Assignment #5

Due: 11:59pm EST, April 8, 2021

Homework 5: EM with Mixtures, PCA, and Graphical Models

This homework assignment will have you work with EM for mixtures, PCA, and graphical models. We encourage you to read sections 9.4 and 8.2.5 of the course textbook.

Please type your solutions after the corresponding problems using this LaTeX template, and start each problem on a new page.

Please submit the **writeup PDF to the Gradescope assignment 'HW5'**. Remember to assign pages for each question.

Please submit your **LaTeX file and code files to the Gradescope assignment 'HW5 - Supplemental'**.

Problem 1 (Expectation-Maximization for Gamma Mixture Models, 25pts)

In this problem we will explore expectation-maximization for a Categorical-Gamma Mixture model.

Let us suppose the following generative story for an observation x: first one of K classes is randomly selected, and then the features x are sampled according to this class. If

$$z \sim \text{Categorical}(\boldsymbol{\theta})$$

indicates the selected class, then x is sampled according to the class or "component" distribution corresponding to z. (Here, $\boldsymbol{\theta}$ is the mixing proportion over the K components: $\sum_k \theta_k = 1$ and $\theta_k > 0$). In this problem, we assume these component distributions are gamma distributions with shared shape parameter but different rate parameters:

$$x|z \sim \text{Gamma}(\alpha, \beta_k).$$

In an unsupervised setting, we are only given a set of observables as our training dataset: $\mathcal{D} = \{x_n\}_{n=1}^N$. The EM algorithm allows us to learn the underlying generative process (the parameters $\boldsymbol{\theta}$ and $\{\beta_k\}$) despite not having the latent variables $\{z_n\}$ corresponding to our training data.

1. **Intractability of the Data Likelihood** We are generally interested in finding a set of parameters β_k that maximizes the likelihood of the observed data:

   $$\log p(\{x_n\}_{n=1}^N; \boldsymbol{\theta}, \{\beta_k\}_{k=1}^K).$$

   Expand the data likelihood to include the necessary sums over observations x_n and to marginalize out the latents $\mathbf{z}_n$. Why is optimizing this likelihood directly intractable?

2. **Complete Data Log Likelihood** The complete dataset $\mathcal{D} = \{(x_n, \mathbf{z}_n)\}_{n=1}^N$ includes latents $\mathbf{z}_n$. Write out the negative complete data log likelihood:

   $$\mathcal{L}(\boldsymbol{\theta}, \{\beta_k\}_{k=1}^K) = -\log p(\mathcal{D}; \boldsymbol{\theta}, \{\beta_k\}_{k=1}^K).$$

   Apply the power trick and simplify your expression using indicator elements z_{nk}.[a] Notice that optimizing this loss is now computationally tractable if we know $\mathbf{z}_n$.

(Continued on next page.)

[a] The "power trick" is used when terms in a PDF are raised to the power of indicator components of a one-hot vector. For example, it allows us to rewrite $p(\mathbf{z}_n; \boldsymbol{\theta}) = \prod_k \theta_k^{z_{nk}}$.

Problem 1 (cont.)

3. **Expectation Step** Our next step is to introduce a mathematical expression for $\mathbf{q}_n$, the posterior over the hidden component variables $\mathbf{z}_n$ conditioned on the observed data x_n with fixed parameters. That is:

$$\mathbf{q}_n = \begin{bmatrix} p(\mathbf{z}_n = \mathbf{C}_1 | x_n; \boldsymbol{\theta}, \{\beta_k\}_{k=1}^K) \\ \vdots \\ p(\mathbf{z}_n = \mathbf{C}_K | x_n; \boldsymbol{\theta}, \{\beta_k\}_{k=1}^K) \end{bmatrix}.$$

   Write down and simplify the expression for $\mathbf{q}_n$. Note that because the $\mathbf{q}_n$ represents the posterior over the hidden categorical variables $\mathbf{z}_n$, the components of vector $\mathbf{q}_n$ must sum to 1. The main work is to find an expression for $p(\mathbf{z}_n | x_n; \boldsymbol{\theta}, \{\beta_k\}_{k=1}^K)$ for any choice of $\mathbf{z}_n$; i.e., for any 1-hot encoded $\mathbf{z}_n$. With this, you can then construct the different components that make up the vector $\mathbf{q}_n$.

4. **Maximization Step** Using the $\mathbf{q}_n$ estimates from the Expectation Step, derive an update for maximizing the expected complete data log likelihood in terms of $\boldsymbol{\theta}$ and $\{\beta_k\}_{k=1}^K$.

   (a) Derive an expression for the expected complete data log likelihood using $\mathbf{q}_n$.

   (b) Find an expression for $\boldsymbol{\theta}$ that maximizes this expected complete data log likelihood. You may find it helpful to use Lagrange multipliers in order to enforce the constraint $\sum \theta_k = 1$. Why does this optimal $\boldsymbol{\theta}$ make intuitive sense?

   (c) Find an expression for β_k that maximizes the expected complete data log likelihood. Why does this optimal β_k make intuitive sense?

5. Suppose that this had been a classification problem. That is, you were provided the "true" components $\mathbf{z}_n$ for each observation x_n, and you were going to perform the classification by inverting the provided generative model (i.e. now you're predicting $\mathbf{z}_n$ given x_n). Could you reuse any of your derivations above to estimate the parameters of the model?

6. Finally, implement your solution in `p1.ipynb` and attach the final plot below.

   You will recieve no points for code not included below.

Solution

1. We can begin by writing down the likelihood of the data, given the parameters:

$$\begin{aligned} p(\{x_n\}_{n=1}^N; \boldsymbol{\theta}, \{\beta_k\}_{k=1}^K) &= \prod_{n=1}^N p(x_n; \boldsymbol{\theta}, \{\beta_k\}_{k=1}^K) \\ &= \prod_{n=1}^N \sum_{k=1}^K p(x_n; \boldsymbol{\theta}, \{\beta_k\}_{k=1}^K) p(z_n = C_k; \boldsymbol{\theta}_k) \end{aligned}$$

Using the Law of Total Probability (LOTP), we discover that the likelihood of each individual data point is equal to the gamma distribution multiplied by the categorical distribution summed over every class. Thus, the log likelihood is given by:

$$\begin{aligned} \log p(\{x_n\}_{n=1}^N; \boldsymbol{\theta}, \{\beta_k\}_{k=1}^K) &= \log \left[\prod_{n=1}^N \sum_{k=1}^K p(x_n; \boldsymbol{\theta}, \{\beta_k\}_{k=1}^K) p(z_n = C_k; \boldsymbol{\theta}_k) \right] \\ &= \log \left[\prod_{n=1}^N \sum_{k=1}^K \text{Gamma}(\alpha, \beta_k) \cdot \boldsymbol{\theta}_k \right] \\ &= \sum_{n=1}^N \log \left[\sum_{k=1}^K \text{Gamma}(\alpha, \beta_k) \cdot \boldsymbol{\theta}_k \right] \end{aligned}$$

Optimizing this log likelihood is intractable because we cannot easily optimize the log of the sum, even though we can adeptly optimize a sum of logs.

2. We can expand the negative complete data log likelihood using LOTP:

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}, \{\beta_k\}_{k=1}^K) &= -\log p(\mathcal{D}; \boldsymbol{\theta}, \{\beta_k\}_{k=1}^K) \\ &= -\log \left[\prod_{n=1}^N p(x_n; \boldsymbol{\theta}, \{\beta_k\}_{k=1}^K) \right] \\ &= -\sum_{n=1}^N \log \left[p(x_n; \boldsymbol{\theta}, \{\beta_k\}_{k=1}^K) \right] \\ &= -\sum_{n=1}^N \log \left[\prod_{k=1}^K p(x_n | z_n = C_k; \boldsymbol{\theta}, \{\beta_k\}_{k=1}^K) p(z_n = C_k; \boldsymbol{\theta}_k) \right] \\ &= -\sum_{n=1}^N \left(\log \left[\prod_{k=1}^K p(x_n | z_n = C_k; \boldsymbol{\theta}, \{\beta_k\}_{k=1}^K) \right] + \log \left[\prod_{k=1}^K p(z_n = C_k; \boldsymbol{\theta}_k) \right] \right) \end{aligned}$$

Applying the power trick, where z_{nk} is the indicator that x_n is sampled from class k (C_k), we get:

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}, \{\beta_k\}_{k=1}^K) &= -\sum_{n=1}^N \left(\log \left[\prod_{k=1}^K p(x_n; \boldsymbol{\theta}, \{\beta_k\}_{k=1}^K)^{z_{nk}} \right] + \log \left[\prod_{k=1}^K \boldsymbol{\theta}_k^{z_{nk}} \right] \right) \\ &= -\sum_{n=1}^N \left(\sum_{k=1}^K z_{nk} \left(\log \left[p(x_n; \boldsymbol{\theta}, \{\beta_k\}_{k=1}^K) \right] + \log \left[\boldsymbol{\theta}_k \right] \right) \right) \\ &= -\sum_{n=1}^N \left(\sum_{k=1}^K z_{nk} \log \left[\text{Gamma}(\alpha, \beta_k) \cdot \boldsymbol{\theta}_k \right] \right) \end{aligned}$$

3. First, find $\mathbf{q}_{nk} = p(\mathbf{z}_n = \mathbf{C}_k | x_n; \boldsymbol{\theta}, \{\beta_k\}_{k=1}^K)$. Using Bayes' Rule and LOTP in the denominator, we get:

$$\begin{aligned}
\mathbf{q}_{nk} &= \frac{p(x_n|\mathbf{z}_n = \mathbf{C}_k; \boldsymbol{\theta}, \{\beta_k\}_{k=1}^K) p(z_n = C_k; \boldsymbol{\theta}_k)}{p(x_n; \boldsymbol{\theta}, \{\beta_k\}_{k=1}^K)} \\
&= \frac{p(x_n|\mathbf{z}_n = \mathbf{C}_k; \boldsymbol{\theta}, \{\beta_k\}_{k=1}^K) p(z_n = C_k; \boldsymbol{\theta}_k)}{\sum_{i=1}^K p(x_n|\mathbf{z}_n = \mathbf{C}_i; \boldsymbol{\theta}, \{\beta_i\}_{i=1}^K) p(z_n = \mathbf{C}_i; \boldsymbol{\theta}_i)} \\
&= \frac{\text{Gamma}(\alpha, \beta_k) \cdot \boldsymbol{\theta}_k}{\sum_{i=1}^K \text{Gamma}(\alpha, \beta_i) \cdot \boldsymbol{\theta}_i}
\end{aligned}$$

Therefore, in general:

$$\mathbf{q}_n = \frac{1}{\sum_{i=1}^K \text{Gamma}(\alpha, \beta_i) \cdot \boldsymbol{\theta}_i} \cdot \begin{bmatrix} \text{Gamma}(\alpha, \beta_1) \cdot \boldsymbol{\theta}_1 \\ \vdots \\ \text{Gamma}(\alpha, \beta_K) \cdot \boldsymbol{\theta}_K \end{bmatrix}.$$

4. (a) Taking the expectation of the complete data log likelihood, i.e. the negation of the negative complete data log-likelihood we derived in 2, and applying linearity:

$$\begin{aligned}
E_{z|x}\left[-\mathcal{L}(\boldsymbol{\theta}, \{\beta_k\}_{k=1}^K)\right] &= E_{z|x}\left[\sum_{n=1}^N \left(\sum_{k=1}^K z_{nk} \log\left[\text{Gamma}(\alpha, \beta_k) \cdot \boldsymbol{\theta}_k\right]\right)\right] \\
&= \sum_{n=1}^N \left(\sum_{k=1}^K E_{z|x}\left[z_{nk}\right] \log\left[\text{Gamma}(\alpha, \beta_k) \cdot \boldsymbol{\theta}_k\right]\right) \\
&= \sum_{n=1}^N \left(\sum_{k=1}^K q_{nk} \log\left[\text{Gamma}(\alpha, \beta_k) \cdot \boldsymbol{\theta}_k\right]\right)
\end{aligned}$$

(b) In order to find the optimal $\boldsymbol{\theta}$, $\boldsymbol{\theta}^*$, we solve the following optimization problem (minimizing the expected negative log likelihood, derived from 4(a), subject to the given constraint on $\boldsymbol{\theta}$):

$$\arg\min_{\boldsymbol{\theta}} E_{z|x}\left[\mathcal{L}(\boldsymbol{\theta}, \{\beta_k\}_{k=1}^K)\right] \quad \text{s.t.} \quad \sum_{k=1}^K \boldsymbol{\theta}_k = 1$$

We set up the Lagrangian and take derivatives with respect to $\boldsymbol{\theta}_k$ and λ:

$$\begin{aligned}
L(\boldsymbol{\theta}, \lambda) &= \sum_{n=1}^N \left(\sum_{k=1}^K q_{nk} \log\left[\text{Gamma}(\alpha, \beta_k)\right] + \log\left[\boldsymbol{\theta}_k\right]\right) - \lambda\left(\sum_{k=1}^K \boldsymbol{\theta}_k - 1\right) \\
\implies \frac{\partial L}{\partial \boldsymbol{\theta}_k} &= -\sum_{n=1}^N \left(\frac{q_{nk}}{\boldsymbol{\theta}_k}\right) - \lambda \qquad (i) \\
\implies \frac{\partial L}{\partial \lambda} &= -\sum_{k=1}^K \boldsymbol{\theta}_k - 1 \qquad (ii)
\end{aligned}$$

The Lagrangian is minimized when these derivatives equal 0. Rearranging (i) gives:

$$\boldsymbol{\theta}_k = -\left(\frac{\sum_{n=1}^N q_{nk}}{\lambda}\right)$$

Substituting this expression into (ii) and rearranging gives:

$$\lambda = -\sum_{k=1}^{K}\sum_{n=1}^{N} q_{nk}$$

Putting this together, we get an expression for the optimal $\boldsymbol{\theta}_k \forall k$:

$$\boldsymbol{\theta}_k^* = \frac{\sum_{n=1}^{N} q_{nk}}{\sum_{k=1}^{K}\sum_{n=1}^{N} q_{nk}} = \frac{\sum_{n=1}^{N} q_{nk}}{N}$$

This makes intuitive sense as $\boldsymbol{\theta}_k^*$ represents the expected proportion of the x_n's sampled from class k.

(c) In order to find the optimal β_k, β_k^*, we solve the following optimization problem (minimizing the expected negative log likelihood, derived from 4(a)):

$$\begin{aligned}\arg\min_{\beta_k \forall k} E_{z|x}\left[\mathcal{L}(\boldsymbol{\theta}, \{\beta_k\}_{k=1}^K)\right] &= \arg\min_{\beta_k \forall k}\left\{\sum_{n=1}^{N}\left(\sum_{k=1}^{K} q_{nk}\log\left[\text{Gamma}(\alpha,\beta_k)\right] + \log\left[\boldsymbol{\theta}_k\right]\right)\right\} \\ &= \arg\min_{\beta_k \forall k}\left\{\sum_{n=1}^{N}\left(\sum_{k=1}^{K} q_{nk}\left(\alpha\log\left[\beta_k\right] - \beta_k x_n + ...\right)\right)\right\}\end{aligned}$$

which we derive by substituting in the PDF of the Gamma distribution, simplifying, and abstracting away terms not including β_k into the ...

Now, we take the derivative with respect to β_k for a specific k:

$$\frac{\partial E_{z|x}[\mathcal{L}(.)]}{\partial \beta_k} = -\sum_{n=1}^{N} q_{nk}\left(\frac{\alpha}{\beta_k} - x_n\right)$$

The expected negative log likelihood is minimized when this derivative equals 0. We can rearrange this equation to solve for the optimal β_k which can be generalized $\forall k$:

$$\beta_k^* = \frac{\alpha\sum_{n=1}^{N} q_{nk}}{\sum_{n=1}^{N} x_n q_{nk}}$$

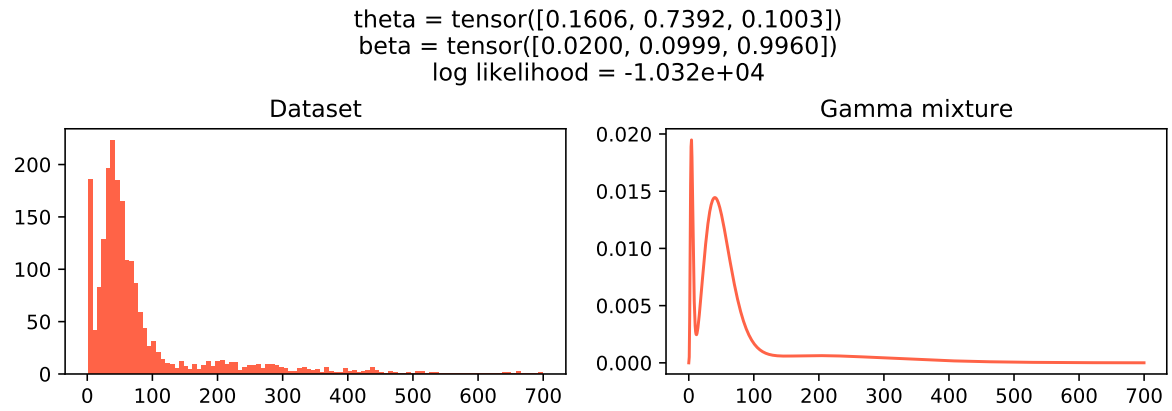
This makes intuitive sense as the mean of the Gamma(α, β_k) is:

$$\mu_k = \frac{\alpha}{\beta_k} = \frac{\sum_{n=1}^{N} x_n q_{nk}}{\sum_{n=1}^{N} q_{nk}}$$

Using our expression for β_k^* and rearranging, as above, we get that the mean of class k is equal to the average of the x_n's weighted according to the probability they were sampled from class k. Using β_k^*, we know that this estimation is the MLE of μ_k. Thus, preserving the invariance of the MLE, it must be true that when we rearrange for β_k, we get the optimal β_k.

5. Yes, we could reuse our derivations from the M-step, but instead of q_{nk} we can simply use the indicators z_{nk} which are known. In doing so, we will end up with optimal parameters β_k^*'s and $\boldsymbol{\theta}^*$, calculated using the classic MLE approach - maximizing the data log likelihood (no longer the expected log likelihood!) with respect to the parameters, including the constraint on $\boldsymbol{\theta}$ from 4(b).

6. Plot:



Code:

```
def e_step(theta, betas):
    q = ds.Gamma(alpha, betas).log_prob(x) + torch.log(theta)
    q -= q.logsumexp(dim=1, keepdim=True)

    return q.exp()


def m_step(q):
    theta_hat = torch.sum(q, axis=0) / len(x)

    x_t = torch.transpose(x,0,1)
    beta_hats = alpha * torch.sum(q, axis=0) / torch.sum(torch.matmul(x_t, q), axis=0)

    return theta_hat, beta_hats


def log_px(x, theta, betas):
    p_gamma = ds.Gamma(alpha, betas).log_prob(x).exp()
    p = torch.sum(p_gamma * theta, axis=1)

    return p.log()


def run_em(theta, betas, iterations=1000):
    for _ in range(iterations):
        q = e_step(theta, betas)
        theta, betas = m_step(q)

    return theta, betas
```

Problem 2 (PCA, 15 pts)

For this problem you will implement PCA from scratch on the first 6000 images of the MNIST dataset. Your job is to apply PCA on MNIST and discuss what kind of structure is found. Implement your solution in `p2.ipynb` and attach the final plots below.

You will recieve no points for using third-party PCA implementations (i.e. `scikit-learn`).

You will recieve no points for code not included below.

1. Compute the PCA. Plot the eigenvalues corresponding to the most significant 500 components in order from most significant to least. Make another plot that describes the cumulative proportion of variance explained by the first k most significant components for values of k from 1 through 500. How much variance is explained by the first 500 components? Describe how the cumulative proportion of variance explained changes with k. Include this plot below.

2. Plot the mean image of the dataset and plot an image corresponding to each of the first 10 principle components. How do the principle component images compare to the cluster centers from K-means? Discuss any similarities and differences. Include these two plots below.

   Reminder: Center the data before performing PCA

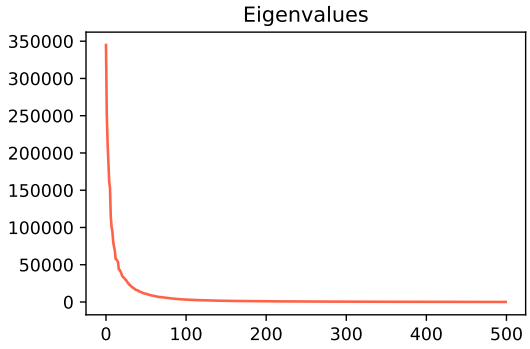
3. Compute the reconstruction error on the data set using the mean image of the dataset. Then compute the reconstruction error using the first 10 principal components. How do these errors compare to the final objective loss achieved by using K-means on the dataset? Discuss any similarities and differences.

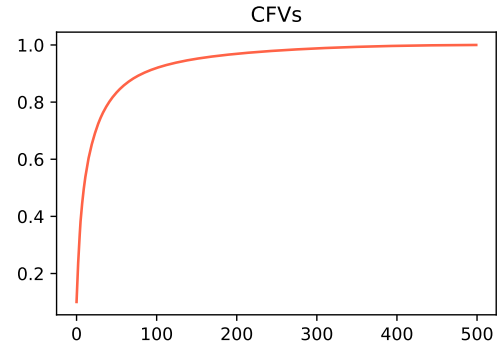
   For consistency in grading, define the reconstruction error as the squared L2 norm averaged over all data points.

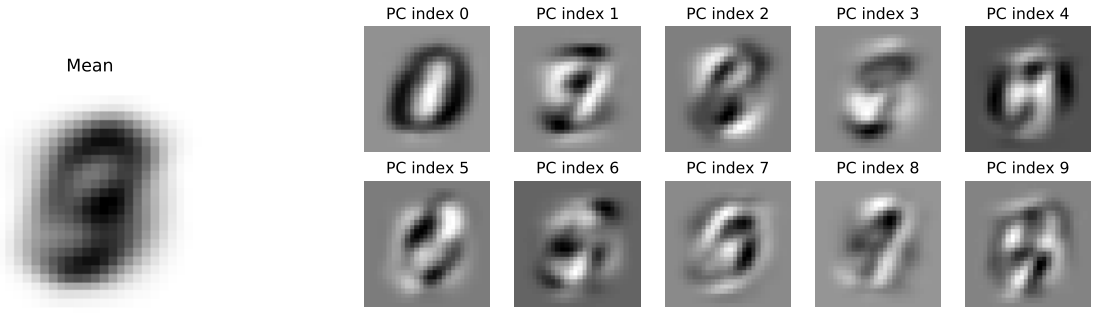
4. Suppose you took the original matrix of principle components that you found U and multiplied it by some rotation matrix R. Would that change the quality of the reconstruction error in the last problem? The interpretation of the components? Why or why not?

Solution

Plots:





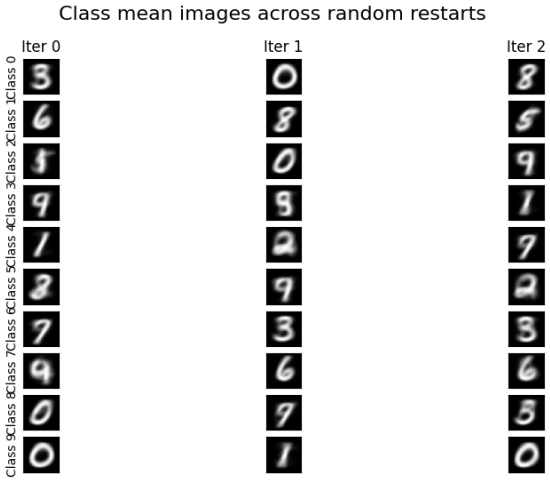


Code:

```
# Standardize data
def standardize(x):
    mean = x.mean(0)
    x_std = (x - mean)
    return x_std

def pca(x, n_comps=500):

    x = standardize(x)

    U, S, V_t = torch.linalg.svd(x, full_matrices=False)

    S = torch.square(S) / N
    top_eigvals, top_indexes = torch.topk(S, n_comps)

    top_pcomps = V_t[top_indexes.tolist()]

    return top_eigvals, top_pcomps


def calc_cfvs(eigvals):
    cum_frac_vars = torch.cumsum(eigvals,0) / torch.sum(eigvals,0)
    return cum_frac_vars
```

```
def calc_errs(x, pcomps):
    x = standardize(x)

    err_mean = torch.linalg.norm(x, axis=1, ord=2)

    pcomps_matrix = torch.matmul(torch.transpose(pcomps[:10],0,1), pcomps[:10])
    err_pcomp = torch.linalg.norm(x - torch.matmul(x, pcomps_matrix), axis=1, ord=2)

    err_mean = err_mean.square().mean(0)
    err_pcomp = err_pcomp.square().mean(0)

    return err_mean, err_pcomp
```

1. Almost all of the variance is explained by the first 500 components. As k increases, the size of the eigenvalues decrease, and so the proportion of variance explained by the kth eigenvalue decreases. Thus, the cumulative proportion of variance explained is concave (down) as k increases, i.e. increases less rapidly as k increases.

2. 

   K-means produces far sharper (less blurry) images than PCA. Intuitively, this makes sense as K-means works to cluster all the images in the same class (e.g. all of the 9s) and then averages the images in that cluster, such that the resulting centroid is a good prototype for images in that class (e.g. the cluster center looks like the average of many 9s as expected). By comparison, reconstructed images using PCA are blurrier because we lose spatial information by projecting the original images into a subspace with a lower dimension. As a result of having less information, reconstructed images could be really accurate, but they could also be really far away from what is the true clustering of identical images (e.g. ends up averaging reconstructed images that have very similar pixel values to real 9s, but also includes some images far away from the area in which there is a clustering of 9s). Thus, PCA gives us images that do not clearly distinguish between what is a number and what is a part of the background. Additionally, you can see that by averaging the different PCA images produced - i.e. creating a new linear combination of images - you could probably get closer to a sharper image (e.g. consider averaging PC index 2 and PC index 4 from the above in order to get a clearer 8). Thus, PCA does a good job but does not necessarily get its linear combinations right; K-means does a better job of producing clear images by placing them in the correct cluster, without having to weight them according to some linear combination, which it could get a bit wrong!

   In terms of similarities, it is clear that both PCA and K-means produce images that look like digits set on single-coloured backgrounds which have low variance. Although the backgrounds of the PCA

images are lighter than the K-means images, background edge pixels have low variance across all the original images, so PCA and K-means are both able to re-produce this in the resulting images which all have a low variance across the background edge pixels.

3.

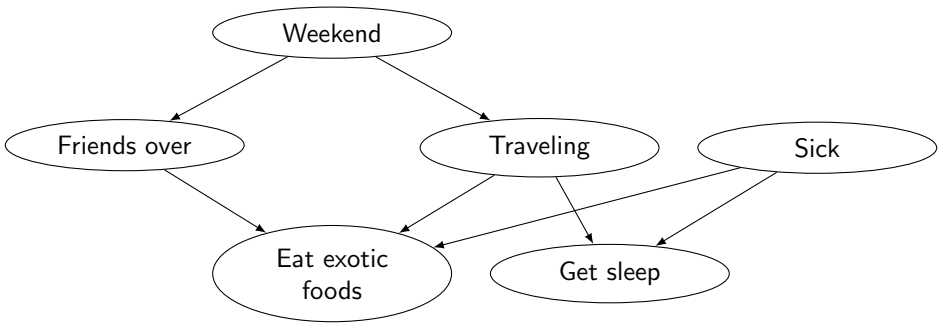
Reconstruction error (using mean)	3.436022×10^6
Reconstruction error (using mean and top 10 pcomps)	1.731315×10^6
K-means objective loss	1.3×10^10

The loss calculated from my implementation of K-means (in the table above) is not well suited for this comparison (it should be c. 2.6×10^6 - from office hours). We would expect the K-means loss to be between the reconstruction errors with and without using the principal components. Clearly, the reconstruction error using the mean should be the highest, since we are not doing any machine learning, just finding the squared loss between every image and the average of all the images. (If we had an algorithm that did worse than this, we would have wasted a lot of time!) The reconstruction error using the top 10 principal components should be the lowest because we know that the PCA algorithm minimizes the reconstruction error by projecting into a lower subspace using the principal components. By comparison, K-means should have a slightly higher loss (albeit still lower than the reconstruction error, not using the principal components) since it is placing images into clusters and then taking averages of those clusters. The loss compares the original images with these averages, which are arguably just more sophisticated averages than the one average of all the data used by the reconstruction error, not using the principal components. Thus, K-means gets us a local minimum ob the objective function, iterating in such a way that the loss always decreases, but never minimizing the objective function to the same extent as PCA with its reconstruction loss using the principal components.

4. Multiplying U by some rotation matrix R would not change the quality of the reconstruction error, since the eigenvectors in U define an orthogonal basis of the lower dimensional subspace, from which we are reconstructing the data. Rotating all of these eigenvectors still results in an orthogonal basis that defines this same subspace, and so we can still reconstruct the images using a linear combination of these rotated eigenvectors. However, the interpretation of these rotated components changes as they no longer point in the directions of maximum variance.

Problem 3 (Bayesian Networks, 10 pts)

In this problem we explore the conditional independence properties of a Bayesian Network. Consider the following Bayesian network representing a fictitious person's activities. Each random variable is binary (true/false).



The random variables are:

- Weekend: Is it the weekend?
- Friends over: Does the person have friends over?
- Traveling: Is the person traveling?
- Sick: Is the person sick?
- Eat exotic foods: Is the person eating exotic foods?
- Get Sleep: Is the person getting sleep?

For the following questions, $A \perp B$ means that events A and B are independent and $A \perp B|C$ means that events A and B are independent conditioned on C.

Use the concept of d-separation to answer the questions and show your work (i.e., state what the blocking path(s) is/are and what nodes block the path; or explain why each path is not blocked).

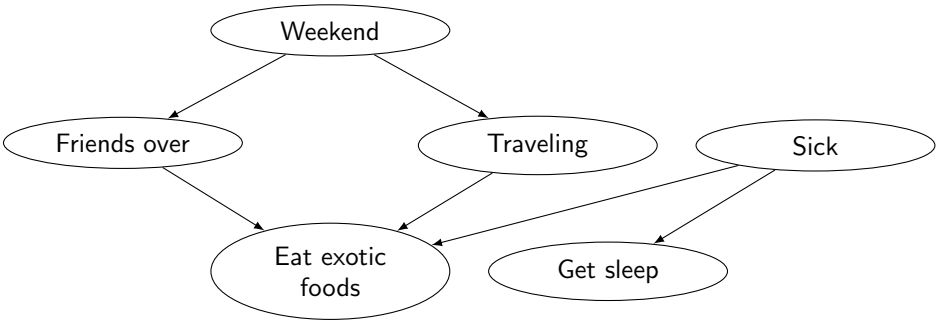
Example Question: Is Friends over $\perp$ Traveling? If NO, give intuition for why.

Example Answer: NO. The path from Friends over – Weekend – Traveling is not blocked following the d-separation rules as we do not observe Weekend. Thus, the two are not independent.

Actual Questions:

1. Is Weekend $\perp$ Get Sleep? If NO, give intuition for why.
2. Is Sick $\perp$ Weekend? If NO, give intuition for why.
3. Is Sick $\perp$ Friends over | Eat exotic foods? If NO, give intuition for why.
4. Is Friends over $\perp$ Get Sleep? If NO, give intuition for why.
5. Is Friends over $\perp$ Get Sleep | Traveling? If NO, give intuition for why.
6. Suppose the person stops traveling in ways that affect their sleep patterns. Travel still affects whether they eat exotic foods. Draw the modified network. (Feel free to reference the handout file for the commands for displaying the new network in LaTeX).
7. For this modified network, is Friends over $\perp$ Get Sleep? If NO, give an intuition why. If YES, describe what observations (if any) would cause them to no longer be independent.

Solution

1. NO. The path from Weekend – Traveling – Get sleep is not blocked following the d-separation rules as we do not observe Traveling. Thus, the two are not independent.
2. YES. The path from Sick – Get sleep – Traveling – Weekend is blocked following the d-separation rules as we do not observe Get Sleep. The other path from Sick – Eat exotic foods – Friends over – Weekend is blocked following the d-separation rules as we do not observe Eat exotic foods. Thus, the two are independent.
3. NO. The path from Sick – Eat exotic foods – Friends over is not blocked following the d-separation rules as we now observe Eat exotic foods. Thus, the two are not independent given Eat exotic foods.
4. NO. The path from Get sleep – Traveling – Weekend – Friends over is not blocked following the d-separation rules as we do not observe Traveling nor Weekend. Thus, the two are not independent.
5. YES. The path from Get sleep – Traveling – Weekend – Friends over is blocked following the d-separation rules as we now observe Traveling. Likewise, the other path from Get sleep – Traveling – Eat exotic foods – Friends over is blocked following the d-separation rules by observing Traveling. Finally, the path from Get sleep – Sick – Eat exotic foods – Friends over is blocked following the d-separation rules as we do not observe Eat exotic foods. Thus, the two are independent given Traveling.
6. 

7. YES. The path from Get sleep – Sick – Eat exotic foods – Friends over is blocked following the d-separation rules as we do not observe Eat exotic foods. Thus, the two are independent.

Name

Ben Ray

Collaborators and Resources

Whom did you work with, and did you use any resources beyond cs181-textbook and your notes?
CS 181 Office Hours
Ty Geri
Josh Michels

Calibration

Approximately how long did this homework take you to complete (in hours)?
14