Part1

A

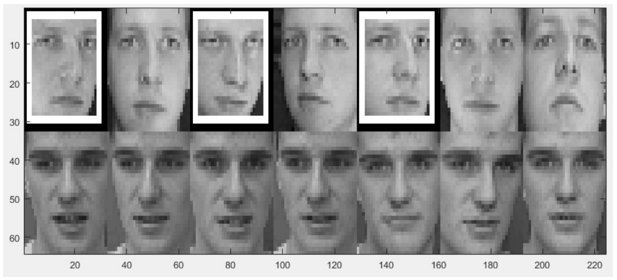
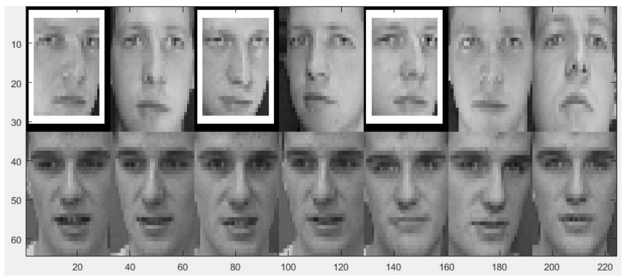
Summary: in this part my goal is to build a k-NN classifier above 2 classes. For each possible k(1 to 6), 2 particular classes are extracted form raw data and used to produce 50 datasets at random. After a set is given, the test set goes through k-NN classifier example by example to give results, correctness of which are checked by actual one and training accuracy is calculated during this. Helpful graphs and statistic results are given after that. All above is for testing accuracy, and that for training accuracy need almost the same. At last, 2 figures are given regarding standard deviation and testing and training accuracy respectively.

Q1. Training accuracy should be 100% when k=1.because in this case the nearest neighbor of training point (or testing point) is the point itself. Training and testing accuracies are different, since testing accuracies calculation uses data outside the training set, whilst training ones uses those inside, which means its accuracy will always be higher by counting itself as one of nearest neighbors.

Q2. Behavior over different dataset should not be the same, since boundaries between classes will have nuances from one iteration to another, so for those samples that are quite close to boundary this time and appear as a testing example, in another loop it is likely to be a training example and participate in drawing another boundary, and chances are that an example in one class in previous loop may be classified as in the other class this time, causing fluctuations in accuracy. Apart from that, noisy training samples also contribute in relatively lower results.

Q3. k should not be an even number, since in this case it's likely that the number of neighbors from either class are the same, causing the k-NN classifier give a random result class between the two (or among many in a multi-class problem). Accuracies differ over different values of k, larger k gets classifier less vulnerable to noisy training examples, yet when k gets too large, let's say, we have 10 examples for each class but set k as 100, extra examples will be disturbance to the classifier then.

Q4

We can see that example in class 1 is harder to classify, as class 1 has quite a few misclassified examples while class 30 have none in 2 graphs above, and this is common among many times of script running. Raw classification results are recorded in res_y_te(14,6) in part1a, from which we know the some testing examples in class 1 are misclassified as examples from class 30, yet those in class 30 are seldom misclassified except when k is 6.

B

Summary: in this part my goal is to practice k-NN classifier above the whole dataset with 40 classes. My k is to be selected from 1 to 10 since an excessively large k means more disturbance on result as we only have 5 examples for training in each class. For each possible k, a k-NN is practiced on the dataset to get an error rate. The minimum one of them is the ideal one and corresponding k is the one we are going to use. Produce 50 datasets at random. On each dataset, practice k-NN with every example from test set using the k we get above to get a series of result label. Compare it with original label in test set so as to get accuracy rate on the whole test set. Accurate result on each individual class is counted as then rate calculated synchronously. Go through the k-NN usage procedure for every set in those 50. Statistic results given at last include averaged precision for each subject, averaged testing accuracy and the averaged standard deviation

Q1. It's not a good idea. Evenly choose examples from all class guarantees that nearest neighbor only depends on distances between training examples and testing examples, yet uneven selection leads to biased training sets in which testing examples are more likely to be classified into classes of which examples make up bigger part in training set. In the worst case, if all examples are from the same class, then all testing data will be classified to that class with no doubt.

Q2. Classification accuracy by subject is given as av_acu_se(1,40). Subjects that is considered the most difficult to recognize by k-NN classifier will have the lowest accuracy among 40 classes, they are class 1(0.5520), 4(0.5800), and 13(0.6520) respectively.

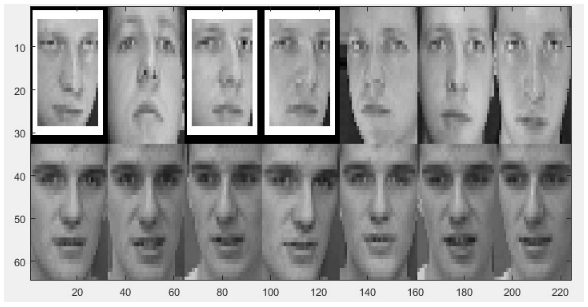
Part2

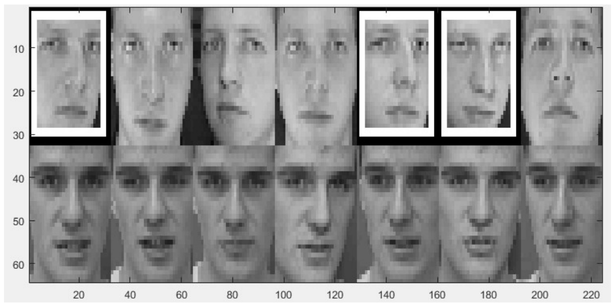
A

Summary: in this part my goal is to practice linear least square approach using normal equation above 2 classes. first we extract class 1 and 30 from the whole set, and randomly divide it to training ones an test ones for 50 times. For each dataset, add one column of 1 on the left to get extended sets of x for 1 dimension. Use training sets of extended matrix Xtr_l and training label vector Ytr to get vector w through pseudo-inverse. Get this w product with every column from test set Xte for result y_th. Put all these results in a vector, determine labels of testing examples according to them then compare with their actual label. Testing accuracy can thus be given. In the same loop for a randomly produces set, testing accuracy can be elicited by the same way. A sample result of classification using examples in testing and training set is given at last.

Q1. Training accuracy of linear classifier should be 1. We should remember that vector w is derived from product of training example matrix and vector, so giving results of training examples is exactly a reversed procedure of getting w, thus they should be completely same and accuracy is 1 then.

Q2.





We can see that example in class 1 is harder to classify, as class 1 has quite a few misclassified examples while class 30 have none in 2 graphs above, and this is common among many times of script running. Raw classification results are recorded in y_th(14,50) in part2a, from which we know the outcome of regression of testing examples in class 1 is more close to boundary, sometimes they even went through it so that misclassified as examples from class 30.

B

Summary: in this part my goal is to practice linear least square approach using normal equation above the whole dataset. For each one loop of 50, separate the whole set, one for training and the other for testing, and extend Xtr as well as Xte in the same way as part2A. Since we are doing multi-class classification, 1-of-K coding scheme should be used here, that is to get a c*N matrix that c for class number and N for example number. Use such a matrix get matrix w, and product it with extended Xte for result 200*40 matrix y_th. For each row that represents a example, the largest number tells us it belongs to this class. Use another 200*40 matrix Yte_lbl to record the same position as 1 and the rest in this row as 0 and record this classification result in a vector lbl_res. After doing so on all 200 examples, check elements in lbl_res by correct label in Yte for overall testing accuracy as well as class-specified accuracy. Compare these results with those given by k-NN method.

Q1. Accuracy of linear classifier is a bit higher than k-NN, as we can see from the result. k-NN has a time complexity of O(n^2), lower than linear classifier with O(n^3), while space complexity of k-NN and linear classifier are both O(n^2).

Q2. Classes with lowest accuracy are 1(0.7240), 3(0.8200) and 13(0.7480) respectively, roughly the same as result in Q2 of part2B.

Name: Haorui Chen, ID:10407315