UCSB Spring 2020

ECE 283: Homework 4

Topics: Sparsity (PCA and Compressive Sensing)

Reading: Handout 7.2-7.4, which cover SVD/PCA, sparse linear regression, and compressive projections (while you can do the homework based on only this material, you are of course highly encouraged to read the other sections of Handout 7).

Reuse the data generation procedure developed for the high-dimensional data in HW3, recapped here (but we will generate data here in higher dimensions):

- Write the following program to generate a random vector $\mathbf{u}$ in d dimensions as follows: The components of $\mathbf{u}$ are i.i.d., with

$$P[u[i]=0]=2/3, \quad P[u[i]=+1]=1/6, \quad P[u[i]=-1]=1/6$$

  Let $\{\mathbf{u}_j, j=1,...,6\}$ be i.i.d. draws from your program in 4). Draw them once, and then fix them. Check that the vectors are quasi-orthogonal. If the normalized correlation between two of them is too large, then purge one of them and draw another vector. We use these vectors to generate data samples coming from a Gaussian mixture distribution, as follows.

- Write a program to use the fixed vectors $\{\mathbf{u}_j, j=1,...,6\}$ to generate d-dimensional data samples for a Gaussian mixture distribution with 3 *equiprobable* components, as follows. In order to generate any given data point $\mathbf{X}$, we will use i.i.d. draws from a standard Gaussian ($N(0,1)$) distribution that we will denote $\{V_m\}$, and we will also draw a "noise vector" $\mathbf{N} \sim N(0, \sigma^2 \mathbf{I}_d)$ (default value $\sigma^2 = 0.01$). A sample from each of the three components can now be described as follows (remember, a given data sample belongs to exactly one of these components):

  Component 1: Generate $\mathbf{X} = 2\mathbf{u}_1 + 0.5V_1\mathbf{u}_2 + V_2\mathbf{u}_3 + \mathbf{N}$.

  Component 2: Generate $\mathbf{X} = 1.5\mathbf{u}_4 + V_1\mathbf{u}_5 + 0.8V_2\mathbf{u}_6 + \mathbf{N}$.

  Component 3: Generate $\mathbf{X} = \mathbf{u}_6 + V_1(\mathbf{u}_1 - \mathbf{u}_2) + 0.7V_2\mathbf{u}_5 + \mathbf{N}$.

  Note that the vectors $\{\mathbf{u}_j\}$ stay the same across data samples, but the random numbers V_1 and V_2, and the noise vector $\mathbf{N}$ are drawn afresh for each sample.

We will use a higher data dimension d than in HW3. Let us set $d = 100$ as a placeholder, but feel free to play with even higher values.

Part I: PCA

1. Generate $N =?$ (to be varied) data samples from the preceding model, saving both the data point $\mathbf{x}_i$ and $\mathbf{z}_i \in \{0,1\}^3$, the one-hot encoding of which component the data point belongs to. Do an SVD of the $N \times d$ data matrix (you can use off-the-shelf code for this purpose).

   (a) How many dominant singular values (call this d_0) do you see? How does this vary as you increase N, starting from, say, $N = 2d$?

(**b**) Perform a PCA, i.e., project the data down to the dominant d_0 components to obtain an $N \times d_0$ data matrix. Now implement the K-means algorithm with different values of $K = 2, 3, 4, 5, 6$. For each K, either use K-means++ or use several different random initializations, choosing the run that leads to the smallest mean squared error. Feel free to use off-the-shelf code.

Let $\{\mathbf{m}_k, k = 1, ..., K\}$ denote the cluster centers, and for each data point, compute $\mathbf{a}_i \in \{0, 1\}^K$, the one-hot encoding of which cluster the data point is assigned to. Plot the empirical probabilities $P[a_i[k] = 1|z_i[l] = 1]$, $l = 1, 2, 3$, $k = 1, ..., K$ in a $3 \times K$ table, indicating how the "ground truth" components map to the clusters you learn.

2. Try to provide geometric insight into how the cluster centers found by K-means relate to the d_0-dimensional projections of the vectors $\{\mathbf{u}_j\}$ in the model.

Part II: Random Projections for Compressive Sensing

3. Generate a $m \times d$ matrix Φ (m to be determined, d as before) with i.i.d. entries drawn as follows:

$$P[\Phi_{ij} = +1] = 1/2, \quad P[\Phi_{ij} = -1] = 1/2$$

(i.e., the elements of Φ are ± 1 with equal probability).

(**a**) Generate a sample $\mathbf{x} = \mathbf{x}_s + \mathbf{N}$ from the mixture distribution as before, where you should now keep track of the sparse part of the signal $\mathbf{x}_s$ (i.e., the part that depends on the vectors $\{\mathbf{u}_i\}$ and is stronger) generated as follows:

Component 1: Generate $\mathbf{x}_s = 2\mathbf{u}_1 + 0.5V_1\mathbf{u}_2 + V_2\mathbf{u}_3$.

Component 2: Generate $\mathbf{x}_s = 1.5\mathbf{u}_4 + V_1\mathbf{u}_5 + 0.8V_2\mathbf{u}_6$.

Component 3: Generate $\mathbf{x}_s = \mathbf{u}_6 + V_1(\mathbf{u}_1 - \mathbf{u}_2) + 0.7V_2\mathbf{u}_5$.

Also, keep track of $\mathbf{z} \in \{0, 1\}^3$, the one-hot encoding of which component the data point belongs to.

Thus, we are thinking of the signal $\mathbf{x}$ as having a strong sparse part, $\mathbf{x}_s$, together with weaker, non-sparse noise $\mathbf{N}$.

Compute the compressive projection:

$$\mathbf{p} = \frac{1}{\sqrt{m}}\Phi\mathbf{x}$$

(**b**) Define the following basis for the signal:

$$\mathbf{B} = [\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3, \mathbf{u}_4, \mathbf{u}_5, \mathbf{u}_6]$$

4. Find a sparse reconstruction of $\mathbf{x}_s$ based on $\mathbf{y}$ by solving the following lasso problem (feel free to use off-the-shelf code).

$$\begin{aligned} \hat{\mathbf{s}} &= \arg\min_{\mathbf{s}} \tfrac{1}{2}||\mathbf{p} - \tfrac{1}{\sqrt{m}}\Phi\mathbf{B}\mathbf{s}||_2^2 + \lambda||\mathbf{s}||_1 \\ \hat{\mathbf{x}_s} &= \mathbf{B}\hat{\mathbf{s}} \end{aligned} \quad (1)$$

Play with the value of m until you get a satisfactory solution (smallest m such that you get "decent" reconstruction).

5. Compute the normalized MSE $\frac{||\hat{\mathbf{x}_s}-\mathbf{x}_s||^2}{||\mathbf{x}_s||^2}$, averaging over many draws. Plot the normalized MSE versus λ averaged over all the data. Plot the normalized MSE versus λ averaging over data drawn from each of the three mixture components, and comment on whether you see any differences in reconstruction performance for data drawn from different components.

Part III: Random Projections for Sketching

6. For the value of m that you found in 4), project the training examples down to m dimensions using

$$\mathbf{p}_i = \frac{1}{\sqrt{m}}\Phi\mathbf{x}_i \ , \ i = 1, ..., N$$

7. For the vectors $\{\mathbf{u}_i, i = 1, ..., 6\}$ used to generate the data, compare the Euclidean distances squared $||\mathbf{u}_i - \mathbf{u}_j||^2$ versus the corresponding quantities in the projected space. Comment on how well these distances squared are preserved.

8. Implement the K-means algorithm post-projection with different values of $K = 2, 3, 4, 5$. For each K, either use K-means++ or use several different random initializations, choosing the run that leads to the smallest mean squared error. Feel free to use off-the-shelf code.

Let $\{\mathbf{m}_k, k = 1, ..., K\}$ denote the cluster centers, and for each data point, compute $\mathbf{a}_i \in \{0, 1\}^K$, the one-hot encoding of which cluster the data point is assigned to. As in HW3, plot the empirical probabilities $P[a_i[k] = 1|z_i[l] = 1]$, $l = 1, 2, 3$, $k = 1, ..., K$ in a $3 \times K$ table, indicating how the "ground truth" components map to the clusters you learn.

9. Try to provide geometric insight into how the cluster centers found by K-means relate to the m-dimensional projections of the vectors $\{\mathbf{u}_j\}$ in the model.