Name: Garvit Pugalia
UID: 504628127

# CS 145 - Homework #2

1.1. Constructing a decision tree from congressional voting records dataset. I have divided the problem into different branches. For each branch, we calculate the information gain for each candidate feature and choose the best one before moving on. First, we create a basic frequency table with the information provided:

| Class | Feature A: Vote for handicapped-infants? | | Feature B: Vote for water-project-cost-sharing | | Feature C: Vote for budget-resolution-adoption | |
|---|---|---|---|---|---|---|
| | Y | N | Y | N | Y | N |
| Democrats | 6 | 4 | 4 | 6 | 9 | 1 |
| Republicans | 2 | 8 | 6 | 4 | 2 | 8 |
| Total votes: | 8 | 12 | 10 | 10 | 11 | 9 |

Name: Garvit Pugalia
UID: 504628127

Branch 1:

There are equal samples of Democrats and Republicans. Therefore:

total entropy = Entropy(10, 10) = Entropy(0.5, 0.5) = 1

(a) If we choose Feature A as the splitting feature, we get:

conditional entropy = (8/20) * Entropy(6, 2) + (12/20) * Entropy(4,8)
= 0.4 * Entropy(0.75, 0.25) + 0.6 * Entropy(0.33, 0.67)
= 0.4 * 0.81 + 0.6 * 0.92
= 0.876

information gain = total entropy – conditional entropy
= 1 – 0.876 = **0.124**

(b) If we choose Feature B as the splitting feature, we get:

conditional entropy = (10/20) * Entropy(4, 6) + (10/20) * Entropy(6,4)
= 0.5 * Entropy(0.4, 0.6) + 0.5 * Entropy(0.6, 0.4)
= Entropy(0.6, 0.4)
= 0.971

information gain = total entropy – conditional entropy
= 1 – 0.971 = **0.029**

(c) If we choose Feature C as the splitting feature, we get:

conditional entropy = (11/20) * Entropy(9, 2) + (9/20) * Entropy(1,8)
= 0.55 * Entropy(0.818, 0.182) + 0.45 * Entropy(0.111, 0.889)
= 0.55 * 0.68 + 0.45 * 0.50
= 0.599

information gain = total entropy – conditional entropy
= 1 – 0.599 = **0.401**

Clearly, the information gain is highest when splitting on “Vote for budget-resolution-adoption”, which will become our first decision node.

Name: Garvit Pugalia
UID: 504628127

<u>Branch 2:</u>

After splitting on Feature C, we need to update our frequency table and divide into two subtables.

Voted 'Yes' for budget-resolution-adoption:

| Class | Feature A: Vote for handicapped-infants? | | Feature B: Vote for water-project-cost-sharing | |
|---|---|---|---|---|
| | Y | N | Y | N |
| Democrats | 6 | 3 | 4 | 5 |
| Republicans | 1 | 1 | 1 | 1 |
| Total votes: | 7 | 4 | 5 | 6 |

Voted 'No' for budget-resolution-adoption:

| Class | Feature A: Vote for handicapped-infants? | | Feature B: Vote for water-project-cost-sharing | |
|---|---|---|---|---|
| | Y | N | Y | N |
| Democrats | 0 | 1 | 1 | 0 |
| Republicans | 1 | 7 | 5 | 3 |
| Total votes: | 1 | 8 | 6 | 3 |

<u>*Subtable 1*</u>:

There are 9 Democrats and 2 Republicans. Therefore:

total entropy = Entropy(9, 2) = Entropy(0.818, 0.182) = 0.684

(a) If we choose Feature A as the splitting feature, we get:

conditional entropy = (7/11) * Entropy(6, 1) + (4/11) * Entropy(3, 1)
= 0.636 * Entropy(0.857, 0.143) + 0.364 * Entropy(0.75, 0.25)
= 0.636 * 0.592 + 0.364 * 0.81
= 0.671

information gain = total entropy – conditional entropy
= 0.684 – 0.671 = **0.013**

Name: Garvit Pugalia
UID: 504628127

(b) If we choose Feature B as the splitting feature, we get:

conditional entropy = (5/11) * Entropy(4, 1) + (6/11) * Entropy(5, 1)
= 0.455 * Entropy(0.8, 0.2) + 0.545 * Entropy(0.833, 0.167)
= 0.455 * 0.722 + 0.545 * 0.651
= 0.683

information gain = total entropy – conditional entropy
= 0.684 – 0.683 = **0.001**

Therefore, we will split Subtable 1 using Feature A.

*Subtable 2:*

There are 1 Democrat, and 8 Republicans. Therefore:

total entropy = Entropy(1, 8) = Entropy(0.111, 0.889) = 0.503

(a) If we choose Feature A as the splitting feature, we get:

conditional entropy = (1/9) * Entropy(0, 1) + (8/9) * Entropy(1, 7)
= 0.111 * Entropy(0, 1) + 0.889 * Entropy(0.125, 0.875)
= 0.889 * 0.544
= 0.484

information gain = total entropy – conditional entropy
= 0.503 – 0.484 = **0.019**

(b) If we choose Feature B as the splitting feature, we get:

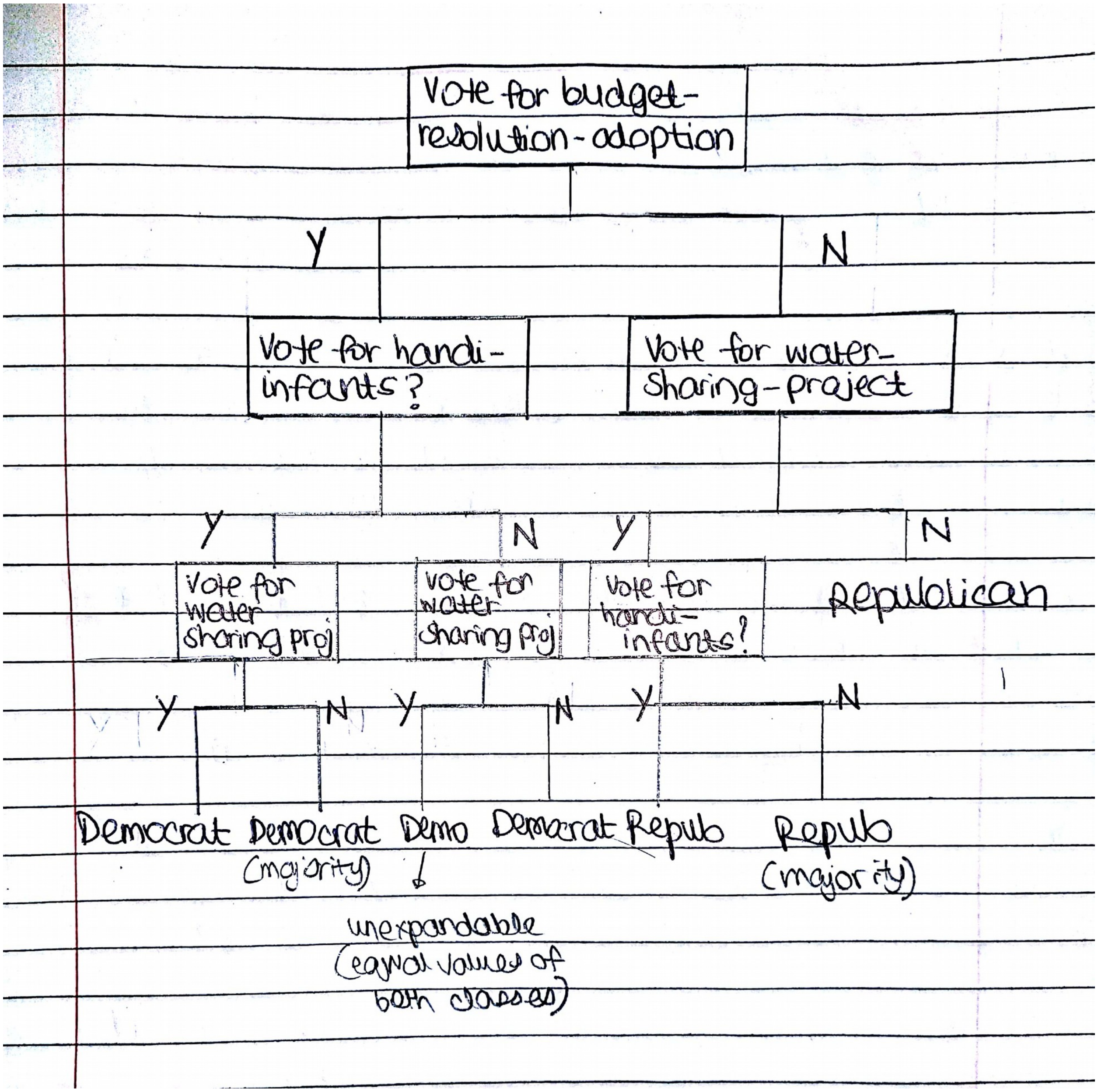
conditional entropy = (6/9) * Entropy(1, 5) + (3/9) * Entropy(0, 3)
= 0.667 * Entropy(0.167, 0.833) + 0.333 * Entropy(0, 1)
= 0.667 * 0.651
= 0.434

information gain = total entropy – conditional entropy
= 0.503 – 0.434 = **0.069**

Therefore, we will split Subtable 2 using Feature B.

Name: Garvit Pugalia
UID: 504628127

Since we have only one Feature left, we can go ahead and make the Decision Tree according to the decision nodes found upto now. Any non-leaf nodes are expanded with the last Feature remaining, and majority is used to classify the leaf. If majority doesn't exist, the node is labelled 'unexpandable'.



Name: Garvit Pugalia
UID: 504628127

1.2.

For information gain, we get the following tree and accuracy:

```
garvit@garvit-Inspiron-13-7359:~/Desktop/CS145/HW2/HW2/HW2 Programming/DecisionTree$ python 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: array([7.]), 1.0: array([3.])}},
                       1.0: {'eggs': {0.0: array([1.]), 1.0: array([4.])}}}},
          2: {'hair': {0.0: array([2.]), 1.0: array([1.])}},
          4: {'hair': {0.0: {'toothed': {0.0: array([7.]), 1.0: array([5.])}},
                       1.0: array([1.])}},
          6: {'aquatic': {0.0: array([6.]), 1.0: array([7.])}},
          8: array([7.])}}
Test accuracy:  0.8571428571428571
```

For gain ratio, we get the following tree and accuracy:

```
garvit@garvit-Inspiron-13-7359:~/Desktop/CS145/HW2/HW2/HW2 Programming/DecisionTree$ python 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: array([6.]),
                                                                     1.0: array([7.])}},
                                                  1.0: array([6.])}},
                               1.0: {'milk': {0.0: {'fins': {0.0: {'legs': {0.0: array([3.]),
                                                                            4.0: array([5.])}},
                                                             1.0: array([4.])}},
                                              1.0: array([1.])}}}},
              1: array([2.])}}
Test accuracy:  0.8095238095238095
garvit@garvit-Inspiron-13-7359:~/Desktop/CS145/HW2/HW2/HW2 Programming/DecisionTree$
```

There are a number of differences in the two trees. When using information gain, the maximum depth in the tree is 3 (legs – fins – toothed, for example) whereas the maximum depth in the gain ratio tree is 5 (feathers – backbone – milk – fins – legs). The tree for information gain is also more symmetric, where the five subtrees have almost equal structures for the feature ‘legs’. While using gain ratio, the tree seems to have a skewed structure with one subtree of ‘feathers’ having depth 1 and the other having depth 4. In addition to these differences, the accuracy for information gain is 0.857, which is greater than 0.809 for gain ratio. Therefore, for improved accuracy and a simpler decision tree, I would pick **information gain** as the splitting factor.

Name: Garvit Pugalia
UID: 504628127

2.1.

(a) Since there are three non-zero values of a, we have three support vectors. These correspond to the number following the non-zero a's – 2, 6, and 18. Therefore, the support vectors are points 2, 6, and 18.

| Point | X1 | X2 | Class (y) |
|---|---|---|---|
| 2 | 0.91 | 0.32 | 1 |
| 6 | 0.41 | 2.04 | 1 |
| 18 | 2.05 | 1.54 | -1 |

(b) The weight vector can be calculated using the Lagrange multipliers and the support vectors.

| Support vector | *y* . ***a*** (Lagrange) |
|---|---|
| (0.91, 0.32) | 1 x 0.5084 = 0.5084 |
| (0.41, 2.04) | 1 x 0.4625 = 0.4625 |
| (2.05, 1.54) | -1 x 0.9709 = -0.9707 |

Therefore, the weight vector is:

w = 0.5084 (0.91, 0.32) + 0.4625 (0.41, 2.04) – 0.9707 (2.05, 1.54)
= (-1.3377, -0.3887)

(c) We use the equation provided in the question. Let's first calculate the RHS for each individual non-zero ***a*** and end with the summation.

| Support Vector | y | w | RHS |
|---|---|---|---|
| (0.91, 0.32) | 1 | (-1.3377, -0.3887) | 1 - (-1.341691) = 2.341691 |
| (0.41, 2.04) | 1 | | 1 - (-1.341405) = 2.341405 |
| (2.05, 1.54) | -1 | | -1 - (-3.340883) = 2.340883 |

Therefore, b = (2.341691 + 2.341405 + 2.340883) / 3
= 2.3413

Name: Garvit Pugalia
UID: 504628127

(d) Therefore, we can finally write the learned decision boundary function as:

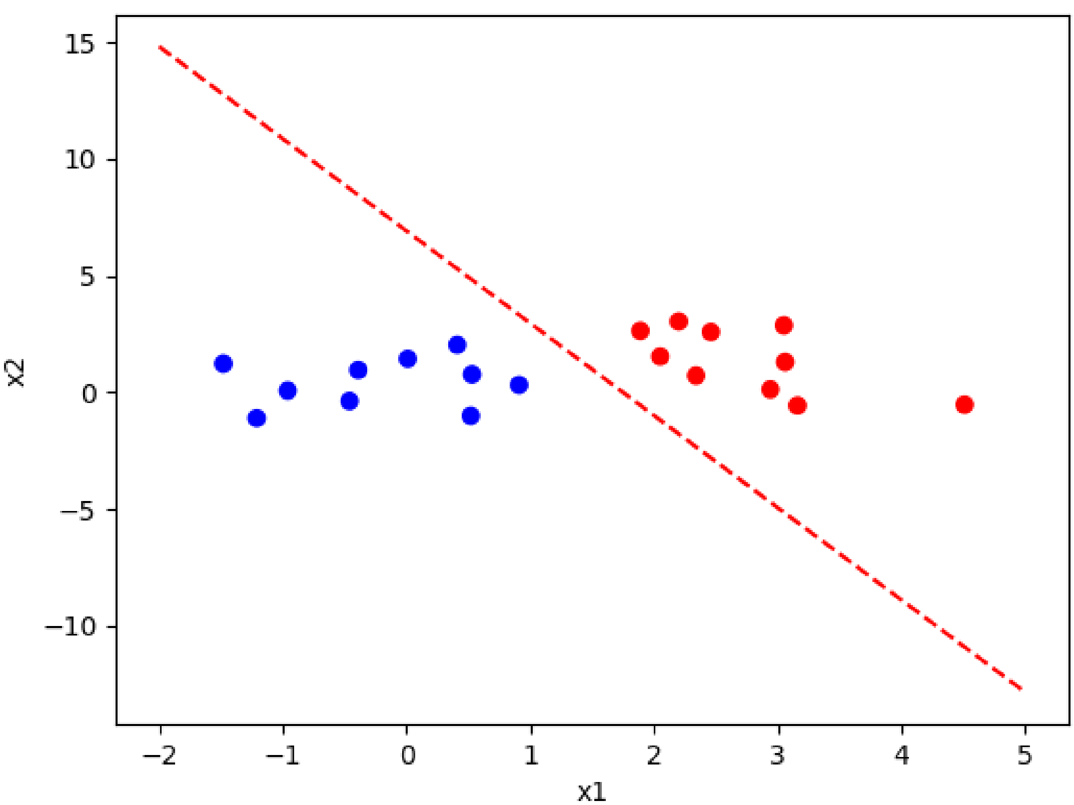
$$f(x) = (-1.3377, -0.3887)^{T}x + 2.3413$$

(e) With x = (-1, 2):

$$f(x) = (-1.3377, -0.3887)^{T} . (-1, 2)^{T} + 2.3413$$
$$= 2.9016$$

Therefore, the class will be 1.

(f) The plot has blue points to represent +1 class, and red points to represent -1 class. The dotted line is the decision boundary.



Name: Garvit Pugalia
UID: 504628127

2.2.

After running svm.py with different arguments, we get the following results:

| Kernel | Margin | Number of support vectors | Solution found? | Accuracy |
|---|---|---|---|---|
| Linear | Hard | N/A | Terminated | 55.47% |
| Linear | Soft | 34 | Optimal | 98.91% |
| Polynomial | Soft | 19 | Optimal | 92.70% |
| Gaussian | Soft | 35 | Optimal | 100.0% |

We would prefer the Gaussian model as it provides a 100% accuracy. While it is not as time-efficient as the linear solution (which also has a comparable 98.91%), the gain in accuracy is definitely worth the lost time. Correspondingly, it finds the most number of support vectors, increasing the dimensionality of the hyperplane.