EECS545 Machine Learning
Homework #5

Due date: 11:55pm, Friday 4/3/2020

Reminder: While you are encouraged to discuss problems in a small group (up to 5 people), you should write your own solutions and source codes independently. In case you worked in a group for the homework, please include the names of your collaborators in the homework submission. Your answers should be as concise and clear as possible. Please address all questions to `http://piazza.com/class#winter2020/eecs545` with a reference to the specific question in the subject line (E.g. RE: Homework 5, Q2(a)).
Submission Instruction: You should submit both **solution** and **source code**. We may inspect your source code submission visaully and run the code to check the results. Please be aware your points can be deducted if you don't follow the instructions listed below:

- Submit **solution** to **Gradescope**
    - Solution should contain your answers to **all** questions (typed or hand-written).
- Submit **source code** to **Canvas**
    - Source code should contain your codes to programming quesitons.
    - Source code should be **1 zip** file named **'HW5_yourUMID_yourfirstname_yourlastname.zip'**
    - The zip file should contain the original files in the **HW5.zip** that we provided; you should not change the file names and folder structure, so that your source code can be run after unzipping your zip file.

In summary, your source code submission for HW5 should be 1 zip file which includes

- q1.ipynb
- q2.ipynb
- q4.ipynb
- q5.ipynb
- q1_data
- q2_data
- q4_data
- q4_unmixed1.wav, . . . , q4_unmixed5.wav

Source Code Instruction: Your source code should run under an environment with following libraries:

- Python 3.6+
- Numpy (for implementations of algorithms)
- Matplotlib (for plots)

Please do not use any other library unless otherwise instructed. You should be able to load the data and execute your code on someone else's computer with the environment described above. In other words, work with the setup we provide and do not change the directory structure. Use relative path instead of absolute path to load the data. Note, the outputs of your source code must match with your solution.

1 [30 points] K-means and Gaussian mixtures for image compression

In this problem, we will apply the K-means algorithm to lossy image compression, by reducing the number of colors used in an image. `q1data` directory contains `mandrill-large.tiff` and `mandrill-small.tiff`. `mandrill-large.tiff` is a 512x512 image of a mandrill represented in 24-bit color. This means that, for each of the 262,144 pixels in the image, there are three 8-bit numbers (each ranging from 0 to 255) that represent the red, green, and blue intensity values for that pixel. The straightforward representation of this image therefore takes about $262,144 \times 3 = 786,432$ bytes (a byte being 8 bits). To compress the image, we will use K-means to reduce the image to $K = 16$ colors. More specifically, each pixel in the image is considered a point in the three-dimensional (r, g, b)-space. To compress the image, we will cluster these points in color-space into 16 clusters, and replace each pixel with the closest cluster centroid. Follow the instructions below. Be warned that the code should run in less than 1 minute; otherwise, you will loose points.

(a) (10 pts) The starter code `q1.ipynb` reads an image `mandrill-small.tiff`. Treating each pixel's (r, g, b) values as an element of $\mathbb{R}^3$, implement and run K-means with 16 clusters on the pixel data from this image, iterating 50 times. Measure the pixel error at each iteration. We provided the initial centroids as `initial-centroids` in the starter code.

(b) (3 pts) After training, read the test image `mandrill-large.tiff`, and replace each pixel's (r, g, b) values with the value of the closest cluster centroid. Display the new image, and measure the pixel error using provided `calculate_error` function.

(c) (2 pts) If we represent the image with these reduced 16 colors, by (approximately) what factor have we compressed the image?

For (d), (e), and (f), we will repeat (a), (b) and (c) by implementing Gaussian mixtures (with full covariances) with $K = 5$.

(d) (10 pts) Repeat (a) by implementing Gaussian mixtures with $K = 5$. Iterate 50 times and measure the log-likelihood of the training data at each iteration. We provided the initial mean and covariance matrices, and prior distribution of latent cluster as `initial-mu`, `initial-sigma`, and `initial-pi` in the starter code.

(e) (3 pts) After training, read the test image `mandrill-large.tiff`, and replace each pixel's (r, g, b) values with the value of latent cluster mean, where we use the MAP estimation for the latent cluster-assignment variable for each pixel. Display the new image, and measure the pixel error using provided `calculate_error` function. In addition, report the model parameters $\{(\mu_k, \Sigma_k) : k = 1, ..., 5\}$.

(f) (2 pts) If we represent the image with these reduced 5 colors, by (approximately) what factor have we compressed the image?

2 [20 points] PCA and eigenfaces

(a) (7 pts) In the lecture, we derived PCA from "maximizing variance" viewpoint. In this problem, we will take the "minimizing squared error" viewpoint. Assume the data is zero-centered. Let $K \in \{0, 1, \ldots, d\}$ be arbitrary and let $\mathbf{x}^{(n)} \in \mathbb{R}^d$. Let $\mathcal{U} = \{\mathbf{U} \in \mathbb{R}^{d \times K} | \{\mathbf{u}_i\}_{i=1}^K$'s are orthonormal vectors$\}$, where $\mathbf{u}_i$ is the i-th column vector of $\mathbf{U}$. Show that the following equation holds

$$\min_{\mathbf{U} \in \mathcal{U}} \frac{1}{N} \sum_{n=1}^{N} \left\| \mathbf{x}^{(n)} - \mathbf{U}\mathbf{U}^T \mathbf{x}^{(n)} \right\|^2 = \min_{\mathbf{U} \in \mathcal{U}} \frac{1}{N} \sum_{n=1}^{N} \left\| \mathbf{x}^{(n)} - \sum_{i=1}^{K} \mathbf{u}_i \mathbf{u}_i^T \mathbf{x}^{(n)} \right\|^2 = \sum_{k=K+1}^{d} \lambda_k,$$

where $\lambda_1 \geq \ldots \geq \lambda_d$ are the (ordered) eigenvalues of the data covariance $\frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}^{(n)} - \bar{\mathbf{x}})(\mathbf{x}^{(n)} - \bar{\mathbf{x}})^T$, $\bar{\mathbf{x}}$ is the data mean vector. Also, show that the $\mathbf{u}_i$'s that solves the minimization problem above is the eigenvectors corresponding to the (ordered) eigenvalues λ_i's. Here, $\mathbf{U}\mathbf{U}^T\mathbf{x}^{(n)}$ is called a projection of $\mathbf{x}^{(n)}$ into the subspace spanned by $\mathbf{u}_i$'s.

Now, you will apply PCA to face images. The principal components (eigenvectors) of the face images are called eigenfaces.

(b) (6 pts) Use the starter code q2.ipynb to load the data from q2.npy. By regarding each image as a vector in a high dimensional space, perform PCA on the face images (sort the eigenvalues in descending order). Report the eigenvalues corresponding to the first 10 principal components, and plot all the eigenvalues where x-axis is the index of corresponding principal components and y-axis is the eigenvalue.

(c) (4 pts) Hand in a 2×5 array of subplots showing the first 10 principal components/eigenvectors ("eigenfaces") (sorted according to the descending eigenvalues) as images, treating the mean of images as the first principal component. Comment on what facial or lighting variations some of the different principal components are capturing (Note: you don't need to comment for all the images. Just pick a few that capture some salient aspects of image).

(d) (3 pts) Eigenfaces are a set of bases for all the images in the dataset (every image can be represented as a linear combination of these eigenfaces). Suppose we have L eigenfaces in total. Then we can use the L coefficients (of the bases, i.e. the eigenfaces) to represent an image. Moreover, we can use the first $K(< L)$ eigenfaces to reconstruct the face image approximately (correspondingly use K coefficients to represent the image). In this case, we reduce the dimension of the representation of images from L to K. To determine the proper K to use, we will check the percentage of variance that has been preserved (recall that the basic idea of PCA is preserving variance). Specifically, we define total variance

$$v(K) = \sum_{i=1}^{K} \lambda_i$$

where $1 \leq K \leq L$ and $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_L$ are eigenvalues. Then the percentage of total variance is

$$\frac{v(K)}{v(L)}$$

How many principal components are needed to represent 95% of the total variance? How about 99%? What is the percentage of reduction in dimension in each case?

3 [10 + 2 points] Expectation Maximization

Consider a probabilistic model with random variables $z \in \{0, 1\}, \epsilon \in \mathbb{R}$, and $x \in \mathbb{R}$. The random variables z and ϵ are independent. The joint distribution of all three variables is as follows:

$$\begin{aligned} z &\sim \text{Bernoulli}(\phi) \\ \epsilon &\sim \text{Normal}(\mu, \sigma^2) \\ x &= z + \epsilon \end{aligned}$$

The parameters of this model are $\phi \in \mathbb{R}, \mu \in \mathbb{R}$, and $\sigma \in \mathbb{R}$.

(a) (2 pts) Write down in English a simple, one-line description of what the marginal distribution of x looks like.

(b) (3 pts) Suppose we have a training set $\{(z^{(1)}, \epsilon^{(1)}, x^{(1)}), \ldots, (z^{(N)}, \epsilon^{(N)}, x^{(N)})\}$, where all three variables z, ϵ, x are observed. Write down the log-likelihood of the variables, and derive the maximum likelihood estimates of the model's parameters.

(c) (5 pts) Now, suppose z and ϵ are latent (unobserved) random variables. Our training set is therefore of the form $\{x^{(i)}, \ldots, x^{(N)}\}$. Write down the log-likelihood of the variables, and derive an EM algorithm to maximize the log-likelihood. Clearly indicate what are the E-step and the M-step.

(d) (Extra 2 pts) Consider the modified probabilistic model in the following:

$$\begin{aligned} z &\sim \text{Bernoulli}(\phi) \\ \epsilon &\sim \text{Normal}(\mu, \sigma^2) \\ x &= \lambda z + \epsilon, \end{aligned}$$

where $\lambda \in \mathbb{R}$ is the extra parameter. Why would you consider such modification in the model? Is there any advantage of the modified model over the original model? [Hint: Which model is more general? Why?]

4 [15 points] Independent Component Analysis

In this problem, you will implement maximum-likelihood Independent Component Analysis (ICA) for blind audio separation. As we learned in the lecture, the maximum-likelihood ICA minimizes the following loss:

$$\ell(W) = \sum_{i=1}^{N} \left(\sum_{j=1}^{m} \log g' \left(w_j^T x^{(i)} \right) + \log |W| \right), \tag{1}$$

where N is the number of time steps, m is the number of independent sources, W is the transformation matrix representing a concatenation of w_j's, and $g(s) = 1/(1+e^{-s})$ is the sigmoid function. This link has some nice demos of blind audio separation: `https://cnl.salk.edu/~tewon/Blind/blind_audio.html`. We provided the starter code `q4.ipynb` and the data `q4.dat`. The file `q4.dat` contains a matrix with 5 columns, with each column corresponding to a mixed signal recorded from one of the microphone.

(a) Implement and run ICA, and report what was the W matrix you found. Also, submit the 5 unmixed tracks (See the instruction in the starter code for downloading each track in `.wav` format) with the filenames as `q4_unmixed1.wav`, ..., `q4_unmixed5.wav`. Please zip it together with the codes and submit to Canvas (See **Submission Instruction**). To make sure your code is correct, you should listen to the resulting unmixed sources. (Some overlap in the sources may be present, but the different sources should be pretty clearly separated.)

5 [25 points] Conditional Variational Autoencoders.

In this problem, you will implement a conditional variational autoencoder (CVAE) from [1] and train it on the MNIST dataset.

(a) (8 pts) Derive the variational lowerbound of a conditional variational autoencoder. Show that:

$$\begin{aligned} \log p_\theta(\mathbf{x}|\mathbf{y}) &\geq \mathcal{L}(\theta, \phi; \mathbf{x}, \mathbf{y}) \\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})} \left[\log p_\theta(\mathbf{x}|\mathbf{z}, \mathbf{y}) \right] - D_{KL}\left(q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y}) \| p_\theta(\mathbf{z}|\mathbf{y}) \right), \end{aligned} \tag{2}$$

where $\mathbf{x}$ is a binary vector of dimension d, $\mathbf{y}$ is a one-hot vector of dimension c defining a class, $\mathbf{z}$ is a vector of dimension m sampled from the posterior distribution $q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y})$. The posterior distribution is modeled by a neural network of parameters ϕ. The generative distribution $p_\theta(\mathbf{x}|\mathbf{y})$ is modeled by another neural network of parameters θ. Similar to the VAE that we learned in the class, we assume the conditional independence on the componenets of $\mathbf{z}$: i.e., $q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y}) = \prod_{j=1}^{m} q_\phi(z_j|\mathbf{x}, \mathbf{y})$, and $p_\theta(\mathbf{z}|\mathbf{y}) = \prod_{j=1}^{m} p_\theta(z_j|\mathbf{y})$.

(b) (7 pts) Derive the analytical solution to the KL-divergence between two Gaussian distributions $D_{KL}\left(q_\phi\left(\mathbf{z}|\mathbf{x},\mathbf{y}\right)\|p_\theta\left(\mathbf{z}|\mathbf{y}\right)\right)$. Let us assume that $p_\theta\left(\mathbf{z}|\mathbf{y}\right)\sim\mathcal{N}(\mathbf{0},\mathbf{I})$ and show that:

$$D_{KL}\left(q_\phi\left(\mathbf{z}|\mathbf{x},\mathbf{y}\right)\|p_\theta\left(\mathbf{z}|\mathbf{y}\right)\right)=-\frac{1}{2}\sum_{j=1}^{m}\left(1+\log\left(\sigma_j^2\right)-\mu_j^2-\sigma_j^2\right), \tag{3}$$

where μ_j and σ_j are the outputs of the neural network that estimates the parameters of the posterior distribution $q_\phi\left(\mathbf{z}|\mathbf{x},\mathbf{y}\right)$.

(c) (10 pts) Fill in code for CVAE network as a `nn.Module` class called `CVAE` in the starter code `q5.ipynb`

- Implement the `recognition_model` function $q_\phi\left(\mathbf{z}|\mathbf{x},\mathbf{y}\right)$.
- Implement the `generative_model` function $p_\theta\left(\mathbf{x}|\mathbf{z},\mathbf{y}\right)$.
- Implement the `forward` function by inferring the Gaussian parameters using the recognition model, sampling a latent variable using the reparametrization trick and generating the data using the generative model.
- Implement the variational lowerbound `loss_function` $\mathcal{L}\left(\theta,\phi;\mathbf{x},\mathbf{y}\right)$.
- Train the CVAE and visualize the generated image for each class (i.e., 10 images).
- Repeat the image generation three times with different random noise. Submit 3 x 10 array of subplots showing all the generated images, where the images in the same row are generated from the same random noise, and images in the same column are generated from the the same class label.

If trained successfully, you should be able to sample images $\mathbf{x}$ that reflect the given label $\mathbf{y}$ given the noise vector $\mathbf{z}$.

6 Change log

- [Mar/23 00:40, Q2(a)] Add $1/N$ in the equation to make the formulation consistent with the lecture note.
- [Mar/24 18:00, Q2(a)] Add $\forall j, \|\mathbf{u}_j\|_2 = 1$ as a constraint for minimization problem.
- [Mar/24 24:00, Q1(b)] Fix a typo in function name: `evaluate` → `calculate_error`
- [Mar/26 18:00, Q2(a)] Make the condition on $\mathbf{U}$ more clear.
- [Mar/26 19:00, Q4] Fix typos
  - "N is the number of mixed signals" → "N is the number of time steps".
  - "each column corresponding to one of the mixed signals $x^{(i)}$" → "each column corresponding to a mixed signal recorded from one of the microphone."
- [Mar/28 20:30, Q5] Fix a typo in summation: $J \to m$.
  Clarify on the conditional independence assumption on the componenets of $\mathbf{z}$.
- [Mar/30 10:00, Q4] Added the instruction of the filename for `.wav` files submission.
- [Apr/1 11:00, Q5] Fix a typo in `q5.ipynb` from `print(... loss.data / len(data))` to `print(... loss.data)`. Note that this **only affects** the print out. We will consider the both answer with previous code and modified code as a correct answer.

Credits

References

[1] Kihyuk Sohn, Xinchen Yan, and Honglak Lee. Learning structured output representation using deep conditional generative models. In *NeurIPS*. 2015.