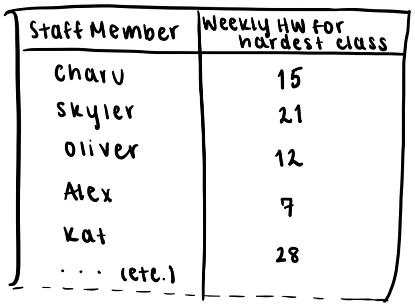
CS 181 Spring 2022 Section 7 Solutions
Mixture Models and Principal Component Analysis

1 Mixture Models

1.1 Motivation

Textbook sections 9.1, 9.2.
A *mixture model* is a type of probabilistic model for unsupervised learning. Let's motivate the need for mixture models through an example:

Example: You run a poll of the CS181 teaching staff asking how many hours a week they spend doing work for their hardest CS class. There are only 3 CS classes at Harvard and every member of the teaching staff is taking at least one of the three. *Using this, can you figure out which staff members are in the same class?* Here, we have some observed data $\{x_n\}_{n=1}^{N}$ corresponding to the hours reported by each staff member. We also know that each individual observation has a discrete *latent variable* $\mathbf{z}_n$ that determines the data generating process. In this case, $\mathbf{z}_n$ is the class that that the nth staff member reported hours for. Note that while z_n is *unknown*, it influences the observed data (ie. some classes have a heavier workload than others, so the CS class a staff member is taking impacts the hours they report).



Staff Member	Weekly HW for hardest class
Charu	15
Skyler	21
Oliver	12
Alex	7
Kat	28
. . . (etc.)	

Data gathered

There are K possible values for each $\mathbf{z}_n$, denoted $\{C_k\}_{k=1}^{K}$ where each C_k is a one-hot encoded vector of length K. In our example, $K = 3$ with $C_1 = [1, 0, 0]$ corresponding to the class CS124, $C_2 = [0, 1, 0]$ corresponding to the class CS61, and $C_3 = [0, 0, 1]$ corresponding to the class CS175. We can think of every observation $\mathbf{x}_n$ as being generated by the following process:

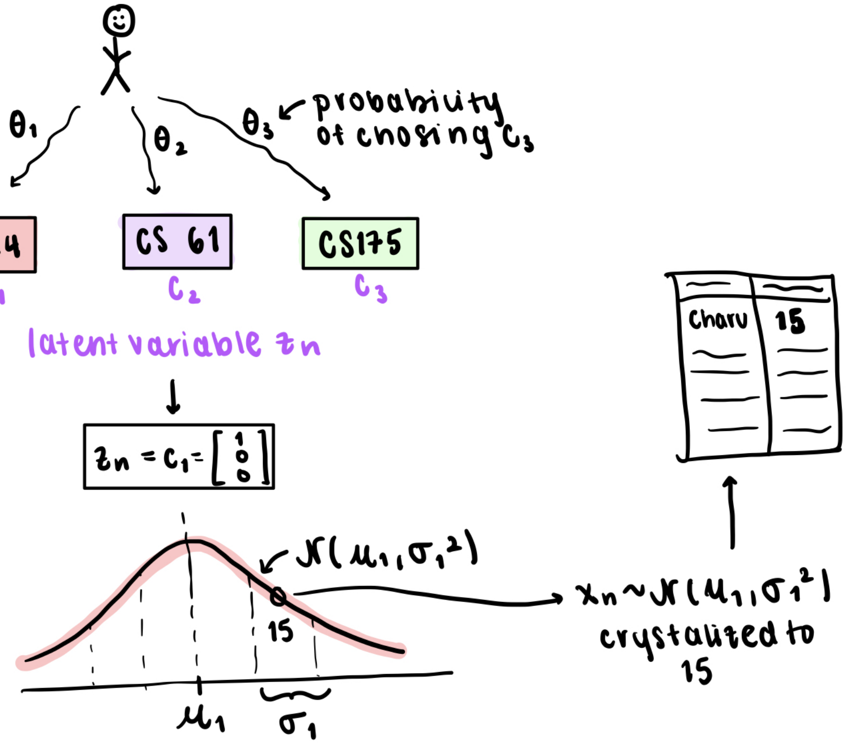
- Sample latent class $\mathbf{z}_n$ from $\boldsymbol{\theta}$, the categorical distribution over $\{C_k\}_{k=1}^{K}$ s.t. $p(\mathbf{z} = C_k; \boldsymbol{\theta}) = \theta_k$. Call this sampled latent class C_S.

- Given that $\mathbf{z}_n = C_S$, sample x_n from the distribution

$$p(\mathbf{x}|\mathbf{z} = C_S; \mathbf{w})$$

This conditional distribution is a modeling assumption (which means we will give it to you in this class), and is specified using unknown parameters $\mathbf{w}$.

In our example, it might make sense to assume $x \sim p(x|\mathbf{z} = C_k) = \mathcal{N}(x; \mu_k, \sigma_k)$, where μ_k, Σ_k are the unknown mean and covariance of the number of hours the k-th class takes a week.



The hypothetical data generating process

Comprehension Question: In our example, what is an intuitive explanation of what the vector $\boldsymbol{\theta}$ represents?
Solution: The proportion of people taking each class

2 Expectation Maximization

Textbook sections 9.3, 9.4.

Expectation maximization is a general technique for maximum-likelihood estimation used primarily for models with latent variables. Here we will show how to use EM to train a mixture model, but EM is also used for a variety of other models!

Returning to our example of weekly homework hours, what are our unknowns?

1. The probabilities $\theta_1, ..., \theta_k$ with which we sample $\mathbf{z}_n$
2. The parameters $\mathbf{w}$ of $p(x|\mathbf{z} = C_s; \mathbf{w})$ (in our example these unknown parameters would be $\mu_1, \mu_2, \mu_3, \sigma_1, \sigma_2, \sigma_3$).

Our goals are to compute the MLE for the parameters above, and to estimate the latent variable $\mathbf{z}_n$ for each x_n. Note that once we have the MLE for $\mathbf{w}, \boldsymbol{\theta}$, it is easy to estimate $\mathbf{z}_n$ since we can just find the class C_k that maximizes $p(\mathbf{z}_n = C_k; \mathbf{w}, \theta) \propto p(x_n|\mathbf{z}; \mathbf{w}, \boldsymbol{\theta})p(\mathbf{z}; \boldsymbol{\theta})$.

2.1 The EM Algorithm

The likelihood of a datapoint can be written as

$$p(\mathbf{x}; \mathbf{w}, \boldsymbol{\theta}) = \sum_{\mathbf{z} \in Z} p(\mathbf{x}, \mathbf{z}; \mathbf{w}, \boldsymbol{\theta})$$

Unfortunately calculating the MLE is often computationally intractable, because the log-likelihood is:

$$\log p(\mathbf{x}; \mathbf{w}, \boldsymbol{\theta}) = \log \sum_{\mathbf{z} \in Z} p(\mathbf{x}, \mathbf{z}; \mathbf{w}, \boldsymbol{\theta}) \tag{1}$$

There is no closed form for the MLE of the log-likelihood because it is the log of a sum of expressions. We know the form of the model $p(\mathbf{x}, \mathbf{z}; \mathbf{w}, \boldsymbol{\theta})$, but in general we cannot solve for the $(\mathbf{w}, \boldsymbol{\theta})$ which maximize the likelihood $p(\mathbf{x}; \mathbf{w}, \boldsymbol{\theta})$ in closed form.

Since finding the MLE directly is difficult, we will use expectation maximization: an *approximate iterative approach.* The steps of the algorithm are:

1. Initialize $\mathbf{w}^{(0)}, \boldsymbol{\theta}^{(0)}$ randomly.
2. *(E-step)* Given the parameters from (1) find the likelihood that $\mathbf{z}_n = C_k$ for each C_k. Store these probabilities in a vector q_n where the kth element of the vector is the probability that $z_n = C_k$. This is called a *soft assignment*:

$$q_{n,k} := p(\mathbf{z}_n = C_k|\mathbf{x}_n; \mathbf{w}^{(t)}, \boldsymbol{\theta}^{(t)}) \propto p(\mathbf{x}_n|\mathbf{z}_n = C_k; \mathbf{w}^{(t)})p(\mathbf{z}_n = C_k; \boldsymbol{\theta}^{(t)}) \tag{2}$$

3. *(M-step)* Choose the value of $\mathbf{w}^{(t+1)}, \boldsymbol{\theta}^{(t+1)}$ that maximizes the *expected complete data log likelihood.*

$$\mathbf{w}^{(t+1)}, \boldsymbol{\theta}^{(t+1)} = \arg\max_{\mathbf{w}, \boldsymbol{\theta}} \mathrm{E}_{\mathbf{Z}|\mathbf{X}} \left[\sum_{n=1}^{N} \log p(\mathbf{x}_n, \mathbf{z}_n; \mathbf{w}, \boldsymbol{\theta}) \right] \tag{3}$$

Let's expand out our expression for the expected complete data log likelihood:

$$\begin{aligned}
\mathrm{E}_{\mathbf{Z}|\mathbf{X}}\left[\log(p(\mathbf{X}|\mathbf{Z})\right] &= \mathrm{E}_{Z|X}\left[\sum_{n=1}^{N}\log p(\mathbf{x}_n|\mathbf{z}_n;\mathbf{w}) + \log p(\mathbf{z}_n;\theta)\right] \\
&= \sum_{n=1}^{N}\mathrm{E}_{Z|X}\left[\log p(\mathbf{x}_n|\mathbf{z}_n;\mathbf{w}) + \log p(\mathbf{z}_n;\theta)\right]
\end{aligned}$$

Taking the expectation gives us

$$\begin{aligned}
&\sum_{n=1}^{N}\sum_{k=1}^{K} p(\mathbf{z}_n = C_k|\mathbf{x}_n)\left[\log p(\mathbf{x}_n|\mathbf{z}_n = C_k;\mathbf{w}) + \log p(\mathbf{z}_n = C_k;\boldsymbol{\theta})\right] \\
&= \sum_{n=1}^{N}\sum_{k=1}^{K} q_{n,k}\left[\log p(\mathbf{x}_n|\mathbf{z}_n = C_k;\mathbf{w}) + \log p(\mathbf{z}_n = C_k;\boldsymbol{\theta})\right]
\end{aligned}$$

Taking the derivative of this expression wrt $\mathbf{w}$, setting equal to 0, and solving, will give us its MLE. We can do the same thing for .

4. Repeat steps 2 and 3 until the MLE's of $\mathbf{w}$ and $\boldsymbol{\theta}$ converge.

2.2 Example: Gaussian Mixture Modeling

Lecture 14 and textbook section 9.5.

Let's think about how we would apply EM to our problem. We already identified that our latent variable $\mathbf{z}_n$ for a staff member n is the CS class that they are actually taking. We assume that the $\mathbf{z}_n$'s are sampled according to a categorical distribution with parameters $\theta_1, \theta_2, \theta_3$ and that *given* we know what class C_k a student is taking the number of hours they report will be $\sim \mathcal{N}(\mu_k, \sigma_k^2)$. We can now run the EM algorithm:

1. Randomly initialize $\theta_1, \theta_2, \theta_3, \mu_1, \mu_2, \mu_3, \sigma_1, \sigma_2, \sigma_3$.

2. Calculate q_n:

$$q_n = \begin{bmatrix} p(z_n = C_1|x_n) \\ p(z_n = C_2|x_n) \\ p(z_n = C_3|x_n) \end{bmatrix} \propto \begin{bmatrix} \theta_1\mathcal{N}(\mu_1, \sigma_1) \\ \theta_2\mathcal{N}(\mu_2, \sigma_2) \\ \theta_3\mathcal{N}(\mu_3, \sigma_3) \end{bmatrix} \tag{4}$$

This is our new estimate of the distribution of $\mathbf{z}_n$ given the data and most recent estimate of $\boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\sigma}$.

3. Find the expected complete data log-likelihood:

$$\mathrm{E}_{\mathbf{Z}|\mathbf{X}}\left[\mathcal{L}\right] = \sum_{n=1}^{N}\sum_{k=1}^{K} q_{n,k}\ln\theta_k + q_{n,k}\ln\mathcal{N}(x_n;\mu_k,\sigma_k) \tag{5}$$

and then optimize it for each of the parameters. Since we require that $\sum_k \theta_k = 1$, we must use Lagrange multipliers to incorporate this constraint before optimizing our parameters. Thus, the equation we want to optimize is:

$$\sum_{n=1}^{N} \sum_{k=1}^{K} q_{n,k} \ln \theta_k + q_{n,k} \ln \mathcal{N}(x_n; \boldsymbol{\mu}_k, \boldsymbol{\sigma}_k) - \lambda \left(\sum_{k=1}^{K} \theta_k - 1 \right)$$

Now lets find the MLE of θ_k. We first take the derivative of the above expression wrt θ_k:

$$\sum_{n=1}^{N} q_{n,k} \cdot \frac{1}{\theta_k} - \lambda = 0 \implies \frac{\sum_{n=1}^{N} q_{n,k}}{\lambda} = \theta_k$$

Note that this implies $\sum_{k=1}^{K} \theta_k = \frac{\sum_{k=1}^{K} \sum_{n=1}^{K} q_{n,k}}{\lambda}$. By the constraint on mixture probabilities, $\sum_{k=1}^{K} \theta_k = 1$, so $\frac{\sum_{k=1}^{K} \sum_{n=1}^{K} q_{n,k}}{\lambda} = 1 \implies \lambda = \sum_{k=1}^{K} \sum_{n=1}^{N} q_{n,k}$. Plugging this back in we get

$$\theta_k = \frac{\sum_{n=1}^{N} q_{n,k}}{\sum_{k=1}^{K} \sum_{n=1}^{N} q_{n,k}} = \frac{\sum_{n=1}^{N} q_{n,k}}{N}$$

We can follow a similar process to find the MLE for μ_k and σ_k:

$$\mu_k^{(t+1)} = \frac{\sum_{n=1}^{N} q_{n,k} x_n}{\sum_{n=1}^{N} q_{n,k}} \tag{6}$$

$$\sigma_k^{(t+1)} = \frac{\sum_{n=1}^{N} q_{n,k} (x_n - \mu_k^{(t+1)})^2}{\sum_{n=1}^{N} q_{n,k}} \tag{7}$$

4. We repeat steps (2) and (3) until convergence.

GMM's with multivariate normals are discussed in section 9.5 of the textbook.

2.3 Example: Modeling Biased Coins with a Binomial Mixture Model

We've seen one case of mixtures of Gaussians, but we can consider mixtures of any distribution. In this example, we'll take a look at EM for a binomial mixture model. To get started, we consider a mixture of Bernoulli model, where $x_n \sim p(x_n | z_n = C_k) = \text{Bern}(x_n; p_k)$.

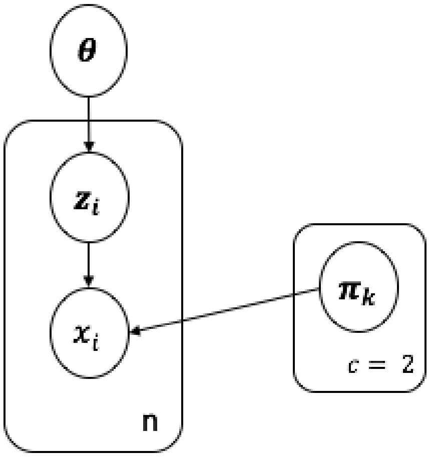
Consider a setup where we have 2 biased coins C_1 and C_2, where $Pr(C_1 = 1) = \pi_1$ and $Pr(C_2 = 1) = \pi_2$.

Data points x_n are generated by:

- First, flip another biased coin C_z.

- If C_z is heads, then x_n is the outcome of flipping C_1.
- Otherwise, if C_z is tails, then x_n is the outcome of flipping C_2.

We can visualize this setup with the following diagram:



We wish to do inference to learn the unknown parameters of the coins (π_1, π_2), but the only data we're given is the outcomes of the flips (the x_n's).[1]

Exercise 1. *In this example, what is a reasonable choice for the latent variables z_n?*

Solution. The latent variable z_n should represent whether coin C_1 or C_2 was flipped to generate x_n. This choice is reasonable because we know the distribution $p(x_n|z_n; \mathbf{w}, \theta)$ when z_n is the outcome of the first coin flip of C_z. □

To be consistent with Textbook Example 9.4.5, which uses the same model for the mixture of multinomials, we'll let $\mathbf{x}_n$ be a one-hot vector s.t. $\mathbf{x}_{n,1} = 1$ if the result of coin flip n was heads; $\mathbf{x}_{n,2} = 1$ otherwise. $\mathbf{z}_n$ is a one-hot vector (of size 2) indicating which coin was flipped to generate $\mathbf{x}_n$.

We'll denote the vector of probabilities for C_z used to choose between coins as $\boldsymbol{\theta} \in [0, 1]^2$, where θ_1 is the probability we'll pick C_1, and θ_2 for C_2. Finally, we'll use $\boldsymbol{\pi}_1, \boldsymbol{\pi}_2 \in [0, 1]^2$

[1] In fact, when we only get 1 coin flip per example, so that each x_n is just a single head or tail, and this is a mixture-of-Bernoulli model, we won't be able to usefully identify the parameters. Consider the case of trying to tell between two coins with $\pi_1 = 0.3$ and π_2 =0.7 and $\theta_1 = \theta_2 = 0.5$ and two coins with $\pi_1 = \pi_2 = 0.5$ and $\theta_1 = \theta_2 = 0.5$. These two parameterizations put the same likelihood on any data set. Still, the work we do in this context extends to the case where x_n represents multiple coin tosses per example and we have a mixture-of-Binomial model. There we can usefully estimate the parameters of a mixture model. We get to this in Exercise 4.

to denote the biases for each coin, where $\boldsymbol{\pi}_1$ is the vector of probabilities for C_1, etc. Our model is a mixture of binomials where we have two binomials (coins 1 and 2), each with 2 outcomes (heads or tails). We let $\mathbf{w} := \{\boldsymbol{\theta}, \boldsymbol{\pi}\}$.

Now that we have the problem set up, let's use expectation maximization to learn parameters $\mathbf{w} := \{\boldsymbol{\theta}, \boldsymbol{\pi}\}$!
First we note that we can calculate $\mathbf{q}_n$ from $\mathbf{w}^{(t)}$ by writing:

$$\begin{aligned}
\mathbf{q}_n &= \begin{bmatrix} p(\mathbf{z}_n = C_1 | \mathbf{x}_n; \mathbf{w}^{(t)}) \\ p(\mathbf{z}_n = C_2 | \mathbf{x}_n; \mathbf{w}^{(t)}) \end{bmatrix} && (8) \\
&\propto \begin{bmatrix} p(\mathbf{x}_n | \mathbf{z}_n = C_1; \mathbf{w}^{(t)}) p(\mathbf{z}_n = C_1; \mathbf{w}^{(t)}) \\ p(\mathbf{x}_n | \mathbf{z}_n = C_2; \mathbf{w}^{(t)}) p(\mathbf{z}_n = C_2; \mathbf{w}^{(t)}) \end{bmatrix} && (9) \\
&\propto \begin{bmatrix} (\pi_{11})^{x_{n,1}} (\pi_{12})^{x_{n,2}} \theta_1 \\ (\pi_{21})^{x_{n,1}} (\pi_{22})^{x_{n,2}} \theta_2 \end{bmatrix} && (10)
\end{aligned}$$

We also have the complete data log-likelihood:

$$\begin{aligned}
\log p(\mathbf{x}_n, \mathbf{z}_n; \mathbf{w}) &= \log p(\mathbf{x}_n | \mathbf{z}_n; \mathbf{w}) p(\mathbf{z}_n; \mathbf{w}) && (11) \\
&= \log \prod_{k=1}^{2} \left(\theta_k \prod_{j=1}^{2} \pi_{kj}^{x_{n,j}} \right)^{z_{n,k}} && (12) \\
&= z_{n,1} \left(\log \theta_1 + x_{n,1} \log \pi_{11} + x_{n,2} \log \pi_{12} \right) && (13) \\
&\qquad + z_{n,2} \left(\log \theta_2 + x_{n,1} \log \pi_{21} + x_{n,2} \log \pi_{22} \right) \\
\log p(\mathbf{X}, \mathbf{Z}; \mathbf{w}) &= \sum_{n=1}^{N} \log p(\mathbf{x}_n, \mathbf{z}_n; \mathbf{w}) && (14)
\end{aligned}$$

Now expand the expected complete data log-likelihood:

$$\begin{aligned}
\mathcal{L}_c &=_{\mathbf{Z}|\mathbf{X};\mathbf{w}} \left[\sum_{n=1}^{N} \log p(\mathbf{x}_n, \mathbf{z}_n; \mathbf{w}) \right] && (15) \\
&=_{\mathbf{Z}|\mathbf{X};\mathbf{w}} \left[\sum_{n=1}^{N} \log p(\mathbf{z}_n; \mathbf{w}) + \log p(\mathbf{x}_n | \mathbf{z}_n; \mathbf{w}) \right] && (16) \\
&= \sum_{n=1}^{N} {}_{\mathbf{Z}|\mathbf{X};\mathbf{w}} \left[\log p(\mathbf{z}_n; \mathbf{w}) + \log p(\mathbf{x}_n | \mathbf{z}_n; \mathbf{w}) \right] && (17) \\
&= \sum_{n=1}^{N} \sum_{k=1}^{K} q_{n,k} \left(\log \theta_k + \sum_{j=1}^{2} x_{n,j} \log \pi_{kj} \right) && (18) \\
&= \sum_{n=1}^{N} q_{n,1} \left(\log \theta_1 + x_{n,1} \log \pi_{11} + x_{n,2} \log \pi_{12} \right) + q_{n,2} \left(\log \theta_2 + x_{n,1} \log \pi_{21} + x_{n,2} \log \pi_{22} \right) && (19)
\end{aligned}$$

Now we can use these derivations to do expectation maximization!:

1. Initialize $\mathbf{w}^{(0)}$ randomly.

2. Use $\mathbf{w}^{(t)}$ to calculate the vector of probabilities $\mathbf{q}_n$ for the distribution of each $\mathbf{z}_n$ (eq. 13).

3. Calculate the current expected likelihood using $\mathbf{q}_n$ and $\mathbf{w}^{(t)}$ (eq. 22).

   This step is not strictly necessary for calculating updates, but can be helpful for a variety of purposes, including debugging and testing convergence. Note that we need both $\mathbf{q}$ and $\mathbf{w}^{(t)}$ to get a value here.

4. Use $\mathbf{q}$ to calculate an updated set of parameters $\mathbf{w}^{(t+1)}$ by maximizing the expected likelihood as a function of $\mathbf{w}$ (eq. 22). Note that here we do not use $\mathbf{w}^{(t)}$.

   During optimization we need to enforce that $\sum_k \theta_k = 1$ and that $\sum_j \pi_{kj} = 1$, so that the distributions parameterized by $\boldsymbol{\theta}$ and $\boldsymbol{\pi}$ are valid.

   In general, we can enforce this constraint using Lagrange multipliers. Here, in the 2-dimensional case, we don't need to use Lagrangian methods and can instead substitute $\theta_2 = 1 - \theta_1$ and $\pi_{k2} = 1 - \pi_{k1}$:

$$\begin{aligned}\mathcal{L}_c = \sum_{n=1}^{N} \; & q_{n,1} \left(\log \theta_1 + x_{n,1} \log \pi_{11} + x_{n,2} \log(1 - \pi_{11})\right) \\ & + q_{n,2} \left(\log(1 - \theta_1) + x_{n,1} \log \pi_{21} + x_{n,2} \log(1 - \pi_{21})\right)\end{aligned} \tag{20}$$

   And then optimize w.r.t. $\theta_1, \pi_{11}, \pi_{21}$:

$$\frac{\partial \mathcal{L}_c}{\partial \theta_1} = \sum_{n=1}^{N} \left(\frac{q_{n,1}}{\theta_1} - \frac{q_{n,2}}{1 - \theta_1} \right) = 0 \tag{21}$$

$$\frac{\partial \mathcal{L}_c}{\partial \pi_{11}} = \sum_{n=1}^{N} q_{n,1} \left(\frac{x_{n,1}}{\pi_{11}} - \frac{x_{n,2}}{1 - \pi_{11}} \right) = 0 \tag{22}$$

$$\frac{\partial \mathcal{L}_c}{\partial \pi_{21}} = \sum_{n=1}^{N} q_{n,2} \left(\frac{x_{n,1}}{\pi_{21}} - \frac{x_{n,2}}{1 - \pi_{21}} \right) = 0 \tag{23}$$

   From here we can solve for the optimal value of $\mathbf{w}$ (i.e. $\theta_1, \pi_{11}, \pi_{22}$), and set $\mathbf{w}^{(t+1)} = \arg\max_{\mathbf{w}\ \mathbf{Z}|\mathbf{X};\mathbf{w}} \mathcal{L}_c$.

Note: Above we show the derivation of all steps of the algorithm, but once you know the closed form expression for $\mathbf{w}^{(t+1)}$, the steps of the algorithm are really just initialization, calculate the distribution $\mathbf{q}_n$ from $\mathbf{w}^{(t)}$, and then calculate $\mathbf{w}^{(t+1)}$ from $\mathbf{q}$. All the difficult work is in deriving the update equations.

In more complicated models, the optimal $\mathbf{w}^{(t+1)}$ may not have a closed form solution; in these cases, instead we can do gradient descent to calculate the optimal value.

Exercise 2. *Derive the closed form updates for* $\boldsymbol{\theta}^{(t)}, \boldsymbol{\pi}^{(t)}$ *from the steps above.*

Solution. For $\theta_k^{(t)}$, we find θ_1 such that

$$\frac{\partial \mathcal{L}_c}{\partial \theta_1} = \sum_{n=1}^{N} \left(\frac{q_{n,1}}{\theta_1} - \frac{q_{n,2}}{1-\theta_1} \right) = 0$$

and can do the same for $\frac{\partial \mathcal{L}_c}{\partial \theta_2}$.

Consider θ_1:

$$\frac{1}{\theta_1} \sum_{n=1}^{N} q_{n,1} - \frac{1}{1-\theta_1} \sum_{n=1}^{N} q_{n,2} = 0$$

After some algebra, we'll ultimately obtain

$$\theta_1 = \frac{\sum_n q_{n,1}}{\sum_n (q_{n,1} + q_{n,2})}$$

Note that for a single observation, $q_{n,1} + q_{n,2} = 1$, so $\sum_n (q_{n,1} + q_{n,2}) = N$. Thus,

$$\theta_k^{(t)} \leftarrow \frac{\sum_{n=1}^{N} q_{n,k}}{N}$$

Now let's look at $\boldsymbol{\pi}_k^{(t)}$:

$$\frac{1}{\pi_{11}} \sum_{n=1}^{N} q_{n,1} x_{n,1} - \frac{1}{1-\pi_{11}} \sum_{n=1}^{N} q_{n,1} x_{n,2} = 0$$

After some algebra, we obtain

$$\pi_{11} = \frac{\sum_{n=1}^{N} q_{n,1} x_{n,1}}{\sum_{n=1}^{N} q_{n,1}(x_{n,1} + x_{n,2})}$$

and similarly

$$\pi_{12} = \frac{\sum_{n=1}^{N} q_{n,1} x_{n,2}}{\sum_{n=1}^{N} q_{n,1}(x_{n,1} + x_{n,2})}$$

therefore,

$$\boldsymbol{\pi}_1 = \frac{\sum_{n=1}^{N} q_{n,1} \mathbf{x_n}}{\sum_{n=1}^{N} q_{n,1}(x_{n,1} + x_{n,2})}$$

We can generalize and find the following result for the update $\boldsymbol{\pi}_k^{(t)}$.

$$\boldsymbol{\pi}_k^{(t)} \leftarrow \frac{\sum_{n=1}^{N} q_{n,k} \mathbf{x}_n}{\sum_{n=1}^{N} \sum_{j=1}^{2} q_{n,k} x_{n,j}}$$

□

Once we have an estimate for the MLE $\mathbf{w}$, we can use it to do prediction of hidden states for a new incoming coin flip, using step 2 from above. So, given a new coin flip, we can predict whether it came from the first or the second coin.

Our model may not be very good, since in particular it is impossible to tell the difference between having one coin chosen with high probability with $\pi_1 = 0.5$ (and another picked almost never with $\pi_2 = 0.1$) and two equally likely coins with biases 0.4 and 0.6. In this case, as discussed above, with only one observation for each coin we cannot successfully estimate the parameters of the mixture model. This problem is due to the data setup: we need a mixture of binomials, with multiple observations per coin.

Exercise 3. (Optional) *Consider the following data generation process: the setup is the same as above, but instead of flipping the chosen coin once, we flip it 10 times before choosing a new coin.*

1. *Find an appropriate choice of latent variables* $\mathbf{z}_n$ *and calculate the distribution of* $\mathbf{z}_n$ *given the data* $\mathbf{x}_{n,j}$ *(where n iterates over each set of 10 coin flips, and* $j \in [1, 10]$*) and an estimate for* $\boldsymbol{\theta}$*.*

2. *Find the expression for the expected complete data log-likelihood*

3. *Find the closed form update equations for* $\boldsymbol{\theta}^{(t)}$*, and compare them to the result from Exercise 3.*

Solution.

1. Here again we can use $\mathbf{z}_n$ to denote the 1-hot choice of the chosen coin, this time used for all 10 of the flips. The distribution of $\mathbf{z}_n$ is again proportional to the prior times the likelihood, which is:

$$p(\mathbf{z}_n|\mathbf{x}_n, \mathbf{w}) \propto p(\mathbf{x}_n|\mathbf{z}_n, \mathbf{w}) \cdot p(\mathbf{z}_n|\mathbf{w}) = \prod_{k=1}^{K=2} \left(\prod_{j=1}^{10} \prod_{l=1}^{2} \pi_{kl}^{x_{njl}} \right)^{z_{n,k}} \cdot \prod_{k=1}^{K=2} \theta_k^{z_{n,k}}$$

   where $x_{n,jl}$ is the indicator variable for the j-th coin in the n-th set of 10 being in class l (in this case, heads or tails).

2. The expected complete data log likelihood is similar to the previous exercise:

$$\mathcal{L}_c = \sum_{n=1}^{N} \sum_{k=1}^{K=2} q_{n,k} \left(\log \theta_k + \sum_{j=1}^{10} \sum_{l=1}^{2} x_{n,jl} \log \pi_{kj} \right)$$

3. In fact, $\boldsymbol{\theta}$ satisfies the same expression as before, so the update is the same:

$$\theta_k^{(t)} \leftarrow \frac{\sum_{n=1}^{N} q_{n,k}}{N}$$

   The parameters of $\boldsymbol{\pi}$ can also be estimated by following an analogous approach, and setting partials to zero.

□

3 Principal Component Analysis

3.1 Motivation

In many supervised learning problems, we try to find rich features that increase the expressivity of our model. In practice, this often involves using basis functions to transform the model input into a higher dimensional space (eg. given data x, using x and x^2 as features, or using features learned by a neural network). However, sometimes we want to *reduce* the dimensionality of our data.

Exercise 4. *Why would we want to reduce the dimensionality of our data? Can you think of example cases?*

Solution. There can be several reasons:

- Fewer features are easier to interpret: we might want to know why our model outputs a certain diagnosis, and only some of the patient record details will be relevant.
- Models with fewer features are easier to handle computationally.
- Our data might be arbitrarily high-dimensional because of noise, so we would like to access the lower-dimensional signal from the data.

□

When working with high-dimensional data, it is likely that not every feature will give us completely new information, for example, multiple features may be correlated. The idea that we might be able to reduce the dimensionality of our data by eliminating redundant information is a powerful one, and is the essence of *Principal Component Analysis (PCA).*

Recall that our data $\mathbf{X}$ is an N x D dimensional matrix were each row corresponds to a datapoint and each column corresponds to a feature. The goal of PCA is to re-express $\mathbf{X}$ in terms of a new basis such that every column of this transformed matrix now gives us completely new information (ie. the features are *linearly independent*). The columns of the corresponding change-of-basis matrix are called *principal components* and are constructed in such a way that each principal component accounts for more of the variance in our original data than the next. After re-expressing $\mathbf{X}$ in terms of the principal components, we can make a decision about how many columns of the new matrix we want to keep depending on how much of the original variation we want to preserve and what our usecase is. For example, if we wanted to plot the resulting data we might only keep the first two principal components.

3.2 Finding the lower dimensional representation

Let's say we want to compress our D-dimensional data into K dimensions ($K < D$) ie. we want to find K D-dimensional basis vectors $\mathbf{u}_1, ..., \mathbf{u}_K$ such that we can represent $\mathbf{x}_n$ as a linear combination of them:

$$\mathbf{x}_n \approx z_{n,1}\mathbf{u}_1 + ... + z_{n,k}\mathbf{u}_k = \mathbf{U}\mathbf{z}_n$$

where $z_{n,1}, ..., z_{n,k}$ are scalars and $\mathbf{U}$ is a matrix of all the basis vectors. We define the *reconstruction loss* to be the distance from the approximation of each $\mathbf{x}_n$ using the new basis to $\mathbf{x}_n$ itself:

$$\mathcal{L}(\{\mathbf{z}_n\}, \mathbf{U}) = \frac{1}{N}\sum_{n=1}^{N} ||\mathbf{x}_n - \mathbf{U}\mathbf{z}_n||_2^2$$

The solution to the above loss is not unique! To find a unique solution, let's impose the constraint that $\mathbf{U}$ must be *orthonormal*, meaning that for any distinct rows k, k' in $\mathbf{U}$, $\mathbf{u}_k \cdot \mathbf{u}_k = 1$ and $\mathbf{u}_k \cdot \mathbf{u}_{k'} = 0$. Using orthonormal basis vectors yields the nice property that $\mathbf{u}_k^T \mathbf{x}_n = z_{n,k}$.

Let's subtract the mean of our data $\bar{\mathbf{x}}$ from each $\mathbf{x}_n$ so that our bases capture variation from the mean. Our loss function is now

$$\mathcal{L}(\{\mathbf{z}_n\}, \mathbf{U}) = \frac{1}{N}\sum_{n=1}^{N} ||(\mathbf{x}_n - \bar{\mathbf{x}}) - \mathbf{U}\mathbf{z}_n||_2^2 \text{ s.t. } \mathbf{U} \text{ is orthonormal}$$

Note that if $K = D$, then we could perfectly reconstruct $\mathbf{x}_n$ since we'd be able to preserve all the features. Thus, $\mathbf{x}_n - \bar{\mathbf{x}} = \sum_{k=1}^{D} z_{n,k}\mathbf{u}_k$. Thus, we can simplify our loss as follows:

$$\begin{aligned}
\mathcal{L}(\{\mathbf{z}_n\}, \mathbf{U}) &= \frac{1}{N}\sum_{n=1}^{N} ||\sum_{k=1}^{D} z_{n,k}\mathbf{u}_k - \sum_{k=1}^{K} z_{n,k}\mathbf{u}_k||_2^2 \\
&= \frac{1}{N}\sum_{n=1}^{N} ||\sum_{k=K+1}^{D} z_{n,k}\mathbf{u}_k||_2^2 \\
&= \frac{1}{N}\sum_{k=K+1}^{D}\sum_{n=1}^{N} \mathbf{u}_k^T(\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^T\mathbf{u}_k \\
&= \sum_{k=K+1}^{D} \mathbf{u}_k^T \left[\frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^T\right]\mathbf{u}_k
\end{aligned}$$

The expression in the square brackets is the *empirical covariance matrix of* $\mathbf{X}$! Finding the directions of greatest variation correspond to finding the *eigenvectors* of this empirical covariance matrix. Eigenvectors with a higher eigenvalue capture more variance in the data than those with a lower eigenvalue.

To summarize, to perform PCA:

1. Center the data by subtracting the mean of each feature from each data point. Steps 2 - 5 will be performed on the centered data $\mathbf{X}$: $(N \times D)$.

2. Calculate the empirical covariance matrix:

$$\mathbf{S} = \frac{1}{N}(\sum_{n=1}^{N} \mathbf{x}_n \mathbf{x}_n^\top) = \frac{1}{N}\mathbf{X}^\top \mathbf{X}$$

3. Decide how many dimensions K out of the original D we want to keep in the final representation.

4. Find the K largest eigenvalues of $\mathbf{S}$. The $K \times 1$ eigenvectors $(\mathbf{u}_1, \ldots, \mathbf{u}_K)$ corresponding to these eigenvalues will be our lower-dimensional basis.

5. Reduce the dimensionality of a data point $\mathbf{x}$ by projecting it onto this basis yielding a new reconstructed vector $\mathbf{z}$:

$$\mathbf{z} = \mathbf{U}^T \mathbf{x}$$

Exercise 5. *You are given the following data set:*

$$\mathbf{x}_1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \quad \mathbf{x}_2 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \quad \mathbf{x}_3 = \begin{bmatrix} -2 \\ -1 \end{bmatrix}$$

You would like to use PCA to find a 1-dimensional representation of the data.

1. *Plot the data set.*

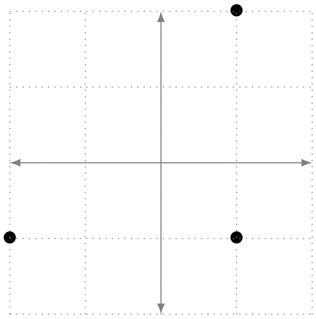
2. *Compute the empirical covariance matrix* $\mathbf{S}$.

3. *You find that* $\mathbf{S}$ *has eigenvector* $[-1\ 1]^\top$ *with eigenvalue* 1 *and eigenvector* $[1\ 1]^\top$ *with eigenvalue* 3. *What is the (normalized) basis vector* $\mathbf{u}_1$ *of your 1-dimensional representation? Add the basis vector* $\mathbf{u}_1$ *to your plot.*

4. *Compute the coefficients* z_1, z_2, z_3. *Add the lower-dimensional representations* $z_1\mathbf{u}_1, z_2\mathbf{u}_1, z_3\mathbf{u}_1$ *to your plot. Based on your plot, what is the relationship between* $z_i\mathbf{u}_1$ *and* $\mathbf{x}_i$ *with respect to the new basis?*

5. *Based on your plot, what would happen if you chose the unused eigenvector to be your basis vector?*

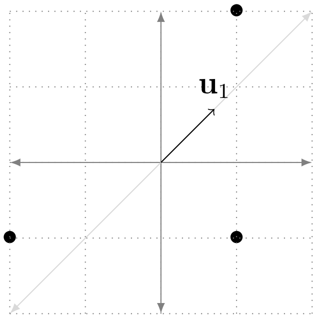
Solution. 1.



2.

$$\mathbf{S} = \frac{1}{3}\mathbf{X}^\top\mathbf{X} = \frac{1}{3}\begin{bmatrix} 1 & -1 \\ 1 & 2 \\ -2 & -1 \end{bmatrix}^\top \begin{bmatrix} 1 & -1 \\ 1 & 2 \\ -2 & -1 \end{bmatrix} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

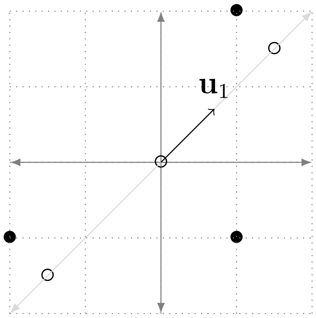
3. We select the eigenvectors with the largest eigenvalues for our basis, so our basis will contain a scalar multiple of $[1\ 1]^\top$. Normalizing $[1\ 1]^\top$ gives us that $\mathbf{u}_1 = [\frac{\sqrt{2}}{2}\ \frac{\sqrt{2}}{2}]^\top$.



4.

$$z_1 = \mathbf{x}_1^\top \mathbf{u}_1 = 0, \quad z_2 = \mathbf{x}_2^\top \mathbf{u}_1 = \frac{3\sqrt{2}}{2}, \quad z_3 = \mathbf{x}_3^\top \mathbf{u}_1 = -\frac{3\sqrt{2}}{2}$$

The open circles in the plot represent the lower-dimensional representation:



$z_i\mathbf{u}_1$ is the projection of $\mathbf{x}_i$ onto the basis vector.

5. If we chose $[-1\ 1]^\top$ to be the basis of our new representation, then the representation would capture less of the variance in the data. For example, $\mathbf{x}_2$ and $\mathbf{x}_3$ would be represented by the same point.

$\square$

Exercise 6. *Suppose that our data are centered (i.e., have sample mean 0). Recall that in lecture, we showed that, when optimizing over (semi)-orthogonal matrices* $\mathbf{U} \in \mathbb{R}^{m\times d}$ *(i.e., where* $\mathbf{U}^T\mathbf{U} = I$*) to minimize the reconstruction loss,*

$$\mathcal{L}(\mathbf{U}) = \frac{1}{n}\sum_{i=1}^{n} ||\mathbf{x_i} - \mathbf{U}\mathbf{U}^T\mathbf{x_i}||_2^2,$$

we found that $\mathbf{U}_d$*, the matrix whose first* d *columns are (in order) the top* d *eigenvectors of the empirical covariance matrix* $\mathbf{\Sigma} = \frac{1}{n}\mathbf{X}^T\mathbf{X}$*, will achieve the minimum (i.e.,* $\mathbf{z} = \mathbf{U}^T\mathbf{x}$ *is the projection of* $\mathbf{x}$ *into* d *dimensions, and* $\mathbf{U}\mathbf{z}$ *is its reconstruction in* $\mathbb{R}^m$*). In class, we showed this for case when* $d = m - 1$ *by using Lagrange multipliers. Show this, in general, for* d*.*

Hint: you may use the following theorem: [Courant-Fischer] Let $\mathbf{A}$ *be a symmetric* $n \times n$ *matrix with eigenvalues* $\lambda_1 \leq \cdots \leq \lambda_n$ *and corresponding eigenvectors* $\mathbf{v_1}, \ldots, \mathbf{v_n}$*. Then*

$$\lambda_1 = \min_{||x||=1} \mathbf{x}^T\mathbf{A}\mathbf{x}$$

$$\lambda_2 = \min_{||\mathbf{x}||=1, \mathbf{x}\perp\mathbf{v}_1} \mathbf{x}^T\mathbf{A}\mathbf{x}$$

$$\vdots$$

$$\lambda_i = \min_{||\mathbf{x}||=1, \mathbf{x}\perp\mathbf{v}_1, \mathbf{x}\perp\mathbf{v}_2, \ldots, \mathbf{x}\perp\mathbf{v}_{i-1}} \mathbf{x}^T\mathbf{A}\mathbf{x}$$

$$\vdots$$

$$\lambda_n = \min_{||\mathbf{x}||=1, \mathbf{x}\perp\mathbf{v}_1, \mathbf{x}\perp\mathbf{v}_2, \ldots, \mathbf{x}\perp\mathbf{v}_{n-1}} \mathbf{x}^T\mathbf{A}\mathbf{x}.$$

Solution. We proceed just as in class. Let $\tilde{\mathbf{U}} \in \mathbb{R}^{m\times m}$ be an orthogonal matrix, and we will let $\mathbf{U}$ be the matrix comprising $\tilde{\mathbf{U}}$'s first d columns. Letting $\tilde{\mathbf{U}}^{(k)}$ denote the kth column of $\tilde{\mathbf{U}}$, the orthogonality of $\tilde{\mathbf{U}}$ implies that the $\tilde{\mathbf{U}}^{(k)}$ form a basis in $\mathbb{R}^m$ so that, for each $\mathbf{x}_i$, we can find $\mathbf{z}_i \in \mathbb{R}^m$ such that

$$\mathbf{x}_i = \sum_{k=1}^{m} \mathbf{z}_{i,k}\tilde{\mathbf{U}}^{(k)}.$$

The orthogonality of $\tilde{\mathbf{U}}$ implies that $\tilde{\mathbf{U}}^T\mathbf{x}_i = (\mathbf{z}_{i,1}, \ldots, \mathbf{z}_{i,m})$ and $\mathbf{U}^T\mathbf{x}_i = (\mathbf{z}_{i,1}, \ldots, \mathbf{z}_{i,d})$. Hence, we can write the reconstruction loss as

$$\mathcal{L}(\mathbf{U}) = \frac{1}{n}\sum_{i=1}^{n} ||\sum_{k=1}^{m} \mathbf{z}_{i,k}\tilde{\mathbf{U}}^{(k)} - \sum_{k=1}^{d} \tilde{\mathbf{U}}^{(k)}\mathbf{z}_{i,k}||_2^2 = \frac{1}{n}\sum_{i=1}^{n} ||\sum_{k=1}^{m} \mathbf{z}_{i,k}\tilde{\mathbf{U}}^{(k)} - \sum_{k=1}^{d} \mathbf{z}_{i,k}\tilde{\mathbf{U}}^{(k)}||_2^2$$

$$= \frac{1}{n}\sum_{i=1}^{n} || \sum_{k=d+1}^{m} \mathbf{z}_{i,k}\tilde{\mathbf{U}}^{(k)}||_2^2 = \frac{1}{n}\sum_{i=1}^{n}\sum_{k=d+1}^{m} \mathbf{z}_{i,k}^2 = \frac{1}{n}\sum_{i=1}^{n}\sum_{k=d+1}^{m} \left(\tilde{\mathbf{U}}^{(k)T}\mathbf{x}_i\right)\left(\mathbf{x}_i^T\tilde{\mathbf{U}}^{(k)}\right)$$

$$= \sum_{k=d+1}^{m} \tilde{\mathbf{U}}^{(k)T}\left(\frac{1}{n}\mathbf{X}^T\mathbf{X}\right)\tilde{\mathbf{U}}^{(k)} = \sum_{k=d+1}^{m} \tilde{\mathbf{U}}^{(k)T}\mathbf{\Sigma}\tilde{\mathbf{U}}^{(k)},$$

where $\mathbf{\Sigma}$ denotes the empirical covariance matrix.

Let $\mathbf{u}_1, \ldots, \mathbf{u}_m$ denote the (unit length) eigenvectors of $\mathbf{\Sigma}$ (ordered by increasing corresponding eigenvalue). When $d + 1 = m$, there's only one element in the sum, and since we know that $\tilde{\mathbf{U}}$ is orthogonal, we must have that $||\tilde{\mathbf{U}}^{(k)}||_2 = 1$ and thus Courant-Fischer says that $\tilde{\mathbf{U}}^{(k)} = \mathbf{u}_1$ will minimize. When $d + 1 = m - 1$, there will be two elements in the sum:

$$\tilde{\mathbf{U}}^{(m-1)T}\mathbf{\Sigma}\tilde{\mathbf{U}}^{(m-1)} + \tilde{\mathbf{U}}^{(m)T}\Sigma\tilde{\mathbf{U}}^{(m)}.$$

Consider solving this optimization problem by first picking $\tilde{\mathbf{U}}^{(m)}$ and then $\tilde{\mathbf{U}}^{(m-1)}$. Now, we either choose $\tilde{\mathbf{U}}^{(m)}$ to be $\mathbf{u}_1$ or we do not. If we choose $\tilde{\mathbf{U}}^{(m)} = \mathbf{u}_1$, then Courant-Fischer implies that picking $\tilde{\mathbf{U}}^{(m-1)} = \mathbf{u}_2$ will minimize. If we do not choose $\tilde{\mathbf{U}}^{(m)}$ to be $\mathbf{u}_1$, then Courant-Fischer implies that picking $\tilde{\mathbf{U}}^{(m-1)} = \mathbf{u}_1$ will minimize and that, if we did not pick $\tilde{\mathbf{U}}^{(m)}$ to be $\mathbf{u}_2$, we could've achieved an even lower value more by picking it to be so. Thus, in either case, we will take $\tilde{\mathbf{U}}^{(m)}$ and $\tilde{\mathbf{U}}^{(m-1)}$ to be $\mathbf{u}_1$ and $\mathbf{u}_2$.

Proceeding iteratively like this shows that we will chose $\tilde{\mathbf{U}}^{(d+1)}, \ldots, \tilde{\mathbf{U}}^{(m)}$ to be $\mathbf{u}_1, \ldots, \mathbf{u}_{m-d}$. Now, note that these are the columns we *exclude* from the matrix $\tilde{\mathbf{U}}$ when we construct $\mathbf{U}$ (which will be the first d column of $\tilde{\mathbf{U}}$). Since $\mathbf{\Sigma}$ is symmetric, the eigenvectors $\mathbf{u}_1, \ldots, \mathbf{u}_n$ are orthogonal (or can be chosen to be in the case of repeated eigenvalues), taking $\mathbf{U}$'s columns to be $\mathbf{u}_{m-d+1}, \ldots, \mathbf{u}_m$ (i.e., the top d eigenvectors) indeed will minimize the reconstruction loss, as desired. □