CS 361 final project

ss60

April 2019

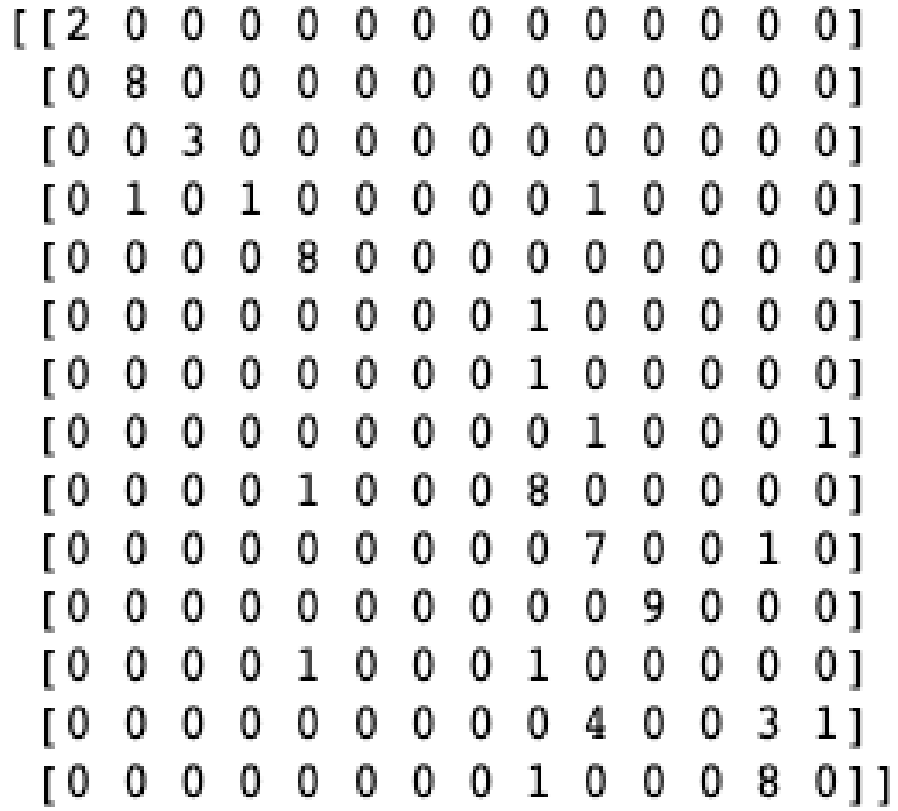
Note to grader: I had to partition the data into training, validation and testing sets manually as I wanted to split into 80 percent training and certain folders only had 3 or 5 files in them. I have attached the data folder along with the code submission. Thank you for your understanding.

1 Initial Error Rate and Confusion Matrix

initially trained on d = 36, k = 400

error rate = 32.877 percent

confusion matrix:



2 Finding optimal values for d and k

At first, I loaded the data from all the files and split the data into n sets of (d times 3) dimensional vectors, where d represents the time in seconds which corresponds to a segment of 3-dimensional spatial vectors. After that, I trained the kmeans model with the vectors to get a set of k cluster centers. Then I converted each file into a k sized vector that corresponded to the frequency of vectors that were closest to each cluster center. I then trained a random forest model to classify the 13 actions given a k dimensional feature vector. At first, I used a d value of 36 and a k value of 400 that yielded a 66 percent accuracy.

To obtain the ideal values of k and d I used an approach more oriented towards calculus as I viewed k, d and the accuracy as a 3-dimensional function to be optimized. I first split the files into training, validation and testing sets with around 80 percent allocated to training and around 10 percent allocated to validation and testing. I did this to prevent over-fitting as although certain values of d and k might be optimal on such a large training set, its accuracy

may be skewed when perhaps a new user submits their data as the model is biased towards one user.

Given a particular d and k value, I generated a kmeans model on the training files and simultaneously used the resulting feature vectors to train a random forest classifier. I then used the same kmeans model to convert the validation files into feature vectors and then computed the accuracy which measured how well the model was able to predict the validation labels given that particular d and k value.

Given this organized structure, I visualized the problem as a 2-dimensional plane of choices of k and d each with its corresponding accuracy. I then tried extreme values of k and d; for instance for any given value of d with the combination of k of 200 and above yielded very low accuracy, and any value of d above 50 also yielded low precision.

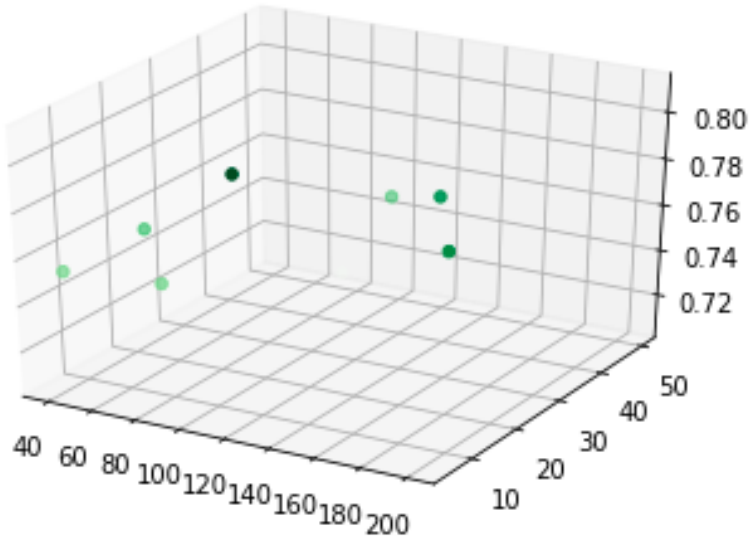
For each value of d for example, there was an array of possible values of k. I fixed a range of reasonable values of k which I set from 50 to 400. For a given d-value, I computed the lower bound accuracy (lower bound value of k) and the upper bound accuracy (upper bound value of k) and if the lower bound was greater then the midpoint is updated to be the upper bound (essentially the range was split in half). I originally wanted to use every 20th value of k in a linear search fashion, however, this process took too long and my laptop was inept at handling such a large array of computations. The range of k values then converges after a certain number of iterations and the difference between the accuracy of the upper bound and the lower bound reduces and it can be assumed that the midpoint of the two bounds is the optimal value of k for that particular value of d.

Now for each value of d there is a separate array of accuracies for each value of k as seen in the graph as each value of d corresponds to its own accuracy curve; this can be thought of as setting one variable constant in a 3-dimensional function and examining the relationship between the other variable and the accuracy. Therefore, for each value of d, a different convergence of the values of k may occur, and hence the combination with the highest accuracy is the most optimal combination of k and d.

I repeated this process again but instead, I fixed values of k and varied the ranges of convergence of d (I used a range of 5 to 40 for d). I assumed that there may be different global maximum accuracy levels when I fix k and when I fix d and hence I take the maximum accuracy in both directions and use the

corresponding d and k values as the optimal parameters.

Generally, the iterative process I used favored lower values of d and k where there was increased accuracy on the validation set; this makes intuitive sense as at times there was only 1 feature vector per label due to the limited number files and a lower cluster size would yield a more descriptive feature vector. the optimal parameters I received was d = 6 and k = 40. The accuracy values however kept changing each time I ran the classifier - the average accuracy of the classifier was 75.3 percent



graph of values of k and d with the corresponding accuracy on the validation set. Includes 8 data points.

3 Error Rate and Confusion Matrix After Selecting Optimal Parameters

Error rate = 24.657 percent

Confusion matrix:

```
[[2 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 8 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 3 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 3 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 8 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 1 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 1 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 2]
 [0 0 0 0 0 0 0 0 9 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 6 0 0 2 0]
 [0 1 0 0 0 0 0 0 0 0 8 0 0 0]
 [0 0 0 0 0 0 0 0 1 0 0 1 0 0]
 [0 0 0 0 0 0 0 0 0 2 0 0 5 1]
 [0 1 0 0 0 0 0 0 0 4 0 0 3 1]]
```