CMSC 25400 — Homework 2

University of Chicago, Winter 2019 — due Tuesday, February 22 at 11:59pm

1. 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:

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

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. Recall that the d dimensional normal distribution with mean $\boldsymbol{\mu} \in \mathbb{R}^d$ and covarianvce matrix $\boldsymbol{\Sigma}$ is of the form

$$p(\mathbf{x}) = C \exp(-(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})/2),$$

where C is a normalization factor. In this question we are going to prove that the correct form of the normalization factor is

$$C = \frac{1}{(2\pi)^{d/2} |\boldsymbol{\Sigma}|^{1/2}}.$$

(a) First consider computing the integral of the one dimensional Gaussian function

$$f(x) = e^{-(x-\mu)^2/(2\sigma^2)}. \tag{3}$$

Here μ just translates f, so without loss of generality we can assume that $\mu = 0$.

i. To further simplify things, let's take $\sigma = 1$. Unfortunately,

$$I = \int e^{-x^2/2}\, dx$$

is still not entirely trivial to compute. Poisson came up with the idea of instead considering

$$I^2 = \left(\int e^{-x^2/2}\, dx\right)\left(\int e^{-y^2/2}\, dy\right) = \int\int e^{-(x^2+y^2)/2}\, dx\, dy. \tag{4}$$

Recall that for any (sufficiently regular) function $f\colon \mathbb{R}^2 \to \mathbb{R}$, converting (x, y) to polar coordinates (r, θ),

$$\int\int f(x,y)\, dx\, dy = \int f(r,\theta)\, r\, d\theta\, dr.$$

Use this transformation (together with the fact that $x^2+y^2 = r^2$) to show that $I^2 = 2\pi$, and hence, in this case the correct normalization factor is $C = (2\pi)^{-1/2}$.

ii. Now consider

$$f(x) = e^{-x^2/(2\sigma^2)}.$$

Use a change of variables technique to compute the normalization factor for this case.

(b) Now consider a higher dimensional, but axis aligned Gaussian function written in the form

$$f(\mathbf{x}) = e^{-\mathbf{x}^\top D\mathbf{x}/2},$$

where

$$D = \begin{bmatrix} (\sigma_1)^{-2} & & & \\ & (\sigma_2)^{-2} & & \\ & & \ddots & \\ & & & (\sigma_d)^{-2} \end{bmatrix},$$

and once again, without loss of generality, we took the liberty of setting $\boldsymbol{\mu} = 0$. Show that in this case

$$I = \int f(\mathbf{x})\, d\mathbf{x} = (2\pi)^{d/2} \prod_{i=1}^{d} \sigma_i,$$

and hence the appropriate normalization factor is $C = (2\pi)^{-d/2}\, |D|^{1/2}$, where $|D|$ is the determinant of D.

(c) Finally consider the general d dimensional case

$$f(\mathbf{x}) = e^{-\mathbf{x}^\top \Sigma^{-1} \mathbf{x}/2},$$

where Σ is a generic symmetric positive definite matrix, but once again we set $\boldsymbol{\mu} = 0$. By the spectral theorem, Σ^{-1} can be written in the form

$$\Sigma^{-1} = Q^\top D Q,$$

where D is diagonal and Q is orthogonal.

i. Explain geometrically why

$$\int e^{-\mathbf{x}^\top \Sigma^{-1} \mathbf{x}/2} d\mathbf{x} = \int e^{-\mathbf{x}^\top D \mathbf{x}/2} d\mathbf{x}.$$

ii. Explain why $|\Sigma^{-1}| = |D|$, and use this to complete the derivation of the final formula (3).

3. Recall the "Mixture of k Gaussians" model used in clustering

$$p(\mathbf{x}, z) = \pi_z \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z), \tag{5}$$

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

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

The parameters of this 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 $\overline{\ell}_{\theta_{\text{old}}}(\theta)$ 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 $\overline{\ell}$ itself is a function of the new values of the parameters.

(d) The expectation maximization algorithm updates the parameters of the mixture by maximizing $\overline{\ell}_{\theta_{\text{old}}}(\theta)$. 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.

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

4. Implement the k-means algorithm in a language of your choice (Python 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) We provided a lower resolution subset of the famous MNIST dataset: `mnist_small.txt` for you to experiment with. The file contains 1797 8×8 pixel grey scale images of handwritten digits equally distributed among the digits $0, 1, ..., 9$. Each image has been flattened to a 64 dimensional vector; each row of this text file represents one image.

   Test your code on this data and plot the resulting cluster centers as an image - your results should look like the digits $0, 1, ..., 9$. Note that because of the random initialization, different runs may produce different results, and in some cases some of cluster centers might not look like digits. Plot the value of the distortion function as a function of iteration number over 20 separate runs(different random seeds) of the algorithm on the same plot. Comment on the plot. You may find it helpful to divide the points by 255 so that each pixel entry is now between 0 and 1 before running kmeans.

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

   Below is some sample code for loading the data and visualizing a length 64 vector as an 8×8 image in Python:

```
import matplotlib.pyplot as plt
import numpy as np

# loads the data in a numpy matrix
data = np.loadtxt('mnist_small.txt')
print(data.shape)

# visualize the 0th image
plt.gray()
plt.imshow(data[0].reshape(8,8))
plt.show()
```