STAT 37710/CAAM 37710/CMSC 35400 Homework 1

University of Chicago, Spring 2024 — due Friday, March 29 at 11:59pm

1. **(25 points)** As we saw in class, k-means clustering minimizes the average square distance distortion

$$J_{\text{avg}^2} = \sum_{j=1}^{k} \sum_{\mathbf{x} \in C_j} d(\mathbf{x}, \boldsymbol{m}_j)^2, \tag{1}$$

where $d(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|$, and C_j is the set of points belonging to cluster j. Another distortion function that we mentioned is the intra-cluster sum of squared distances,

$$J_{\text{IC}} = \sum_{j=1}^{k} \frac{1}{|C_j|} \sum_{\mathbf{x} \in C_j} \sum_{\mathbf{x}' \in C_j} d(\mathbf{x}, \mathbf{x}')^2.$$

(a) Given that in k-means, $\boldsymbol{m}_j = \frac{1}{|C_j|} \sum_{\mathbf{x} \in C_j} \mathbf{x}$, show that $J_{\text{IC}} = 2\, J_{\text{avg}^2}$.

(b) Let $\gamma_i \in \{1, 2, \ldots, k\}$ be the cluster that the i'th datapoint is assigned to, and assume that there are n points in total, $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$. Then (1) can be written as

$$J_{\text{avg}^2}(\gamma_1, \ldots, \gamma_n, \boldsymbol{m}_1, \ldots, \boldsymbol{m}_k) = \sum_{i=1}^{n} d(\mathbf{x}_i, \boldsymbol{m}_{\gamma_i})^2. \tag{2}$$

Recall that k-means clustering alternates the following two steps:

1. Update the cluster assignments:

$$\gamma_i \leftarrow \underset{j \in \{1,2,\ldots,k\}}{\operatorname{argmin}} \; d(\mathbf{x}_i, \boldsymbol{m}_j) \qquad i = 1, 2, \ldots, n.$$

2. Update the centroids:

$$\boldsymbol{m}_j \leftarrow \frac{1}{|C_j|} \sum_{i \,:\, \gamma_i = j} \mathbf{x}_i \qquad j = 1, 2, \ldots, k.$$

Show that the first of these steps minimizes (2) as a function of $\gamma_1, \ldots, \gamma_n$, while holding $\boldsymbol{m}_1, \ldots, \boldsymbol{m}_k$ constant, while the second step minimizes it as a function of $\boldsymbol{m}_1, \ldots, \boldsymbol{m}_k$, while holding $\gamma_1, \ldots, \gamma_n$ constant. The notation "$i \,:\, \gamma_i = j$" should be read as "all i for which $\gamma_i = j$".

(c) Prove that as k-means progresses, the distortion decreases monotonically iteration by iteration.

(d) Give an upper bound on the maximum number of iterations required for full convergence of the algorithm, i.e., the point where neither the centroids, nor the cluster assignments change anymore (note: we do not expect you to prove the bound from Inaba et al., something much looser and simpler will do.)

2. **(25 points)** Implement the k-means algorithm in a language of your choice (Python is recommended), initializing the cluster centers randomly, as explained in the slides. The algorithm should terminate when the cluster assignments (and hence, the centroids) don't change anymore.

Note: if you use a relatively high level language like Python, your code can use linear algebra primitives, such as matrix/vector multiplication, eigendecomposition, etc. directly. However, you *are* expected to write you own k–means and k–means++ functions from scratch. Please don't submit code consisting of a single call to some pre-defined "`kmeans`" function.

(a) The toy dataset `toydata.txt` contains 500 points in $\mathbb{R}^2$ coming from 3 well separated clusters. Test your code on this data and plot the final result as a 2D plot in which each point's color or symbol indicates its cluster assignment. Note that because of the random initialization, different runs may produce different results, and in some cases the algorithm might not correctly identify the three clusters. Plot the value of the distortion function as a function of iteration number for 20 sepearate runs of the algorithm on the same plot. Comment on the plot.

(b) Now implement the k–means++ algorithm discussed in class and repeat part (a) using its result as intialization (except for the 2D plot). Comment on the convergence behavior of k–means++ vs. the original algorithm.

(c) Test both the original k-means and k-means++ algorithms on the MNIST dataset. Each image in MNIST is 28x28 pixels, represented as a 784-dimensional vector. Again, comment on the convergence behavior of k-means++ vs. the original algorithm. Comment on how the results differ when you set the number of clusters to 10 and 16.

3. **(50 points)** Recall the "Mixture of k Gaussians" model used in clustering

$$p(\mathbf{x}, z) = \pi_z \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z), \tag{3}$$

where $\mathbf{x} \in \mathbb{R}^d$, $z \in \{1, 2, \ldots, k\}$ is its cluster assignment, and $\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z)$ is the d–dimensional normal density

$$\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z) = (2\pi)^{-d/2} \left|\boldsymbol{\Sigma}_z\right|^{-1/2} \exp(-(\mathbf{x}-\boldsymbol{\mu}_z)^\top \boldsymbol{\Sigma}_z^{-1} (\mathbf{x}-\boldsymbol{\mu}_z)/2).$$

The parameters of this restricted model are $\theta = (\pi_1, \ldots, \pi_k, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_k)$.

(a) Let $\{(\mathbf{x}_1, z_1), (\mathbf{x}_2, z_2), \ldots, (\mathbf{x}_n, z_n)\}$ be an n–point sample from this model. Write down the corresponding log-likelihood $\ell(\theta)$. As usual, you may ignore constant terms in the log-likelihood.

(b) Let $p_{i,j}$ be the probability $p(z_i = j | \mathbf{x}_i)$ of the i'th data point coming from the j'th cluster (given that its position is $\mathbf{x}_i$). Derive an expression for this probability.

(c) Derive the expected log-likelihood $\bar{\ell}_{\theta_{\text{old}}}(\theta)$ in terms of these $p_{i,j}$ conditional probabilities. Here the expectation is taken over the values of the hidden variables $(z_1, \ldots, z_n)$, and the subscript θ_{old} signifies that the $p_{i,j}$'s are computed with respect to the old values of the parameters, whereas $\bar{\ell}$ itself is a function of the new values of the parameters.

(d) The expectation maximization algorithm updates the parameters of the mixture by maximizing $\bar{\ell}_{\theta_{\text{old}}}(\theta)$. Let us start with the "cluster priors" $\pi_1, \ldots, \pi_k$. Since $(\pi_1, \pi_2, \ldots, \pi_k)$ is a discrete distribution, we must have that $\sum_{j=1}^k \pi_j = 1$. Using this constraint, derive the update rule (mentioned on the slides)

$$\pi_j \leftarrow \frac{1}{n} \sum_{i=1}^n p_{i,j}.$$

(e) Similarly derive the update rule for the $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_k$ location parameters and $\boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2, \ldots, \boldsymbol{\Sigma}_k$ covariance matrices.

(f) Compare these update rules to the k-means update rules derived in Question 1.

(g) Apply the Mixture of Gaussians clustering algorithm to the toy data in Question 2. Plot the final result as a 2D plot in which each point's color or symbol indicates its cluster assignment. Comment on the plot and the convergence speed compared to that of k–means.

4. **(up to 20 points extra credit)** You likely found that on the "toydata" dataset, most of the time even vanilla k–means clustering produces acceptable solutions. Create a dataset of your own for which (on average over many runs) k–means++ improves the performance of k–means by at least a factor of 10 in terms of the distortion function value of the final clustering. Submit the code that you used to generate the dataset.

Chris