LING 572 Homework 4

David McHugh and Chris Curtis

5 Feb 2013

Question 1

The script for this question, `build_kNN.sh`, is located in the directory `scripts/`. The source code is in `src/`, and the compiled class files are in `bin/`.

Question 2

The results of running `build_kNN.sh` with the specified parameters are given below:

k	Euclidean distance	Cosine function
1	0.620000	0.720000
5	0.636667	0.666667
10	0.656667	0.633333

Table 1: (Q2) Test accuracy using **real-valued** features

Question 3

The results of running `build_kNN.sh` with `train2.vectors.txt` and `test2.vectors.txt` (i.e. with binary features) are given below:

k	Euclidean distance	Cosine function
1	0.630000	0.823333
5	0.553333	0.813333
10	0.546667	0.816667

Table 2: (Q3) Test accuracy using **binary** features

Question 4

The script to generate the chi-square results, `rank_feat_by_chi_square`, is located in the `scripts/` directory.[1]

[1] As per the discussion on GoPost, our script takes the name of the input file as an argument (rather than reading stdin).

Question 5

The effects of running the `build_kNN.sh` script using features filtered by χ^2 threshold[2] and using cosine simliarity are given below:

[2] For this problem the script takes the filtered feature file as a sixth argument.

p_0	χ^2	Number of related features	Test accuracy
baseline		32846	0.720000
0.001	13.816	3853	0.830000
0.01	9.210	5783	0.850000
0.025	7.378	8172	0.836667
0.05	5.991	12314	0.780000
0.1	4.605	13513	0.766667

Table 3: (Q5) Test accuracy using **real-valued** features, $k = 1$, cosine function

Question 6

The effects of running the same tests as in Q5 but with binary features, $k = 10$, and the cosine function are shown below:

p_0	Test accuracy
baseline	0.823333
0.001	0.846667
0.01	0.856667
0.025	0.840000
0.05	0.853333
0.1	0.836667

Table 4: (Q6) Test accuracy using **binary** features, $k = 10$, cosine function

Question 7

These results show that the kNN algorithm is quite sensitive to the tuning parameters. In particular, the k value chosen has a significant impact on accuracy. Because kNN doesn't attempt to separate class clusters, it assumes that the local neighborhood of each vector represents a natural grouping based on the relative similarity of features. However, there is no guarantee that the k closest training vectors belong to the same class. Thus, if a test vector (projected into the training vector space) lies on the edge of a class boundary, the k nearest neighbors could all belong to an incorrect class. We see this effect at work, where the accuracy decreases with increasing k.

For the same reasons, and to an even greater extent, kNN is extremely sensitive to feature selection. Features that lack discriminatory power serve to artificially increase the measured similarity (in feature space) and so decrease the probability of any given neighborhood being comprised of true class members. We see this confirmed in the χ^2 testing, where filtering features below even a moderate significance threshold ($p = 0.1$) increases accuracy by over 3%, and a stricter significance threshold ($p = 0.01$) by nearly 14%. (It is also worth noting that further reducing the significance threshold begins to reduce accuracy again, as features begin to be filtered out that, although vanishingly likely to be significant by chance, are in fact significant.[3])

[3] Sometimes you do roll the hard six.