HW 1 Report

Jack Stevenson

What is the best number of neighbors (k) you observe?

- I found that in general, 9 or 99 was usually the most accurate value of k. This makes sense, as the dataset is large enough that smaller k values might not be dominated by either label.

When k = 1, is training error 0%? Why or why not?

- When k is 1, training error is around 1% for most runs. I'm not certain where this error originates – the algorithm should be able to access every single entry in the dataset, except for k-fold cross validation. Validation error for k-fold is very high, as the point can only be in one of the four sections.

What trends (train and cross-validation accuracy rate) do you observe with increasing k?

- As k increases to a point, both train and cross-validation accuracy increase. After than point, both decrease. Peak accuracy usually occurred around k = 9 or k = 99.

How do they relate to underfitting and overfitting?

- Having an unreasonably low k (ie 1) vastly overfits the data and usually produces a training accuracy of 100%. Having an unreasonably high k (ie 8000) underfits the data such that every point will receive the label of the dominant type in the dataset. This means that every point in the test set will receive a label of 0, with about 75% accuracy. For RPLSH, the training time greatly increased for k = 8000 (from 40 seconds to 16 minutes).

What are the best configurations of M , L and k that you found?

- The best combination of M, L, and k was usually around M=2, L=2, k=99. Based on the random hyperplanes, these parameters vary somewhat, but there was a sharp drop off in accuracy as M increases past 2 or 4.

What are the training and test accuracies with this best configuration? Are there any trends in accuracy that you noticed for M and L?

- I found that increased M led to increased speed and decreased accuracy. Increased L led to decreased speed and increased accuracy. These are both intuitive trends.

Debriefing

1. Approximately how many hours did you spend on this assignment?

- I probably spent 15 or 20 hours on this assignment. A lot of it was trying to plan out the pseudocode for what the program would do. I also hit a wall with the hyperplane implementation, and had to spend a few hours figuring out the code structure.

2. Would you rate it as easy, moderate, or difficult?

- I would rate this as somewhat difficult, solely because of the complexity of developing the program. The concepts are not too hard to figure out, but I struggled at the beginning to understand what the actual code implementation would look like. Everything worked out fine, but it took a little longer than I thought it would.

3. Did you work on it mostly alone or did you discuss the problems with others?

- I mostly worked on the assignment alone. I offered some high-level help to two classmates.

4. How deeply do you feel you understand the material it covers (0% - 100%)?

- I feel that I understand the fundamental concepts very deeply, but there are a few particular strategies that I was unsure about when doing the assignment. For instance, I was unsure whether or not a RPLSH nearest neighbors call should consider the adjacent hash buckets if there are fewer than k values in the current hash bucket. I implemented that, and it led to an increase in hashing time but a significant increase in accuracy.

5. Any other comments?

- In my code, RPLSH at 8000 is incredibly slow, so if it looks to be hanging, it probably isn't.
- I was intimidated by the complexity of the project at first, but two weeks was enough time to figure it out comfortably.