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 scale 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 write out the data likelihood by marginalizing over all possible values of z_n for each data point:

$$p(\{x_n\};\theta,\{\beta_k\}) = p(\{x_n\}|\{z_n\},\{\beta_k\})p(\{z_n\}|\theta)$$

   iid assumption on the data points + LOTP:

$$= \prod_{n=1}^{N}\sum_{k=1}^{K} p(x_n|\beta_k, z_n = C_k)p(z_n = C_k|\theta_k)$$

   Plugging in the gamma PDF as well as the categorical probabilities:

$$= \prod_{n=1}^{N}\sum_{k=1}^{K} \frac{\beta_k^{\alpha} x_n^{\alpha-1} e^{-\beta x_i}}{\Gamma(\alpha)}\theta_k$$

   Now for the log-likelihood, we can just take the log of the expression above:

$$\log p(\{x_n\},\theta,\{\beta_k\}) = \sum_{n=1}^{N}\log\left(\sum_{k=1}^{K}\frac{\beta_k^{\alpha} x_n^{\alpha-1} e^{-\beta_k x_n}}{\Gamma(\alpha)}\theta_k\right)$$

   Now we're stuck with an intractable expression: there is a sum over all values of k inside the logarithm, so after differentiating, we won't be able to separate out individual β_k and θ_k terms and solve for the MLE explicitly.

2. Now we assume knowledge of the latents z_n. Note that z_n is a vector of length k which is one-hot-encoded, meaning that the element $z_{n,k} = 1$ if z_n falls into C_k and all other elements are zero. For each observation, we only want to add log likelihood for the correct category, so the likelihood function looks like:

$$p(\mathcal{D},\theta,\{\beta_k\}) = \prod_{n=1}^{N}\sum_{k=1}^{K} p(x_n|\beta_k, z_n = C_k)p(z_n = C_k|\theta_k)I[z_n = C_k]$$

   We can rewrite the indicator function using the power trick: only the desired term gets a power of 1, while all other terms are raised to the power of zero and thus don't affect a product:

$$= \prod_{n=1}^{N}\prod_{k=1}^{K} (p(x_n|\beta_k, z_n = C_k)p(z_n = C_k|\theta_k))^{z_{n,k}}$$

   Taking the negative log:

$$\mathcal{L}(\theta,\{\beta_k\}) = -\sum_{n=1}^{N}\sum_{k=1}^{K} z_{n,k}\left(\log p(x_n|\beta_k, z_n = C_k) + \log p(z_n = C_k|\theta_k)\right)$$

   Plugging in the functional forms from part 1):

$$= -\sum_{n=1}^{N}\sum_{k=1}^{K} z_{n,k}\left(\log \frac{\beta_k^\alpha x_n^{\alpha-1}e^{-\beta_k x_n}}{\Gamma(\alpha)} + \log\theta_k\right)$$

$$= -\sum_{n=1}^{N}\sum_{k=1}^{K} z_{n,k}\left(\alpha\log(\beta_k) + (\alpha-1)\log(x_n) - \beta_k x_n - \log(\Gamma(\alpha)) + \log\theta_k\right)$$

3. For this part, note that Bayes' rule gives us:

$$p(\mathbf{z_n}|x_n, \theta, \{\beta_k\}) = \frac{p(x_n|\mathbf{z_n}, \theta, \{\beta_k\})p(\mathbf{z_n}|\theta, \{\beta_k\})}{p(x_n|\theta, \{\beta_k\})} \propto p(x_n|\mathbf{z_n}, \theta, \{\beta_k\})p(\mathbf{z_n}|\theta, \{\beta_k\})$$

Where we can drop the proportionality constant because we can simply standardize results to sum to 1 across the K possible values of $\mathbf{z_n}$.

Now using the power trick from part 2), we can rewrite this product as:

$$p(\mathbf{z_n}|x_n, \theta, \{\beta_k\}) \propto \prod_{k=1}^{K}(p(x_n|\mathbf{z_n} = C_k, \theta, \{\beta_k\})p(\mathbf{z_n}|\theta, \{\beta_k\}))^{z_{n,k}}$$

And plugging in functional forms:

$$= \prod_{k=1}^{K}\left(\frac{\beta_k^\alpha x_n^{\alpha-1}e^{-\beta_k x_n}}{\Gamma(\alpha)}\theta_k\right)^{z_{n,k}}$$

For a specific one-hot-encoded vector C_k, with a 1 in the k-th place and 0s everywhere else, this gives us:

$$q_{n,k} = p(\mathbf{z_n} = C_k|x_n, \theta, \{\beta_k\}) \propto \frac{\beta_k^\alpha x_n^{\alpha-1}e^{-\beta_k x_n}}{\Gamma(\alpha)}\theta_k$$

So finally standardizing to make sure that $\sum_k q_{n,k} = 1$ (so we get a valid probability distribution), we get:

$$q_{n,j} = \frac{1}{\sum_{k=1}^{K}\frac{\beta_k^\alpha x_n^{\alpha-1}e^{-\beta_k x_n}}{\Gamma(\alpha)}\theta_k}\frac{\beta_j^\alpha x_n^{\alpha-1}e^{-\beta_j x_n}}{\Gamma(\alpha)}\theta_j = \frac{1}{\sum_{k=1}^{K}\beta_k^\alpha e^{-\beta_k x_n}\theta_k}\beta_j^\alpha e^{-\beta_j x_n}\theta_j$$

$$\mathbf{q_n} = \frac{1}{\sum_{k=1}^{K}\beta_k^\alpha e^{-\beta_k x_n}\theta_k}\begin{bmatrix}\beta_1^\alpha e^{-\beta_1 x_n}\theta_1 \\ \cdots \\ \beta_K^\alpha e^{-\beta_K x_n}\theta_K\end{bmatrix}$$

4. (a) Note that $q_{n,k}$ is the probability that $\mathbf{z_n}$ is the vector C_k with a one in the k-th element. Then using the result for the complete data log likelihood from 2) as well the definition of discrete expectation:

$$E_{\mathbf{z_n}}\left[\log p(\mathcal{D}, \theta, \{\beta_k\})\right] = \sum_{n=1}^{N}\sum_{k=1}^{K}\log p(\mathcal{D}, \theta, \{\beta_k\}|\mathbf{z_n} = C_k)p(z_n = C_k)$$

$$= \sum_{n=1}^{N}\sum_{k=1}^{K} q_{n,k} \log p(\mathcal{D}, \theta, \{\beta_k\} | \mathbf{z_n} = C_k)$$

$$= \sum_{n=1}^{N}\sum_{k=1}^{K} q_{n,k} \left(\alpha \log(\beta_k) + (\alpha - 1)\log(x_n) - \beta_k x_n - \log(\Gamma(\alpha)) + \log \theta_k\right)$$

(b) First, note that whenever we're optimizing a log likelihood with respect to θ, we can drop all additive terms that don't depend on θ since they won't change the arg max. Therefore, we're interesting in maximizing the objective:

$$\sum_{n=1}^{N}\sum_{k=1}^{K} q_{n,k} \left(\log \theta_k\right)$$

with respect to the constraint:

$$\sum_{k=1}^{K} \theta_k = 1$$

Writing down a Lagrangian with a multiplier λ, taking partials, and setting them equal to zero:

$$L(\mathcal{D}, \theta, \lambda) = \sum_{n=1}^{N}\sum_{k=1}^{K} q_{n,k} \left(\log \theta_k\right) - \lambda \left(\sum_{k=1}^{K} \theta_k - 1\right)$$

$$\frac{\partial L}{\partial \theta_k} = \sum_{n=1}^{N} \frac{q_{n,k}}{\theta_k} - \lambda = 0 \quad (1)$$

$$\frac{\partial L}{\partial \lambda} = \sum_{k=1}^{K} \theta_k - 1 = 0 \quad (2)$$

From (1), we get:

$$\hat{\theta}_k = \frac{\sum_{n=1}^{N} q_{n,k}}{\lambda}$$

So we can express all θ_k in terms of θ_1:

$$\lambda = \frac{\sum_{n=1}^{N} q_{n,1}}{\hat{\theta}_1}$$

$$\hat{\theta}_k = \frac{\sum_{n=1}^{N} q_{n,k}}{\frac{\sum_{n=1}^{N} q_{n,1}}{\hat{\theta}_1}} = \frac{\sum_{n=1}^{N} q_{n,k}}{\sum_{n=1}^{N} q_{n,1}} \hat{\theta}_1$$

Substituting into (2):

$$\sum_{k=1}^{K} \frac{\sum_{n=1}^{N} q_{n,k}}{\sum_{n=1}^{N} q_{n,1}} \hat{\theta}_1 = 1$$

$$\frac{\hat{\theta}_1}{\sum_{n=1}^{N} q_{n,1}} \sum_{k=1}^{K} \sum_{n=1}^{N} q_{n,k} = 1$$

Since we're allowed to swap the order of finite summation, and since $\sum_{k=1}^{K} q_{n,k} = 1$ by design:

$$\hat{\theta}_1 = \frac{\sum_{n=1}^{N} q_{n,1}}{N}$$
$$\hat{\theta}_k = \frac{\sum_{n=1}^{N} q_{n,k}}{\sum_{n=1}^{N} q_{n,1}} \frac{\sum_{n=1}^{N} q_{n,1}}{N} = \frac{\sum_{n=1}^{N} q_{n,k}}{N}$$

These values make perfect sense! Our best estimate for the probability of $\mathbf{z_n}$ falling into category k is just the average probability of observations having their latent variable $\mathbf{z_n}$ equal C_k, which is precisely why the expression is a simple mean.

(c) For this part, we can drop all additive terms that don't depend on β_k before optimizing, so we are looking to solve unconstrained maximization for the objective:

$$\sum_{n=1}^{N} \sum_{k=1}^{K} q_{n,k} \left(\alpha \log(\beta_k) - \beta_k x_n\right)$$

Taking the first orded condition wrt β_k:

$$\frac{\partial E_{\mathbf{z_n}} \left[\log p(\mathcal{D}, \theta, \{\beta_k\})\right]}{\partial \beta_k} = \sum_{n=1}^{N} q_{n,k}(\frac{\alpha}{\beta_k} - x_n) = 0$$

$$\hat{\beta}_k = \alpha \frac{\sum_{n=1}^{N} q_{n,k}}{\sum_{n=1}^{N} q_{n,k} x_n}$$

This expression makes sense, but is a little more subtle. Note that the mean of a $Gamma(\alpha, \beta)$ distribution is $\frac{\alpha}{\beta}$. We can therefore view the expression for the MLE for β_k as a weighted mean of the observations, where weighting comes from the empirical probabilities $\mathbf{q_n}$ which express our certainty about a given observation falling into the k-th category.

5. If this were a discriminative problem, I'd start by writing down the complete data log likelihood from part 2):

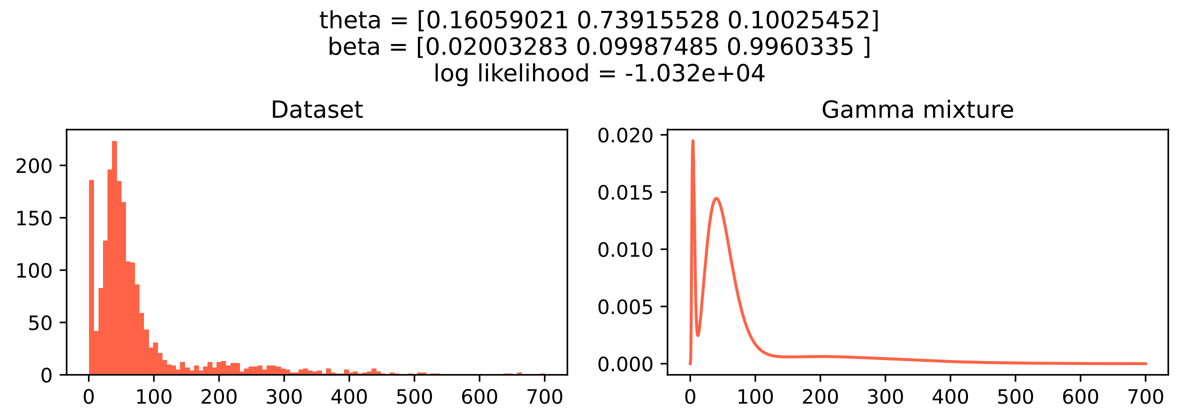
$$\log p(\mathcal{D}, \theta, \{\beta_k\}) = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{n,k} \left(\log p(x_n | \beta_k, z_n = C_k) + \log p(z_n = C_k | \theta_k)\right)$$

Given knowledge of the true parameters $z_{n,k}$, this likelihood could be tractably optimized, so I'd use first-order conditions to solve for the MLEs $\hat{\theta}$ and $\{\beta_k\}$; note that these would be the same as the estimates derived in part 4(b) and 4(c). Subsequently, we would regard these as the "true" parameters of the model, and for a new data point x^*, we'd compute:

$$\mathbf{q} = \begin{bmatrix} p(\mathbf{z_1}|x^*, \theta, \{\beta_k\}) \\ \dots \\ p(\mathbf{z_K}|x^*, \theta, \{\beta_k\}) \end{bmatrix}$$

By treating the parameters as fixed and computing $p(\mathbf{z}^*|x^*, \theta, \{\beta_k\}) \propto p(x^*|\mathbf{z}^*, \theta, \{\beta_k\})p(\mathbf{z_n}|\theta, \{\beta_k\})$ just like in part 3). These would be the probabilities of a given data point falling into a category given the learned parametric model for the data. We would then assign the category for x^* by $\arg\max q_k$.

6. Plot:



Code:

```python
def e_step(theta, betas):
    q = np.zeros((x.shape[0], K))
    for n in range(x.shape[0]):
        lq = gamma.logpdf(x[n], alpha, scale=1/betas) + np.log(theta)
        lq  = lq - logsumexp(lq)
        q[n,] = np.exp(lq)
    return q


def m_step(q):
    theta_hat = np.mean(q, axis = 0)
    beta_hats = alpha*q.sum(axis=0)/(x*q).sum(axis=0)
    return theta_hat, beta_hats


def log_px(x, theta, betas):
    p = (gamma.pdf(x, alpha, scale = 1/betas) * theta).sum(axis=1)
    return np.log(p)


def run_em(theta, betas, iterations=1000):
    t = theta
    b = betas
    qu = np.zeros((x.shape[0], K))
    for i in range(iterations):
        qu = e_step(t, b)
        t, b = m_step(qu)
    return t, b
```

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.

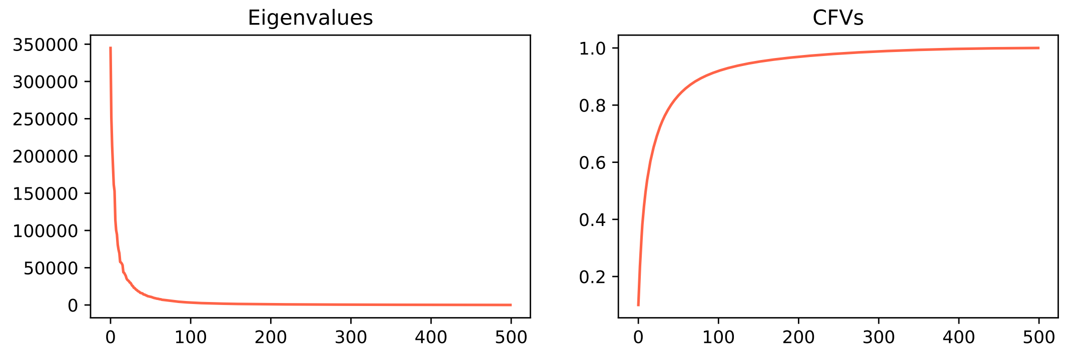
1. Compute the PCA. Plot the eigenvalues corresponding to the most significant 500 components in order from most significant to least. Make another plot that describes the cumulative proportion of variance explained by the first k most significant components for values of k from 1 through 500. How much 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.

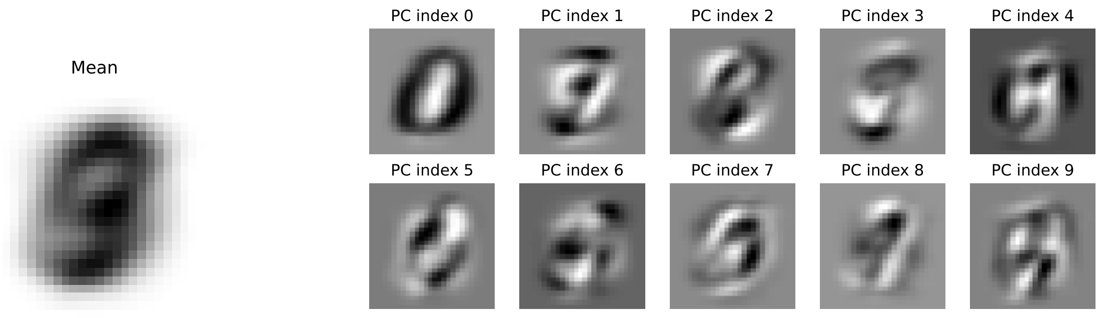
   Reminder: Center the data before performing PCA
3. Compute the reconstruction error on the data set using the mean image of the dataset. Then compute the reconstruction error using the first 10 principal components. How do these errors compare to the final objective loss achieved by using K-means on 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:

```
def pca(x, n_comps=500):
    X = x - x.mean(axis = 0)
    sv = torch.linalg.svd(X)
    top_eigvals = (sv[1]**2 /N)[:n_comps]
    top_pcomps = sv[2][:n_comps,]
    return top_eigvals, top_pcomps


def calc_cfvs(eigvals):
    cum_var = eigvals.cumsum(0)
    cum_frac_vars = cum_var / cum_var[-1]
    return cum_frac_vars


def calc_errs(x, pcomps):
    err_mean = torch.sum((x-x.mean(axis=0))**2)/N
    err_pcomp = 0
    for j in range(x.shape[0]):
        scores = pcomps @ x[j,]
        proj = torch.zeros(x.shape[1])
        for s in range(scores.shape[0]):
            proj += scores[s] * pcomps[s,]
        err_pcomp += torch.sum((x[j,] - proj)**2)/N
    return err_mean, err_pcomp
```

1. The first 500 principal components explain almost all of the variance in the data (proportion ≈ 1.0). Note that the eigenvalues for the first cca 50 principal components are much larger than the subsequent

eigenvalues, meaning that the marginal impact of one more principal component on explained variance rapidly decreases. In fact, the first 50 principal components explain around 80% of all variance in the data.

2. The principal components are much less similar to actual digits than the cluster means in KMeans. A significant reason for this might be the centering of data required for PCA: the clusters represent deviations from the mean image, we would therefore expect an image for a digit like 0 to be composed of the mean image and a large score times the principal component corresponding to the zero digit. This is why the principal components themselves don't look like crisp digits.

3. Look. This question is really unclear and on the day when this pset was due, there was no staff Ed answer despite 4+ questions. Therefore, I don't really know how to compare anything to my KMeans solution.

   To be able to do something at all, I chose to 1) run my KMeans algorithm from HW4 on the same data set used for this problem, 2) divide the resulting loss (which is the RSS) by the number of data points. Here are my results:

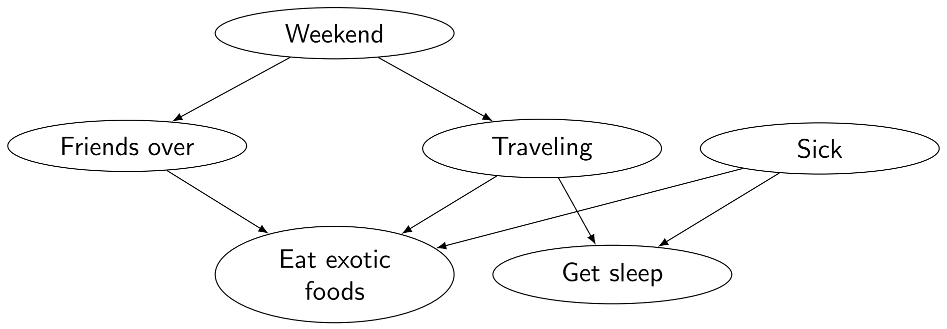
Method	Loss
Reconstruction using dataset mean	3.436024e+06
Reconstruction with 10 principal components	2.077421e+03
KMeans loss with 10 clusters	2.521619e+6

   Thus, reconstruction using 10 principal components seems to do much better than the alternatives. The KMeans loss is smaller than, but on the same order of magnitude as, the loss from reconstruction using the mean.

4. The quality of reconstruction would not change, since the orthogonality structure of principal components would not change, and with different principal components scores, we could still find a reconstruction of each data point as a linear combination of the PCs. However, the interpretation of the principal components would change, since they would no longer point in the directions with 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).

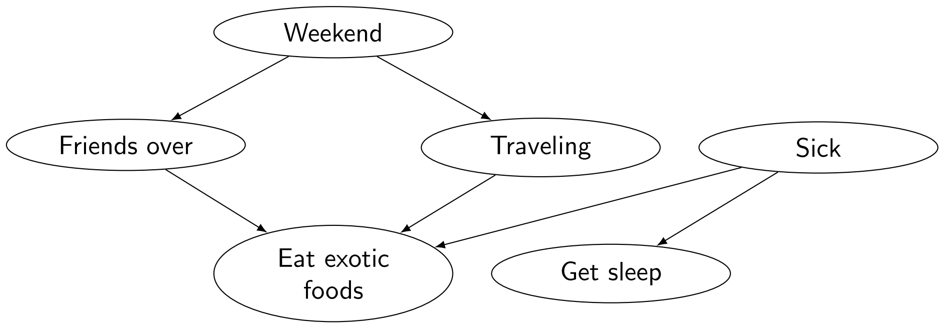
Example Question: Is Friends over $\perp$ Traveling? If NO, give intuition for why.

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. These variables are NOT independent. Not observing Traveling creates an unblocked information path between Weekend and Get sleep. Intuitively, observing Get sleep allows inference on Traveling, and in turn, on Weekend.
2. Yes, these variables are independent. Not observing Eat exotic foods blocks the path between them.
3. These variables are NOT independent. Observing Eat exotic foods (a child node for both) creates an unblocked path between Sick and Friends over. Intuitively, observing Eat exotic foods and Friends over gives us intuition for the contribution of Sick.
4. These variables are NOT independent. Not observing Weekend and Traveling makes for an unblocked flow of information, allowing inferences between Friends over and Get sleep.
5. Yes, the variables are conditionally independent, since observing Traveling blocks the path between them (i.e., Friends over only affects Get sleep only through Traveling).
6. 

7. Yes, in this network, we have independence between Friends over and Get sleep, since Eat exotic foods is not observed. If the variable was observed, it would unblock a path between Friends over and Get sleep.

Name

Matej Cerman

Collaborators and Resources

Whom did you work with, and did you use any resources beyond cs181-textbook and your notes?

Arpit Bhatte, no resources

Calibration

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

15 hours