STAT 376 PROJECT 1 REPORT

MICHELLE YEO AND ZHENGJIAN SONG

Exercise 0.1. (Question 4)

Roughly speaking, given the same k, the kNN classifiers perform better as the number of training examples increase. Still for each size of training set, we choose the best k, which has least error rate on average, and then compare the best error rate that the classifiers can achieve.

For the our pairs, cars and bicycles, after 50 times of random sampling the training set from the data set, the best k we got for the training sizes of 10, 20, 30, 40 and 50 are 7, 15, 21, 21, 15, and the corresponding mean error rates are 0.2288, 0.1952, 0.1838, 0.1756, 0.1696.

We repeated this procedure on a few other pairs of data. For instance, for the pair of bananas and starfish, we got best k values of 1, 3, 5, 9, 9 and error rates of 0.1834, 0.1375, 0.1182, 0.1123, and 0.1015 for the respective training sizes. For the pair of brains and pumpkins, we got best k values of 1, 5, 5, 7, 7 and error rates of 0.3314, 0.2761, 0.2577, 0.2304, and 0.2242 for the respective training sizes. For the pair of deserts and forests, we got best k values of 9, 7, 11, 45, and 11 and error rates of 0.2148, 0.2017, 0.1905, 0.1865, and 0.1843. For the pair of dolphins and tomatoes, we got best k values of 3, 3, 1, 7, and 1, and error rates of 0.088, 0.0798, 0.0714, 0.0693, and 0.0672. For the pair of bears and rainbows, we got best k values of 5, 3, 11, 7, and 3 and error rates of 0.2404, 0.2062, 0.2026, 0.1940, and 0.1834.

The results were expected. As the size of the training set increases, the error rate decreases, which can be empirically justified from all the examples we tried our kNN classifier on. As for the comparison of the overall error rates, it appears that pairs which look pretty dissimilar have very low error rates (eg. dolphins and tomatoes, bears and rainbows) while pairs which look pretty similar or share similar features (eg. brains and pumpkins, cars and bikes) have higher error rates.

Exercise 0.2. (Question 5)

For our results, we for k = 19. For the pair of bananas and starfish, we got k = 19. For the pair of brains and pumpkins, we got k = 23. For the pair of deserts and forests, we got k = 26. For the pair of dolphins and tomatoes, we got k = 23. For the rainbows and bears, we got k = 20. As required, we only chose k for some sub-collections. The k s here are different between sub-collection. This is due to the different distributions of gist distances between images in each sub-collection.

Exercise 0.3. (Question 6)

For our results, we got a mean error rate of 0.3119, 0.2156, 0.1911, 0.1971, and 0.196 for the training sizes of 10, 20, 30, 40 and 50 respectively. For the pair of bananas and starfish, we get mean error rates of 0.2267, 0.1622, 0.1389, 0.1154,

and 0.115 respectively. For the pair of brains and pumpkins, we get mean error rates of 0.4572, 0.3581, 0.2768, 0.2663, and 0.2455 respectively. For the deserts and forests, we get mean error rates of 0.2695, 0.2087, 0.2161, 0.2001, and 0.1950. For the dolphins and tomatoes, we got mean error rates of 0.0783, 0.07625, 0.0726, 0.0614, and 0.06933. For the rainbows and bears, we got the mean error rates of 0.2541, 0.2138, 0.2163, 0.2022, and 0.2003.

Note that the tuning parameter in the gaussian that we use here is 100. One possible way to choose the parameter by cross-validation. What should be carefully is that the optimal tuning parameter here depends heavily on the size of training data. For example, one should not apply the optimal parameter chosen for training size = 10 onto the whole data set, and then predict other test sets.

Again like before, pairs which look pretty dissimilar have very low error rates (eg. bananas and starfish) while pairs which look pretty similar or share similar features (eg. brains and pumpkins) have higher error rates. We also have the rate of error decreasing as the size of the training set increases.

When we compare the performance of this classifier to the kNN classifier, we note that the kNN classifier outperforms this classifier overall. There are perhaps some explanations for this and they can be found in Xiaojin Zhu's dissertation *Semi-supervised learning with graphs* (2005). In the section on harmonic functions, Zhu notes that the 0.5 threshold which we use to see if $f(i)$ should be classified as 1 or 0, makes sense theoretically but is not commonly used in practise, since the data manifold is often too poorly estimated for us to trust the graph structure. The paper also proposes an alternative method to determine the threshold, namely class mass normalisation. We did not implement this into our classifier due to lack of time. Also, the success of this classifier is dependent on the sparsity of the graph and according to the paper, the sparser the graph, the better this classifier performs. The paper also notes that in fully connected graphs the edges between different classes, even with relatively small weights, can create unwarrantedly strong connections across the classes. This means that when we chose k such that 95% of the graph is connected and just connected the other 5% of the unconnected vertices to their nearest neighbour, we could have been adding edges which strengthens the connection between the two classes and consequently lead to a higher rate of misclassification. Also, the smallest k we got which gives us a 95% connected graph was quite large (k=19) and thus this could have added to the density of our graphs and thus an increase in the misclassification rate.