CS328: Homework 1

Due date 26/1/2019 @11:59 pm

While usage of Python/C/C++/Java etc are all fine, submitting a Jupyter notebook is preferable for each of the coding questions. Copying code is not allowed, from others or any sources. Discussion is fine, but please write down names of people you discussed with.

---

1. (10 points) Write code for the following experiment. Create a function `generateMixture` that does the following: it first reads a set of files. Comma is the separator within a line. The file `centers.txt` contains in its first line k, the number of Gaussians, and d the number of dimensions. This is followed by a line that contains all the w_i values, i.e. the proportions. This is then followed by k lines where each line contains a center location in d dimensions.The covariance matrices are available in files `cov_1.txt,...`. The format is what you would get if we used:

   ```
   numpy.savetxt(fname, Cov, fmt='%.5e', delimiter=',', newline='\n', header='', footer='', comments='# ')
   ```

   After reading this, generate $100k$ samples from this mixture distribution. Store the data along with its label, i.e. the id of the Gaussian that generated it.

2. (10 points) Write another function `fitEM` that takes a list with the generated samples and then runs an EM algorithm for the mixture model. We will measure the following errors:

   - $\sum_i \|\mu_i - \tilde{\mu}_i\|_2$ and $\sum_i \|C_i - \tilde{C}_i\|_F$, where (μ_i, C_i) are the original parameters and $\tilde{\mu}_i, \tilde{C}_i$ are the estimated parameters.

   Plot as a function of number of EM iterations how these two errors change. For now, use a single run of EM.

3. (10 points) Given d, generate points from a d dimensional Gaussian. Generate 1M points and find out what fraction of them lie within, say a distance of $[\sqrt{d} - 2d^{1/4}, \sqrt{d} + 2d^{1/4}]$ from center. Plot how this fraction changes as we increase d from 10 to 1000, say in steps of 10 (x-axis = d, y-axis = fraction of points in the above annulus).

4. (10 points) We will try to develop a efficient (linear time) algorithm for computing the median of an array A of size n, when we can only keep a small part of the data in memory (say at most $n^{3/4}$ values). The following is an outline of the algorithm, fill it in with appropriate proofs. Let $S_1 = \{x \in A, x \leq m\}$ and $S_2 = \{x \in A, x > m\}$, where m is the median. Assume that the elements of A are all distinct.

   (a) Sample $n^{3/4}$ elements of the array without replacement, call this C. Find out a bound B so that the probability that you have at most $n^{3/4}/2 - B$ elements from S_1 in your sample C is at most $o(1/n)$.

   (b) Show that this means that if $u = (n^{3/4}/2 - B)^{th}$element of C, then $u \in S_1$.

   (c) Similarly, argue that if $v = (n^{3/4}/2 + B)^{th}$element of $C\}$, then $v \in S_2$.

   (d) Define $M = \{x \in A, u \leq x \leq v\}$, $\ell_u = |\{x \in A, x < u\}|$ and $\ell_v = |\{x \in A, x > v\}|$ Argue that M contains the median and can be found in linear time.

   (e) If $|M| > 4n^{3/4}$ or $\ell_u > n/2$ or $\ell_v > n/2$, declare FAIL.

   (f) Using M and ℓ_u describe how to find the median element m.

   (g) Argue that if the algorithm did not declare FAIL, it returned the correct median. What are the total time and space required? What is probability that algorithm returns FAIL?

5. (5 points) Suppose that we have an algorithm that takes as input a string of n bits. We are told that the expected running time is $O(n^2)$ if the input bits are chosen independently and uniformly at random. What can Markov's inequality tell us about the worst-case running time of this algorithm on inputs of size n?

6. (5 points) List two topics taught in class that you think you did not understand.