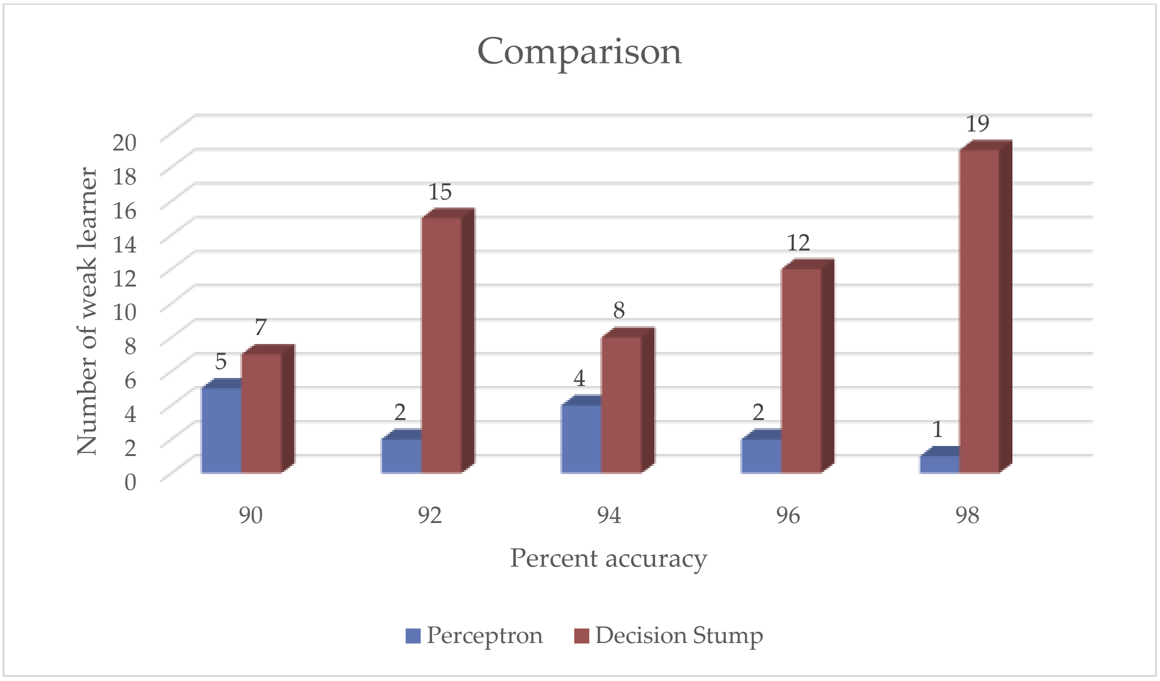
Below are the 10 iteration and number of weak classifiers created.

```
Iteration- 19
Iteration- 8
Iteration- 19
Iteration- 19
Iteration- 15
Iteration- 7
Iteration- 16
Iteration- 12
Iteration- 19
Iteration- 19
```

Accuracy- [98.1651376146788898, 94.495412844036693, 98.165137614678898, 94.495412844036693, 90.825688073394488, 94.495412844036693, 95.412844036697251, 95.412844036697251, 96.330275229357795, 99.082568807339456]

Comparison between the two:

Considering the experiments carried out, at times, decision stumps seems to be weaker than the perceptron. Number of weak classifiers per model are less in perceptron i.e. model with 2 perceptron gives same performance as model with 15 decision stumps. This may depends on external factors like- Pseudo-inverse to calculate the initial weights.



The above graph is approximated as for simplicity. Below are the actual results.

For Perceptron:
Min Accuracy- 91.7431192661
Average Accuracy- 94.5871559633
Max Accuracy- 97.247706422

For Decision Stumps:
Min Accuracy- 93.5779816514
Average Accuracy- 96.9724770642
Max Accuracy- 100.0
The results may vary as data is picked randomly.