MS & E 317: Algorithms for Modern Data Models

Instructor: Reza Zadeh (rezab@stanford.edu), Ashish Goel (ashishg@stanford.edu)

HW#3 - Due Monday June 2nd

1. The traditional secretary problem assumes that the number of candidates grows to infinity. Under this assumption, the optimal stopping rule to maximize the probability of finding the best candidate is to say 'no' to the first n/e candidates, and then take the next candidate that is better than the first n/e. The probability of finding the best candidate under this rule is $1/e$.

   In the regime where n is small however, the best proportion may not be $1/e$. Under the small n case of $n = 3$, to maximize the probability of finding the best candidate, how many candidates should be interviewed, and what is the probability of finding the best one?

2. For The Perceptron, we proved in class that if all points are scaled to have *unit norm*, and the two classes can be separated by margin at least γ, then the perceptron will make at most $1/\gamma^2$ mistakes. Notice that the margin changes depending on the scaling of the points. Prove that if all the points have norm at most R (i.e. $|x_i| < R$ for all i - all points live within radius R of the origin), then the perceptron makes at most R^2/γ^2 mistakes.

3. Assume we are optimizing a separable function of n separable parts, where the function has p parameters to be tuned. Recall that Parallel Stochastic Gradient Descent needs one shuffle of the data and one Reduce to compute an average. What is the communication cost for this when k machines are used? Similarly, ADMM requires T iterations, where the communication cost of each iteration consists of an AllReduce. What is the total communication cost of ADMM with T iterations? Finally, notice that the communication cost of ADMM is independent of n, whereas Parallel SGD is not.

4. Design a (c, R, P_1, P_2)-LSH family for the J-dimensional Hamming cube, i.e., assuming N points lie in $\{0, 1\}^J$ and the distance between two points is given by their Hamming distance. Assume $R << J$ and $cR << J$. Analyze K, the number of hash functions used to create an LSH bucket, and L, the number of hash buckets for each point, assuming that we need to satisfy $P_1^K = 1/L$ and $L = N^\rho$ where $\rho = \log(1/P_1)/\log(1/P_2)$. Note that in this setting, the LSH family discussed in class for the Hamming cube has $\rho = 1/c$, but a large value of K. Try to get ρ to be roughly $1/c$ while making K as small as possible.

5. Write down (in pseudo-code) the MAP-REDUCE implementation for generating LSH buckets for solving the (c, R)-NN problem in l_2. The output of the REDUCE job should be LSH buckets, with all the points in a bucket being output along with that bucket. Implement the MAP function of the above algorithm so that it works with the single node map-reduce code we had handed out in the last HW. Assume the file "input.txt" contains the data, one point in each line with the coordinates of the points represented as d real numbers separated by whitespace characters. Also, assume that the configuration parameters L, K, d, Δ are given as part of a configuration file, "lsh.conf", which looks like

L 5
K 6
d 25
Δ 0.1

6. Prove that the shuffle size of the above algorithm can be reduced to $O(LS)$ where the original size of the data is S (i.e. $S = Nd$ if the data is in d dimensions) regardless of the value of K while having negligible impact on the error probability.

   Here, and in all other exercises, we are counting size using long/double as the unit, not bits, and making the additional assumption that each input real number (e.g. each coordinate of each point) is given as a double, and N is much smaller than the maximum value of a long integer variable. In other words, the meaning of "size" corresponds to how it is thought of in practice as opposed to a formal bit-level accounting. Also, for every key value pair that is shuffled, we assume that both the key and the value are included while computing shuffle size.