Mechine Learning - Assignment 3

Michael Dadush 206917908

Shay Gali 315202242

February 2025

1 K-Nearest Neighbors Classifier

Based on our experiment results with different k-nearest neighbor parameters, we can make important observations about the classifier performance. We will analyze the results for different k and p values.

1.1 Code Output

This is the code output for different k and p values:

```
    round       | Average Empirical Errors  |  Average True Errors  |  Error Differences
p = 1   k = 1   |         0.014600          |       0.130800        |      0.116200
p = 1   k = 3   |         0.055800          |       0.093800        |      0.038000
p = 1   k = 5   |         0.060400          |       0.090000        |      0.029600
p = 1   k = 7   |         0.063800          |       0.090000        |      0.026200
p = 1   k = 9   |         0.065400          |       0.090000        |      0.024600
                |                           |                       |
p = 2   k = 1   |         0.014800          |       0.126200        |      0.111400
p = 2   k = 3   |         0.054400          |       0.092000        |      0.037600
p = 2   k = 5   |         0.062800          |       0.085400        |      0.022600
p = 2   k = 7   |         0.064200          |       0.085200        |      0.021000
p = 2   k = 9   |         0.063600          |       0.084400        |      0.020800
                |                           |                       |
p = inf k = 1   |         0.014800          |       0.136200        |      0.121400
p = inf k = 3   |         0.056600          |       0.090400        |      0.033800
p = inf k = 5   |         0.062400          |       0.085200        |      0.022800
p = inf k = 7   |         0.063000          |       0.086400        |      0.023400
p = inf k = 9   |         0.065400          |       0.084200        |      0.018800
```

1.2 Best Parameters

From looking at the data results, we found that the best parameters are:

- p = 2 with k = 9 (error rate = 0.0844 or 8.44%)
- Also good is p = ∞ with k = 9 (error rate = 0.0842 or 8.42%)

These parameters give us the lowest Average True Errors, which is most important for real performance.

1.3 Analysis of k Parameter

When we look at how k affects the results:

- k = 1 gives worst performance on test data for all p values
- When k gets bigger, the true error usually gets smaller
- This shows us that using more neighbors helps make better predictions

1.4 Analysis of p Parameter

For the distance metric p:

- p = 1 does not work as good as p = 2 or p = ∞
- p = 2 (Euclidean) and p = ∞ work almost same, but p = 2 is little better
- We think this shows that using regular distance (p = 2) works better for our data than other options

1.5 Overfitting Analysis

We see clear overfitting in our results, especially with k = 1:

For k = 1:

- Very small empirical errors ($\approx$ 0.014-0.015)
- Much bigger true errors ($\approx$ 0.126-0.136)
- Big differences between errors ($\approx$ 0.111-0.121)

When k gets bigger:

- Empirical errors get little bigger
- True errors get much smaller
- Differences between errors get smaller too

This pattern shows classic overfitting when k = 1. The model learns training data too well (small empirical error) but cannot work good on new data (big true error). Using bigger k helps fix this by taking average of more neighbors.

1.6 Why These Parameters Work Best

k = 9 with p = 2 gives best results because:

- It uses enough neighbors to make predictions more stable
- It reduces overfitting (smallest difference between empirical and true errors)
- Euclidean distance (p = 2) seems to work better for relationships between features than Manhattan (p = 1) or maximum distance ($p = \infty$).
- Euclidean distance represents how the human eye sees distances, which is good for our data
- The big k value (k = 9) tells us there is probably lot of noise in data that needs to be averaged

1.7 Conclusion

Our analysis shows that k = 9 and p = 2 give best performance for this classification task. These parameters help balance between learning from data and not overfitting. The results also show importance of choosing right k value to avoid overfitting, especially avoiding k = 1 which memorizes training data too much.

2 Decision Tree Classifier

We tested two ways to build decision trees: brute-force and binary entropy. Both methods were limited to 3 levels (k=3). We used the Iris dataset and focused on separating Iris-versicolor (marked as 0) from Iris-virginica (marked as 1).

2.1 Results

Our accuracy results were:

- Brute force method: 95%
- Binary entropy method: 93%

2.2 The Trees We Got

You can see both trees in Figures 1 and 2.

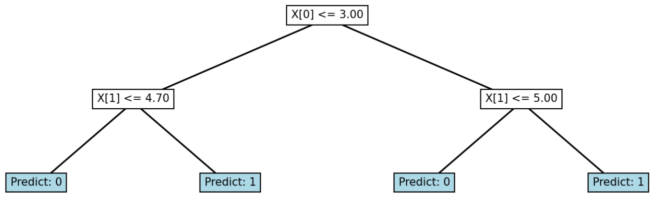

Figure 1: The tree from brute force method

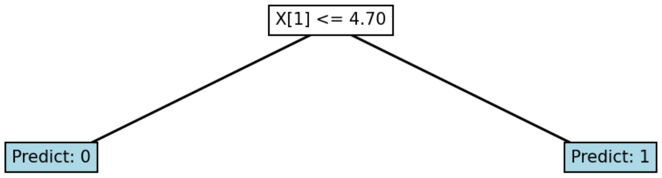

Figure 2: The tree from entropy method

2.2.1 Analyze the results

Let's look at what each method did:

Brute Force Method

- Got better accuracy (95%)
- Used all 3 levels we allowed - need both features to make the decision (the VC dimension is 2)
- Made these splits:
  - First split: X[0] $\leq$ 3.00
  - Left side: X[1] $\leq$ 4.70
  - Right side: X[1] $\leq$ 5.00

  Where X[0] is the second feature and X[1] is the third feature from the Iris dataset

Binary Entropy Method

- Got 93% accuracy
- Used just two level even though it could use three - we need only one feature to make the decision
- Made one split: X[1] $\leq$ 4.70

Comparing the Methods The entropy method made a much simpler tree and still worked almost as well (only 2% worse). This tells us:

- The simpler tree might work better on new data
- X[1] is probably the more important feature
- The brute force tree might be too complex for what we need

For real use, we think the entropy method is better because it's simpler and faster, even though it got slightly lower accuracy. The brute force method might be trying too hard to fit our specific data.