CS 337, Fall 2023
Expectation Maximization Algorithm and PCA

Scribes*: Kevin Prafull Baua, Polu Lakshmi Sai Lokesh Reddy, Ayush Ramteke, Pranay Midathana, Nithin Islavath, Thota Honey Sahith
Edited by: Sanjeev Kumar

October 25, 2023

Recap

Recall that in the fully observable setting (when both x_i and z_i are observed variables), the MLE objective is

$$\sum_i \log P(x_i, z_i) = \sum_i \sum_k \mathbb{1}[z_i = k](\log \pi_k + \log \mathcal{N}(x_i; \mu_k, I))$$

where $\mathbb{1}$ is the indicator function
In the real setting however, only $x_1, x_2, ..., x_N$ are observed and $z_i's$ are hidden/latent. Since we don't know $z_i's$ we can estimate them based on our current model.
We define the following variables:

$$\gamma_{k,i} \triangleq P(z_i = k|x_i) = \frac{\pi_k \mathcal{N}(x_i; \mu_k, I)}{\sum_j \pi_j \mathcal{N}(x_i; \mu_j, I)}$$

If we have a model with parameters μ_k and π_k, then the variable γ_k, i describes how much the k^{th} cluster is responsible for the i^{th} point under this model.
If we treat the $\gamma_{k,i}$ values as fixed, then the MLE objective[a] becomes

$$LL_g(\mu_k, \pi_k) = \sum_i \sum_k \gamma_{k,i}(\log \pi_k + \log \mathcal{N}(x_i; \mu_k, I))$$

Optimizing with respect to μ_k and π_k we get,

$$\nabla_{\mu_k} LL_g(\mu_k, \pi_k) = 0 \quad \text{and} \quad \nabla_{\pi_k} LL_g(\mu_k, \pi_k) = 0$$

$$\implies \mu_k = \frac{\sum_i \gamma_{k,i} x_i}{\sum_i \gamma_{k,i}} \quad \text{and} \quad \pi_k = \frac{\sum_i \gamma_{k,i}}{N} \tag{1}$$

*Equal contributions by all scribes

Now, using these updated parameters we can get updated values for $\gamma_{k,i}$. This motivates the Expectation Maximization (EM) Algorithm.

[a]The above log-likelihood function is not the true log-likelihood function since we are keeping $\gamma_{k,i}$'s fixed. EM iteratively calculates the lower bound of the log-likelihood function and optimizes it.

1 Expectation Maximization Algorithm for GMMs

The EM algorithm is a general algorithm for optimizing models in the presence of latent variables. The EM algorithm has an iterative approach that performs two steps alternately until convergence similar to the K-Means algorithm. The first step attempts to estimate the latent variables called the E-step. The second step attempts to optimize the parameters of the model to best explain the data called the M-step.

1.1 Algorithm

1. **Initialization:** Initialize the model parameters

$$\mu_k^{curr} = \mu_k^{init}, \quad \pi_k^{curr} = \pi_k^{init} \qquad \forall k \in \{1, 2, ..., k\}$$

2. **E-Step:** Compute the posterior probabilities determining how much each datapoint commits to a cluster given current parameters

$$\gamma_{k,i} = \frac{\pi_k^{curr} \mathcal{N}(x_i; \mu_k^{curr}, I)}{\sum_j \pi_j^{curr} \mathcal{N}(x_i; \mu_j^{curr}, I)} \qquad \forall k \in \{1, ..., k\}, \forall i \in \{1, ..., N\}$$

3. **M-Step:** Re-estimate u_k, π_k model parameters given $\gamma_{k,i}$'s using equation 1

$$\mu_k^{new} = \frac{\sum_i \gamma_{k,i} x_i}{\sum_i \gamma_{k,i}} \quad \text{and} \quad \pi_k^{new} = \frac{\sum_i \gamma_{k,i}}{N} \qquad \forall k \in \{1, 2, ..., k\}$$

4. **Check convergence:** Terminate if log-likelihood value is within a threshold ϵ of the previous log-likelihood else loop back to E-step

   if $((\sum_i \log \sum_k (\pi_k^{new} \mathcal{N}(x_i; \mu_k^{new}, I)) - \sum_i \log \sum_k (\pi_k^{curr} \mathcal{N}(x_i; \mu_k^{curr}, I)) < \epsilon)$ **then**

   <**TERMINATE**>

   else

   $\mu_k^{curr} = \mu_k^{new}, \quad \pi_k^{curr} = \pi_k^{new} \qquad \forall k \in \{1, 2, ..., k\}$

   <**Goto 2. E-Step**>

   end if

1. Note that instead of maximizing the actual likelihood function (which is difficult due to summation inside the log), here we are maximizing a lower bounding auxiliary function which is the expected value of a log-likelihood function under the current model by fixing the values of $\gamma_{k,i}$'s

2. Here we have kept the covariance matrix as Identity. To generalize it we can use the same algorithm introducing the covariance matrix in the model parameters.

3. This method is sensitive to initialization and may give different results on different initial values, hence it is common to run this algorithm multiple times with different initial values to get the best results.

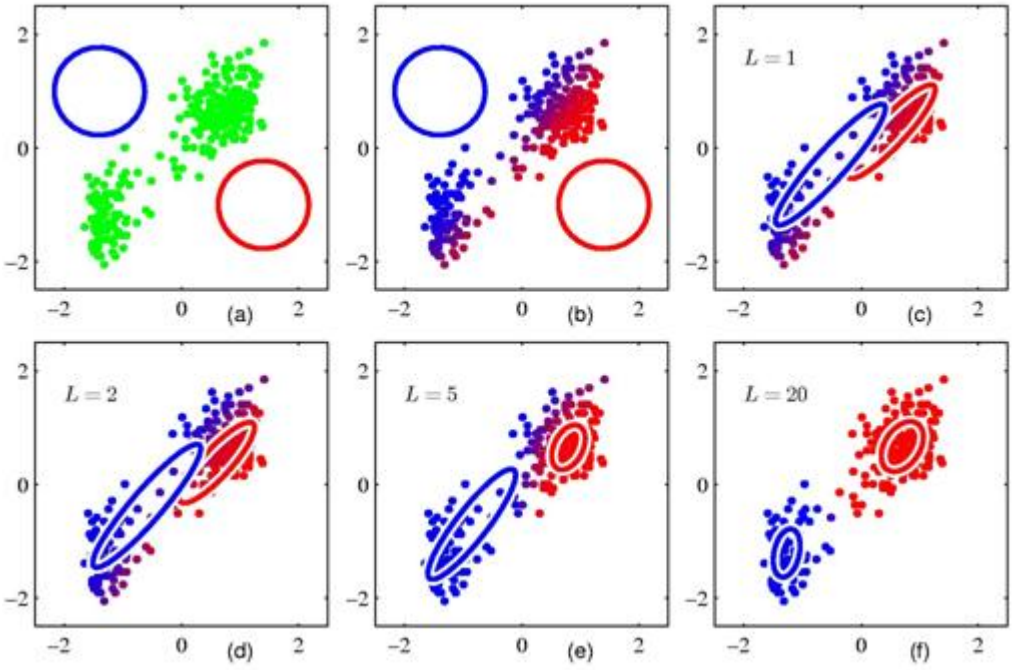

Figure 1: Iterations of EM algorithm

2 Dimensionality Reduction

Dimensionality reduction is a technique used to reduce the number of features or variables in a dataset while preserving the most important information. High-dimensional data can suffer from the "**Curse of Dimensionality**". This means that as the number of features increases, the amount of data needed to describe the space between data points grows exponentially. This leads to problems like increased computational complexity, overfitting, and difficulty in visualizing or interpreting the data.

The preeminent method for addressing this in Machine Learning is Principal Component Analysis (PCA).

2.1 Principal Component Analysis (PCA)

Consider a point in d-dimensional space $\mathbf{x} \in \mathbb{R}^d$. The aim of PCA is to find a low-dimensional projection of $\mathbf{x}$,

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

where, $\mathbf{U} \in \mathbb{R}^{d\times k}$, $\mathbf{U}^T\mathbf{U} = \mathbf{I}$ ($\mathbf{U}$ is an orthonormal matrix and $k < d$), and $\mathbf{z}$ is projection of $\mathbf{x}$ in the lower dimension

Here, we are projecting data from d-dimensional space to k-dimensional space. **How do we estimate the matrix, U?**

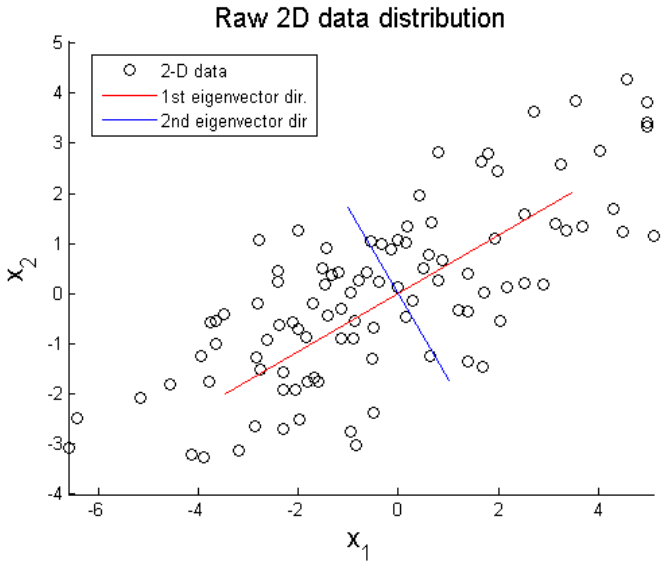

Figure 2: PCA on a 2-Dimensional data to reduce it to 1-Dimensional data

Let's say we want to project the above 2-D data into 1-D space. If we project along x_1 or x_2 much of the information will be lost. It makes sense here to project along the red line as it captures the direction of maximum variance. The goal of PCA is to find these directions of maximal variations and store information along them to minimize information loss.

2.2 Objectives

- **Objective 1: Minimizing Reconstruction Error**

$$ENCODE: \mathbf{z} = \mathbf{U}^T\mathbf{x}$$
$$DECODE: \hat{\mathbf{x}} = \mathbf{U}\mathbf{z} = \mathbf{U}\mathbf{U}^T\mathbf{x}$$

If we project data from $\mathbf{x} \to \mathbf{z}$ and then reconstruct it back from $\mathbf{z} \to \hat{\mathbf{x}}$, we would like the difference between the original vector $\mathbf{x}$ and $\hat{\mathbf{x}}$ to be minimized.
Reconstruction objective is defined as the sum of squared differences between $\mathbf{x}$ and $\hat{\mathbf{x}}$ over all data points. We need to minimize this objective.

$$\min_{\mathbf{U}^T\mathbf{U}=\mathbf{I}} \sum_i \|\mathbf{x}_i - \mathbf{U}\mathbf{U}^T\mathbf{x}_i\|^2$$

- **Objective 2: Maximizing Variance Of Projected Data**
  Let's mean center the data $\implies E[\mathbf{x}] = \mathbf{0}$. Henceforth, we take $\mathbf{x}$ to be mean centred. We would like the variance of projected data to be maximized.

$$\max_{U^TU=I} E[\|\mathbf{U}^T\mathbf{x}\|^2] - (E[\mathbf{U}^T\mathbf{x}])^2 = \max_{U^TU=I} E[\|\mathbf{U}^T\mathbf{x}\|^2] - (\mathbf{U}^T E[\mathbf{x}])^2$$

$$\max_{U^TU=I} E[\|\mathbf{U}^T\mathbf{x}\|^2] \qquad \text{, since } E[\mathbf{x}] = \mathbf{0}$$

Equivalence of objectives for PCA

$$\begin{aligned}
x &= UU^Tx + (I - UU^T)x \\
\|x\|^2 &= \|UU^Tx + (I - UU^T)x\|^2 \\
&= \|UU^Tx\|^2 + \|(I - UU^T)x\|^2 + (UU^Tx)^T(I - UU^T)x + ((I - UU^T)x)^T(UU^Tx) \\
&= \|UU^Tx\|^2 + \|(I - UU^T)x\|^2 + (x^TUU^T)(I - UU^T)x + (x^T(I - UU^T))(UU^Tx) \\
&= \|UU^Tx\|^2 + \|(I - UU^T)x\|^2 \\
&\quad + x^T(UU^T - UU^TUU^T)x + x^T(UU^T - UU^TUU^T)x \\
&= \|UU^Tx\|^2 + \|(I - UU^T)x\|^2 \qquad \text{using } U^TU = I \\
&= (UU^Tx)^T(UU^Tx) + \|(I - UU^T)x\|^2 \\
&= (x^TUU^TUU^Tx) + \|(I - UU^T)x\|^2 \\
\|x\|^2 &= \|U^Tx\|^2 + \|(x - UU^Tx)\|^2 \qquad \text{using } U^TU = I
\end{aligned}$$

Now take expectation

$$E[\|x\|^2] = E[\|U^Tx\|^2] + E[\|(x - UU^Tx)\|^2]$$

Here, the first term $E[\|U^Tx\|^2]$ is Objective 2 and the second term $E[\|(x - UU^Tx)\|^2]$ is Objective 1. Since the original dataset is fixed $\implies E[\|x\|^2]$ is a constant. Hence, maximizing Objective 1 will automatically result in minimizing Objective 2. Hence, the optimization problem of both Objectives is equivalent.

2.3 Finding the Projection matrix U

Consider objective 2:
Define $\mathbf{X}$ to be the $d \times n$ matrix with n columns of the **mean centred** d-dimensional data points. Let us assume we're projecting down to $k = 1$. Hence, we need to find a $d \times 1$ unit vector $\mathbf{u}$ which maximizes the variance of projected data

$$\begin{aligned}
\max_{||u||=1} \frac{1}{n}\sum_i (u^Tx_i) &= \max_{||u||=1} \frac{1}{n}||(X^Tu)||^2 \\
&= \max_{||u||=1} u^T(\frac{1}{n}XX^T)u
\end{aligned}$$

here $(\frac{1}{n}XX^T)$ is the covariance matrix of data, let's denote it by $\mathbf{S} = \frac{1}{n}XX^T$

$$\max_{||u||=1} \frac{1}{n}\sum_i (u^Tx_i) = \max_{||u||=1} u^TSu$$

The u which maximizes the above equation turns out to be the eigenvector of S with the largest eigenvalue and the maximum value achieved is the largest eigenvalue of S. Proof is given below. The above constraint will have the same solution as that of:

$$\max_{||u|| \leq 1} u^T S u$$

Why?
Even if the new constraint is $||u|| \leq 1$, we can show that the solution u of the new constraint will always have $||u|| = 1$. This is because if it were less than one say $u = k\hat{u}$, then objective value becomes $k^2 \hat{u}^T S \hat{u}$, then scaling k to $k = 1$ while keeping the same $\hat{u}$ would strictly increase the objective value and hence k has to be 1 in order to maximize it. Therefore, any solution of the above new constraint problem will also be a solution of the original constraint problem and vice versa.

We can prove this using the method of Lagrange multipliers

$$L = u^\top S u + \lambda(1 - u^\top u)$$

Differentiating with respect to u and setting the derivative equal to zero gives:

$$2Su - 2\lambda u = 0$$

Rearranging, we see that u must satisfy:

$$Su = \lambda u$$

$\implies$ Picking u as any eigenvector of S and λ as the corresponding eigenvalue satisfies above problem. The objective becomes $u^T S u = \lambda u^T u = \lambda$ which is maximized at the largest eigenvalue.
Solution: Pick the eigenvector with the largest eigenvalue.

Similarly, if we want to project our data onto k-dimensional space we choose the k eigen vectors corresponding to the top k-eigen values of the covariance matrix. We are guaranteed to be able to do so since the covariance matrix is symmetric and positive semidefinite and hence has a full set of orthogonal eigenvectors.

2.4 How to find top k eigen vectors

1. Use Eigen decomposition to find top k eigenvectors
2. Use SVD to find the right singular vectors which are same as eigenvectors of covariance matrix

In fact, most implementations of PCA actually perform SVD under the hood rather than doing eigen decomposition on the covariance matrix because SVD can be much more efficient and is able to handle sparse matrices. In addition, there are reduced forms of SVD that are even more economical to compute

Optional-PCA

Proof that right singular vectors of data are eigenvectors of covariance matrix
Let the real values data matrix $\mathbf{X}$ be of $n \times p$ size, where n is the number of samples and p is the number of variables. Let us assume that it is centred, i.e. column means have been subtracted and are now equal to zero.
Then the $p \times p$ covariance matrix $\mathbf{C}$ is given by

$$\mathbf{C} = \frac{\mathbf{X}^\top \mathbf{X}}{n-1}.$$

It is a symmetric matrix and can be diagonalized:

$$\mathbf{C} = \mathbf{V}\mathbf{L}\mathbf{V}^\top,$$

where $\mathbf{V}$ is a matrix of eigenvectors (each column is an eigenvector) and $\mathbf{L}$ is a diagonal matrix with eigenvalues λ_i in decreasing order on the diagonal. The eigenvectors are called principal axes or principal directions of the data. Projections of the data on the principal axes are called principal components, also known as PC scores; these can be seen as new, transformed variables. The j-th principal component is given by the j-th column of $\mathbf{XV}$. The coordinates of the i-th data point in the new PC space are given by the i-th row of $\mathbf{XV}$.
If we now perform singular value decomposition of $\mathbf{X}$, we obtain a decomposition

$$\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^\top,$$

where $\mathbf{U}$ is a unitary matrix (with columns called left singular vectors), $\mathbf{S}$ is the diagonal matrix of singular values s_i, and $\mathbf{V}$ columns are called right singular vectors. From here, one can easily see that

$$\mathbf{C} = \mathbf{V}\mathbf{S}^2\mathbf{V}^\top = \frac{\mathbf{V}\mathbf{S}^2\mathbf{V}^\top}{n-1},$$

meaning that right singular vectors $\mathbf{V}$ are principal directions (eigenvectors), and singular values are related to the eigenvalues of the covariance matrix via $\lambda_i = \frac{s_i^2}{n-1}$.

Optional-EM

As we have observed, we defined two different log-likelihoods. The **expected complete log-likelihood** used in the EM algorithm and the **observed log-likelihood** we defined in GMM. Let's refer to them as L_{exp} and L_{obs} respectively. We know that it is hard to find the closed-form solution from L_{obs} which is why we use the EM algorithm. We use L_{exp} in the EM algorithm because it incorporates latent variables/soft assignments enabling more robust parameter estimation by accounting for uncertainty in cluster assignments.
L_{exp} also provides a **lower bound** on L_{obs}. This is due to **Jensen's inequality**. To prove this, consider a single datapoint x_i.

$$
\begin{aligned}
L_{exp}(x_i) &= \sum_{k=1}^{N} \gamma_{k,i} \log P(x_i, z_i = k) \\
L_{obs}(x_i) &= \log P(x_i) = \log \sum_{k=1}^{K} P(x_i, z_i = k) \\
&= \log \sum_{k=1}^{K} \gamma_{k,i} \cdot \frac{P(x_i, z_i = k)}{\gamma_{k,i}} \geq \sum_{k=1}^{K} \gamma_{k,i} \cdot \log \frac{P(x_i, z_i = k)}{\gamma_{k,i}} \\
&\geq L_{exp}(x_i) - \sum_{k=1}^{K} \gamma_{k,i} \cdot \log \gamma_{k,i} \\
L_{obs}(X) &\geq L_{exp}(X) - \sum_{i=1}^{n} \sum_{k=1}^{K} \gamma_{k,i} \cdot \log \gamma_{k,i} = L_{exp}(X) + H_{\gamma}
\end{aligned}
$$

H_{γ} is the entropy of the distribution and doesn't depend on model parameters, but solely on cluster assignments