Homework 2

Rajiv Anisetti

October 30, 2018

1 Decision Trees

1.1 Constructing Decision Tree

In order to create the decision tree, I utilized the program from problem 1.2 to recursively create the tree, and I printed out each info gain and attribute name for each split. The resulting output, consisting of the info gains and the tree, is shown below. For the attribute values, I used the following convention: in order to implement *YES*, I chose the 1 value. In order to implement *NO*, I chose the 0 value. The leaf nodes' value correspond to the following: **1 for democrat, 0 for republican.**

```
Attribute Selection Criterion:  0
Attr: vote for handicapped-infants , Gain Info: 0.12451124978365313
Attr: vote for water-project-cost-sharing , Gain Info: 0.02904940554533142
Attr: vote for budget-resolution , Gain Info: 0.39731260974948646
best_feature is:  vote for budget-resolution
Attr: vote for handicapped-infants , Gain Info: 0.02008994082044896
Attr: vote for water-project-cost-sharing , Gain Info: 0.06991005367674302
best_feature is:  vote for water-project-cost-sharing
Attr: vote for handicapped-infants , Gain Info: 0.048415675908885514
best_feature is:  vote for handicapped-infants
Attr: vote for handicapped-infants , Gain Info: 0.012509167646966857
Attr: vote for water-project-cost-sharing , Gain Info: 0.012509167646966857
best_feature is:  vote for handicapped-infants
Attr: vote for water-project-cost-sharing , Gain Info: 0.31127812445913283
best_feature is:  vote for water-project-cost-sharing
Attr: vote for water-project-cost-sharing , Gain Info: 0.0760098536627829
best_feature is:  vote for water-project-cost-sharing
{'vote for budget-resolution': {0: {'vote for water-project-cost-sharing': {0.0: 0.0,
                                                                            1.0: {'vote for handicapped-infants': {0.0: 0.0,
                                                                                                                   1.0: 0.0}}}},
                                1: {'vote for handicapped-infants': {0.0: {'vote for water-project-cost-sharing': {0.0: 1.0,
                                                                                                                   1.0: 0.0}},
                                                                     1.0: {'vote for water-project-cost-sharing': {0.0: 1.0,
                                                                                                                   1.0: 1.0}}}}}}
[]
```

1.2 Coding

After implementing the function necessary, I ran the decision tree file using gain info to create the decision tree, and then using gain ratio. The output of each respectively are as follows (next page).

```
Rajivs-MBP:DecisionTree rajivanisetti$ python3 ./DecisionTree.py 0
Attribute Selection Criterion:  0
best_feature is:  legs
best_feature is:  fins
best_feature is:  toothed
best_feature is:  eggs
best_feature is:  hair
best_feature is:  hair
best_feature is:  toothed
best_feature is:  aquatic
{'legs': {0: {'fins': {0.0: {'toothed': {0.0: 7.0, 1.0: 3.0}},
                       1.0: {'eggs': {0.0: 1.0, 1.0: 4.0}}}},
          2: {'hair': {0.0: 2.0, 1.0: 1.0}},
          4: {'hair': {0.0: {'toothed': {0.0: 7.0, 1.0: 5.0}}, 1.0: 1.0}},
          6: {'aquatic': {0.0: 6.0, 1.0: 7.0}},
          8: 7.0}}
Test accuracy:  0.8571428571428571
Rajivs-MBP:DecisionTree rajivanisetti$ python3 ./DecisionTree.py 1
Attribute Selection Criterion:  1
best_feature is:  feathers
best_feature is:  backbone
best_feature is:  airborne
best_feature is:  predator
best_feature is:  milk
best_feature is:  fins
best_feature is:  legs
{'feathers': {0: {'backbone': {0.0: {'airborne': {0.0: {'predator': {0.0: 6.0,
                                                                     1.0: 7.0}},
                                                  1.0: 6.0}},
                               1.0: {'milk': {0.0: {'fins': {0.0: {'legs': {0.0: 3.0,
                                                                            4.0: 5.0}},
                                                             1.0: 4.0}},
                                              1.0: 1.0}}}},
              1: 2.0}}
Test accuracy:  0.8095238095238095
```

It is interesting to note that our gain ratio method results in a lower test accuracy than gain info. This may be because gain ratio is used in order to reduce bias towards multivalued attributes. In our dataset, most of our attributes only take on binary values, so our information gain method does not induce much bias. Thus, I would use the information gain method as an attribute selection method.

2 Support Vector Machines

a) The support vectors are those whose alpha values are nonzero. Thus, data points 2, 6, and 18 contain our support vectors, which correspond to:

$$\left\langle 0.91, 0.32 \right\rangle, \left\langle 0.41, 2.04 \right\rangle, \left\langle 2.05, 1.54 \right\rangle$$

b) The normal vector to the hyperplane is

$$\mathbf{w} = \left\langle -1.34, -0.39 \right\rangle,$$

c) The bias calculated using the formula is $\mathbf{b} = 0.7814 + 0.7817 + 0.7825 = \mathbf{2.3456}$.

d) The correct values can now be substituted into the decision boundary function. The function is as follows:

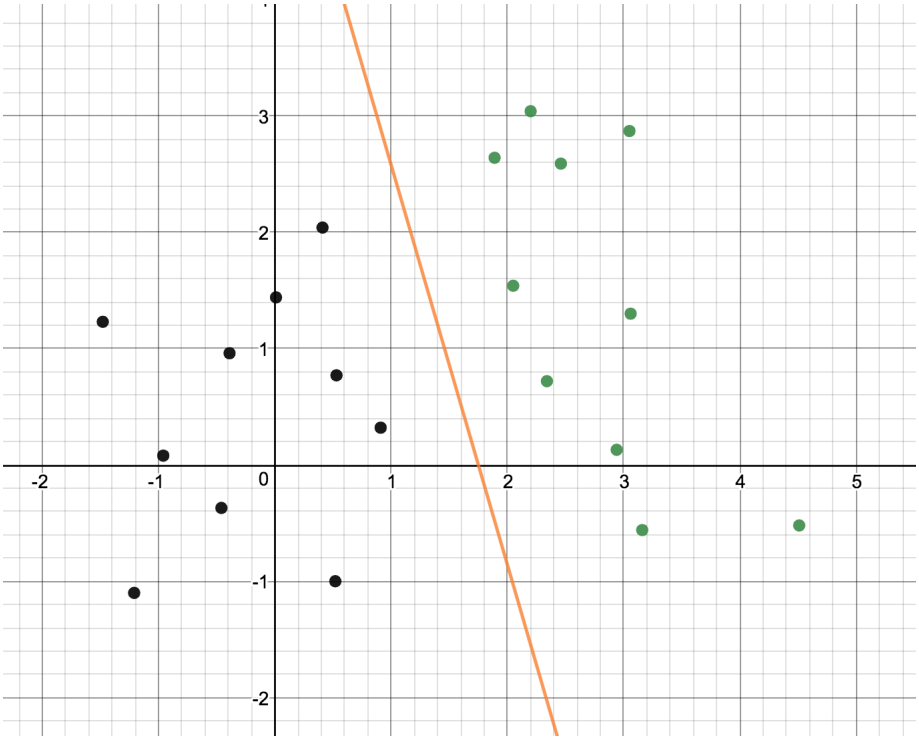
$$f(x) = -1.34x_1 - 0.39x_2 + 2.3456$$

e) Substituting these x values into our decision boundary function, we get

$$-1.34 * -1 - 0.39 * 2 + 2.3456 = 2.91$$

Because this value is $1, this data point can be assigned to class 1$.

f) The plot of this graph is as follows. The green dots represent our -1 class, and the black dots represent the 1 class.



2.1 Coding

The test accuracy for hard margin SVM with linear classifier is 0.5547. This is due to the fact that our data set is not linearly separable.

The test accuracy for soft margin SVM with linear classifier is 0.9890. As we can see, when we use the soft margin approach, we can achieve a much higher accuracy. This is because we can replot some noisy data points to achieve a better model.

The test accuracy for for soft margin SVM with linear kernel, polynomial kernel, and gaussian kernel are 0.989, 0.927, and 1 respectively.

Obviously, our gaussian kernel is our most accurate classifier, but it is very inefficient. Our data is well plotted under linear classification as well, and because our data set is not linearly separable, it is better to use soft margin. Thus, I would choose soft margin linear classification, as it exhibits good results without hindering performance. The gaussian implementation could be better if faster.