Large-Scale Machine Learning EECS 6898, Homework 2

Junfeng He and Sanjiv Kumar

October 20, 2010

Notes:
20 points in total. There are also extra bonus questions, which you are not required but encouraged to work on. Bonus questions are of 7 points in total. The due date is 23:59, Nov. 2, 2010. 2 points would be dropped for each late day.
Feel free to discuss with anyone. But you should answer the questions, complete the programming implementations, and write the report totally on your own. You can not download code from web and use it, unless specified in the question.

1 Review Exercises (4 points)

1.1 Distance metrics (1 point)

Prove KL divergence does not satisfy triangle inequality.

1.2 LSH and Kernelized LSH (2 points)

As we discussed in the class, one definition of LSH is: For a hash function $h()$, if $P(h(x) = h(y))$ is monotonically increasing with some kind of similarity between x and y: $sim(x, y)$, then $h()$ is a locality sensitive hash function.

Design a hash function such that $sim(x, y) = \frac{x^T Ay}{((x^T Ax)(y^T Ay))^{1/2}}$, where A is a positive definite matrix. You need to prove $P(h(x) = h(y))$ is monotonically increasing with $\frac{x^T Ay}{((x^T Ax)(y^T Ay))^{1/2}}$ for your hash function $h()$.

1.3 Min hash (1 point)

Show the relationship between Jaccard distance and L-1 distance.

2 Programming assignments (16 points)

Write an experimental report with your results, observations, etc., together with a brief (a couple sentences) description of each module in your code. Your sub-

mitted file should be named as "hw2_uni_name.zip", e.g., "hw2_jh2700_Junfeng.zip", which contains your code and your report. Please **do not** include data files or intermediate results files in your submission, but store them in your computer. We may request it if necessary. In your report, do not just show your results. Try to provide your observations, analysis of the results, etc. In other words, show what you have learned from the experiments.

You can use any programming language for the experiments, but Matlab (especially 64-bit Matlab) is highly recommended, which would really make your life much easier.

You are given 2 datasets, each of which contains 50,000 training points and 400 query points in Matlab format. For every step in the following tasks, you are supposed to use both datasets.

Notes: Each point in the first dataset is a 512-dim gist feature extracted from a web image. Each point in the second data set is a 784-dim vector, representing the $28 * 28 = 784$ pixels in a handwritten digit image. You can use matlab commands like *im=reshape(x, 28,28);imshow(im);* to see the digit image, where x is one 784-dim vector.

2.1 Exhaustive search (1 points)

To create the ground truth, for each query point, find its top 300 nearest neighbors in the training points by exhaustive scan, using L_2 distance. These 300 nearest neighbors are defined as "good neighbors" for each query.

2.2 Hashing methods (9 points)

Apply the following hashing method on the data to create the hash bits. Use 1 table and m bits. m = 32,64,128. For each query, rank the training points according to their hamming distance to the query. Count the number of "good neighbors" in top k points from hamming ranking. Report the average number of "good neighbors" in top k points of all queries. $k = 100, 300, 900$.

2.2.1 LSH (3 points)

Implement LSH for L_2 distance.

2.2.2 KLSH with RBF kernel (4 points)

Recall, the process of computing each bit for KLSH:

1. Randomly select p points from training data, denoted as $x_1, ..., x_i, ..., x_p$. Build the kernel matrix K among the p points. With RBF kernel, $K(i, j) = \exp(-||x_i - x_j||^2/\sigma^2)$, for $i, j = 1, ..., p$.

2. Apply SVD to K, suppose $K = U\Sigma U^T$. $K^{-1/2} = U\Sigma^{-1/2}U^T$.

3. Form a p-dim vector e_S, where t dimensions are 1 while others are 0. These t dimensions are chosen randomly.

4. $w = K^{-1/2} e_S$. For any x, the bit is created as $h(x) = sign(\sum_{i=1}^{p} w_i k(x, x_i))$.

Note that every e_S would provide one bit, so if you need m bits, you need to repeat the process m times. Each time, e_S is formed with randomly selected t dimensions to be 1.

You can use σ obtained in assignment 1 for RBF kernel parameter. Other parameters to use are: p=500. t = 30.

2.2.3 Principal Component Hashing (2 points)

Compute binary codes using PCA-hashing, i.e., compute the Principal directions and project each point on these directions. Then, threshold each projection at median to generate the corresponding binary bit.

2.3 Hamming radius (3 points)

For every query, find all the points whose hamming distance is within d to the query. You should report the recall (averaged over all queries): number of good neighbors returned / total number of good neighbors, and also precision (averaged over all queries): number of good neighbors returned / number of total returned neighbors, when $d = 0, 1, 2$. Do this for all the three hashing methods in the above question.

Notes: Ideally, to retrieve points within a certain hamming radius of a code, one should build "hash tables" where key is the binary code and value is list of ids of points that have the same code. Then, given a query code, one first enumerates all the possible codes that lie within hamming radius d and returns the set of the points that fall in corresponding buckets. For this, one can use *hash_map* or *map* data structures in C++. If you are not familiar with these structures, for this assignment you can use a simple brute force search in all the training points and return all the points that have hamming distance less than or equal to d.

2.4 Multiple hash tables (3 points)

For LSH method, create 5 hash tables, each of m bits. For a given query, convert it into codes corresponding to each table and retrieve points that have the same code in each hash table. The retrieved points are simply union of all the returned points.

You should report the recall (averaged over all queries): number of good neighbors returned / total number of good neighbors, and also precision (averaged over all queries): number of good neighbors returned / number of total returned neighbors.

3 Bonus questions (7 points)

3.1 p-stable distribution (2 points)

Prove cauchy distribution is 1-stable.

3.2 Tree methods (3 points)

FLANN is a toolbox of tree based methods to find approximate nearest neighbors. Download it from http://people.cs.ubc.ca/ mariusm/index.php/FLANN/FLANN and try it with the two data sets of this homework.

Report the average number of good neighbors in returned k = 100, 300, 900 points.

3.3 Shift-Invariant Kernel Hashing (2 points)

Implement the SIKH method, and apply it to question 2.2 and 2.3.