Clustering, Expectation-Maximization, Gaussian Mixture Model, Feature Visualization, and Machine Learning Theories

Ying Nian Wu and Quanshi Zhang

Contents

1 Principle component analysis (PCA)

Credit: `https://en.wikipedia.org/wiki/Principal_component_analysis`

PCA can be thought of as fitting a p-dimensional ellipsoid to the data, where each axis of the ellipsoid represents a principal component. If some axis of the ellipsoid is small, then the variance along that axis is also small, and by omitting that axis and its corresponding principal component from our representation of the dataset, we lose only a commensurately small amount of information.

To find the axes of the ellipsoid, we must first subtract the mean of each variable from the dataset to center the data around the origin. Then, we compute the covariance matrix of the data, and calculate the eigenvalues and corresponding eigenvectors of this covariance matrix. Then we must normalize each of the orthogonal eigenvectors to become unit vectors. Once this is done, each of the mutually orthogonal, unit eigenvectors can be interpreted as an axis of the ellipsoid fitted to the data. This choice of basis will transform our covariance matrix into a diagonalised form with the diagonal elements representing the variance of each axis. The proportion of the variance that each eigenvector represents can be calculated by dividing the eigenvalue corresponding to that eigenvector by the sum of all eigenvalues.

This procedure is sensitive to the scaling of the data, and there is no consensus as to how to best scale the data to obtain optimal results.

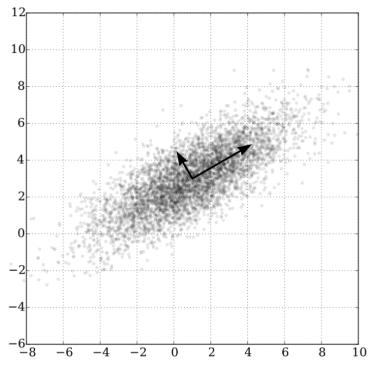

Figure 1: Data distribution. https://en.wikipedia.org/wiki/Principal_component_analysis

PCA is mathematically defined as an orthogonal linear transformation that transforms the data to a new coordinate system such that the greatest variance by some projection of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, and so on.

Consider a data matrix, $X \in \mathbb{R}^{n \times p}$, with column-wise zero empirical mean (the sample mean of each column has been shifted to zero), where each of the n rows represents a different repetition of the experiment, and each of the p columns gives a particular kind of feature (say, the results from a particular sensor).

Mathematically, the transformation is defined by a set of p-dimensional vectors of weights or coefficients $\mathbf{w}_{(k)} = (w_1, \ldots, w_p)_{(k)}$ that map each row vector $\mathbf{x}_{(i)}$ of X (the i-th data point) to a new vector of principal component scores $\mathbf{t}_{(i)} = (t_1, \ldots, t_l)_{(i)}$, given by

$$t_{k(i)} = \langle \mathbf{x}_{(i)}, \mathbf{w}_{(k)} \rangle \qquad \text{for} \qquad i = 1, \ldots, n \qquad k = 1, \ldots, l$$

in such a way that the individual variables $t_1, t_2, \ldots, t_l$ of $\mathbf{t}$ considered over the data set successively inherit the maximum possible variance from $\mathbf{x}$, with each coefficient vector $\mathbf{w}$ constrained to be a unit vector, i.e., $\|\mathbf{w}_{(k)}\| = 1$.

1.1 First component

In order to maximize variance, the first weight vector $\mathbf{w}_{(1)}$ thus has to satisfy

$$\mathbf{w}_{(1)} = \underset{\|\mathbf{w}\|=1}{\operatorname{argmax}} \, Var_i(t_1) = \underset{\|\mathbf{w}\|=1}{\operatorname{argmax}} \left\{ \sum_i \left(t_{1(i)} - E_i[t_{1(i)}]\right)^2 \right\} = \underset{\|\mathbf{w}\|=1}{\operatorname{argmax}} \left\{ \sum_i \left(t_{1(i)} - 0\right)^2 \right\} = \underset{\|\mathbf{w}\|=1}{\operatorname{argmax}} \left\{ \sum_i \left\langle \mathbf{x}_{(i)}, \mathbf{w} \right\rangle^2 \right\}$$

Equivalently, writing this in matrix form gives

$$\mathbf{w}_{(1)} = \underset{\|\mathbf{w}\|=1}{\operatorname{argmax}} \{\|\mathbf{X}\mathbf{w}\|^2\} = \underset{\|\mathbf{w}\|=1}{\operatorname{argmax}} \left\{ \mathbf{w}^\top \mathbf{X}^\top \mathbf{X} \mathbf{w} \right\}$$

Since $\mathbf{w}_{(1)}$ has been defined to be a unit vector, it equivalently also satisfies

$$\mathbf{w}_{(1)} = \operatorname{argmax}_{\mathbf{w}} \left\{ \frac{\mathbf{w}^\top \mathbf{X}^\top \mathbf{X} \mathbf{w}}{\mathbf{w}^\top \mathbf{w}} \right\}$$

The quantity to be maximised can be recognised as a Rayleigh quotient. A standard result for a positive semidefinite matrix such as $\mathbf{X}^\top \mathbf{X}$ is that the quotient's maximum possible value is the largest eigenvalue of the matrix, which occurs when $\mathbf{w}$ is the corresponding eigenvector.

The Lagrange multiplier:

$$\begin{aligned} 0 &= \frac{\partial}{\partial \mathbf{w}} \left[\mathbf{w}^\top \mathbf{X}^\top \mathbf{X} \mathbf{w} - \lambda \mathbf{w}^\top \mathbf{w} \right] = 2\mathbf{X}^\top \mathbf{X} \mathbf{w} - 2\lambda \mathbf{w} \\ \implies \quad & \mathbf{X}^\top \mathbf{X} \mathbf{w} = \lambda \mathbf{w} \end{aligned}$$

With $\mathbf{w}_{(1)}$ found, the first principal component of a data vector $\mathbf{x}_{(i)}$ can then be given as a score $t_{1(i)} = \langle \mathbf{x}_{(i)}, \mathbf{w}_{(1)} \rangle$ in the transformed co-ordinates, or as the corresponding vector in the original variables, $\langle \mathbf{x}_{(i)}, \mathbf{w}_{(1)} \rangle w_{(1)}$.

1.2 Further components

The k-th component can be found by subtracting the first $k-1$ principal components from X:

$$\hat{\mathbf{X}}_k = \mathbf{X} - \underbrace{\sum_{s=1}^{k-1} \mathbf{X} \mathbf{w}_{(s)} \mathbf{w}_{(s)}^\top}_{\substack{\text{Coordinates that can be represented by} \\ \text{the first } k-1 \text{ principle components}}}$$

and then finding the weight vector which extracts the maximum variance from this new data matrix

$$\mathbf{w}_{(k)} = \underset{\|\mathbf{w}\|=1}{arg\max} \left\{ \|\hat{\mathbf{X}}_k \mathbf{w}\|^2 \right\} = \operatorname{argmax}_{\mathbf{w}} \left\{ \frac{\mathbf{w}^\top \hat{\mathbf{X}}_k^\top \hat{\mathbf{X}}_k \mathbf{w}}{\mathbf{w}^\top \mathbf{w}} \right\}$$

It turns out that this gives the remaining eigenvectors of $\mathbf{X}^\top \mathbf{X}$, with the maximum values for the quantity in brackets given by their corresponding eigenvalues. Thus the weight vectors are eigenvectors of $\mathbf{X}^\top \mathbf{X}$.

The k-th principal component of a data vector $\mathbf{x}_{(i)}$ can therefore be given as a score $t_{k(i)} = \langle \mathbf{x}_{(i)}, \mathbf{w}_{(k)} \rangle$ in the transformed co-ordinates, or as the corresponding vector in the space of the original variables, $\{\langle \mathbf{x}_{(i)}, \mathbf{w}_{(k)} \rangle\} \mathbf{w}_{(k)}$, where $\mathbf{w}_{(k)}$ is the k-th eigenvector of $\mathbf{X}^\top \mathbf{X}$.

The full principal components decomposition of X can therefore be given as

$$\mathbf{T} = \mathbf{X}\mathbf{W}$$

where $\mathbf{W}$ is a p-by-p matrix of weights whose columns are the eigenvectors of $\mathbf{X}^\top\mathbf{X}$. The transpose of $\mathbf{W}$ is sometimes called the whitening or sphering transformation. Columns of $\mathbf{W}$ multiplied by the square root of corresponding eigenvalues, i.e., eigenvectors scaled up by the variances, are called loadings in PCA or in Factor analysis.

2 Stochastic Neighbor Embedding (t-SNE) (in order to visualize the data distribution)

In most cases, the feature x_i is a high dimensional feature. We need to project x_i into a low dimensional space h_i for visualization, e.g., projecting it into a two dimensional space.

The main concern of dimension reduction is that we need to ensure that if x_i and x_j are close to each other, then they may represent the same concept, so the corresponding features h_i and h_j should also be close to each other in the low-dimensional space.

Note that we can also try to capture the relationship between exemplars x_j and x_i by using the following definition:

$$\begin{aligned} P_{ij} &= P(x_j|x_i) \\ &= \frac{e^{-|x_j-x_i|^2/2\sigma_i^2}}{\sum_{m=1}^n e^{-|x_m-x_i|^2/2\sigma_i^2}} \\ &\propto e^{-|x_j-x_i|^2/2\sigma_i^2} \\ &\propto N(x_i,\sigma_i^2) \end{aligned}$$

This equation provides us with a notion of how "close" x_j is to x_i, from the perspective of x_i. In other words, one can think about it as telling us how x_i "sees" other examples - it measures how relevant x_j is from the viewpoint of x_i. Note that we are not really using a probability model here: instead, we are simply using a density function to capture the relationship between different x_i's and $x_j's$ (the denominator in the second line is simply a normalizing constant).

The central idea of SNE is to then use lower dimensional vectors so as to preserve relationships that are initially measured through the above mentioned equation. How would we write our h in this context? We can initially think about just replacing x_i and x_j with h_i and h_j, and write: $q_{ij} = P(h_j|h_i)$[1]. And the h_i terms could then be found by solving:

$$Loss = KL(P||Q) = \sum_{i,j} P_{i,j} log\left(\frac{P_{ij}}{q_{ij}}\right), \qquad \min_{\{h_i\}} Loss$$

, which is simply the Kullback-Leibler divergence from q to P.

What about the "t" in $t-SNE$? In truth, in $t-SNE$, in addition to replacing the $x_i's$ and $x_j's$ with $h_i's$ and $h_j's$, we also use t - distributions instead of normal ones, when computing our q_{ij}. Our q_{ij} become:

$$\begin{aligned} q_{ij} &= P(h_j|h_i) \\ &= t_1(h_i) \\ &= \frac{(1+|h_j-h_i|^2)^{-1}}{\sum_{m=1}^n (1+|h_m-h_i|^2)^{-1}}, \end{aligned}$$

,where the 1 in the second line indicates we are using t distributions with one degree of freedom. We can obtain the optimal $\{h_i\}$ via gradient descent.

[1]We'll typically here select $\sigma_i = 1$.

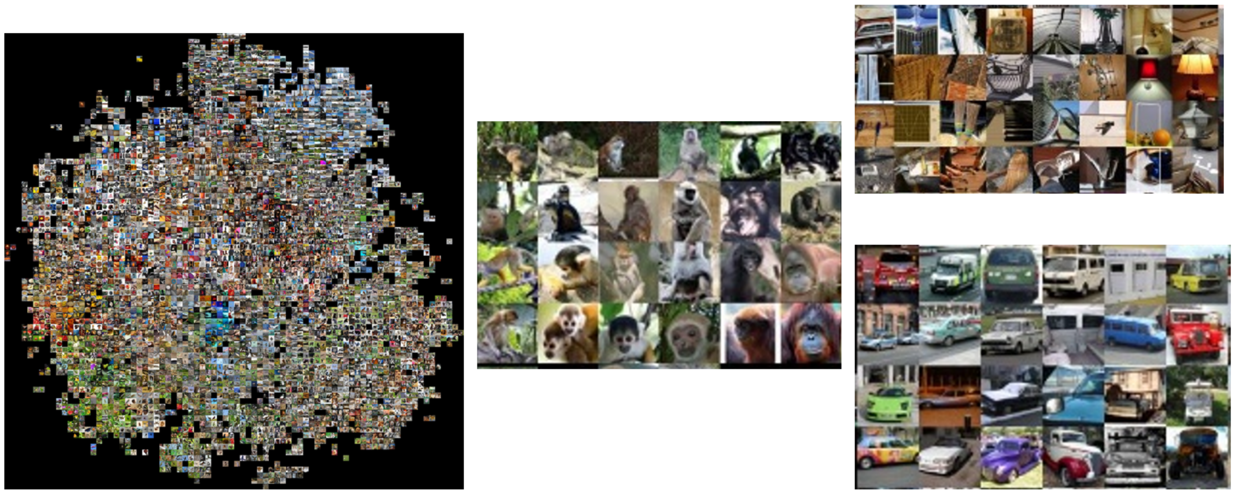

Figure 2: Examples of t-SNE visualization of image features that are extracted using a deep neural network. http://cs.stanford.edu/people/karpathy/cnnembed/

2.1 Local Linear Embedding

With local linear embedding, our goal is to preserve local linear relationships. We can represent x_i as $\sum_{j \in N_i} w_{ij} x_j$, where N_i stands for "*neighborhood*" of i.

$$x_i \approx \sum_{j \in N_i} w_{ij} x_j$$

Note that in $t-SNE$, the neighborhoods where implied by the σ terms, while here they are explicitly defined via N. We can look for the w terms via least-squares, by solving:

$$min_{\{w_{ij}\}} |x_i - \sum_{j \in N_i} w_{ij} x_j|^2$$

To implement local linear embedding, we then seek h's that preserve the local relationships that are being captured by the w_{ij} terms. Define $h_i \approx \sum_{j \in N_i} w_{ij} h_j$ (i.e., we simply replaced the x terms with h ones). Then, our goal is accomplished by solving:

$$min_{\{h_i\}} |h_i - \sum_{j \in N_i} w_{ij} h_j|^2$$

The w terms are the same as those obtained when solving the least squares problem (problem with the original $x's$); the resulting h terms will hopefully capture the linear relationships in the specified neighborhoods.

3 The Expectation Maximization (EM) Algorithm

Credit: `https://en.wikipedia.org/wiki/Expectation%E2%80%93maximization_algorithm`

In statistics, an expectation-maximization (EM) algorithm is an iterative method to find maximum likelihood or maximum a posteriori (MAP) estimates of parameters in statistical models, where the model depends on unobserved latent variables. The EM iteration alternates between performing

- an expectation (E) step, which creates a function for the expectation of the log-likelihood evaluated using the current estimate for the parameters;

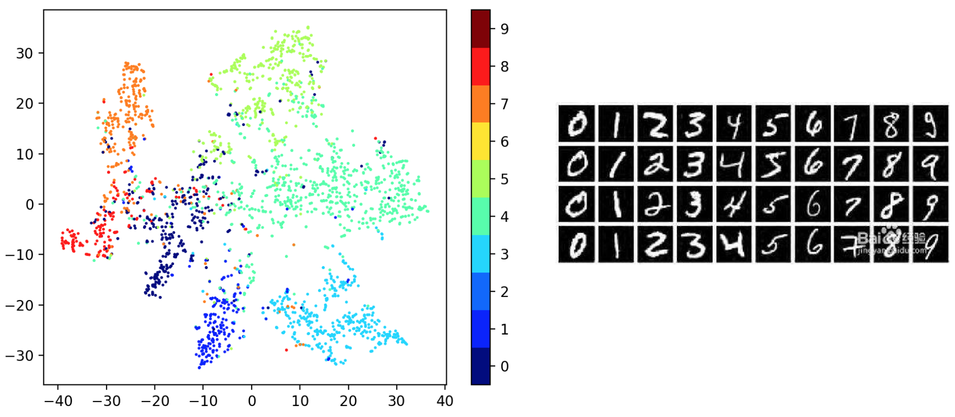
Figure 3: Examples of t-SNE visualization of image features that are extracted using a deep neural network. https://www.cnblogs.com/huangshiyu13/p/6945239.html

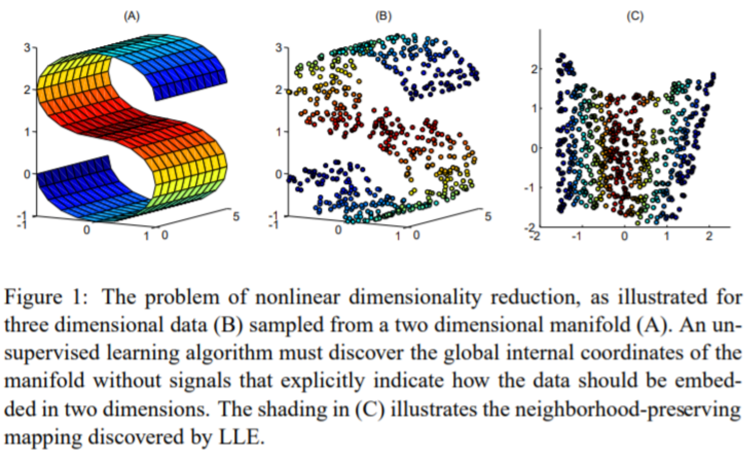

Figure 1: The problem of nonlinear dimensionality reduction, as illustrated for three dimensional data (B) sampled from a two dimensional manifold (A). An unsupervised learning algorithm must discover the global internal coordinates of the manifold without signals that explicitly indicate how the data should be embedded in two dimensions. The shading in (C) illustrates the neighborhood-preserving mapping discovered by LLE.

Figure 4: Examples of local linear embedding. https://cs.nyu.edu/ roweis/lle/papers/lleintro.pdf

- a maximization (M) step, which computes parameters maximizing the expected log-likelihood found on the E step.

These parameter-estimates are then used to determine the distribution of the latent variables in the next E step.

Given the statistical model which generates a set $\mathbf{X}$ of observed data, a set of unobserved latent data or missing values $\mathbf{Z}$, and a vector of unknown parameters θ, along with a likelihood function $L(\theta;\mathbf{X},\mathbf{Z}) = p(\mathbf{X},\mathbf{Z}|\theta)$, the maximum likelihood estimate (MLE) of the unknown parameters is determined by maximizing the marginal likelihood of the observed data

$$\max_{\theta} L(\theta,\mathbf{X}), \qquad L(\theta,\mathbf{X}) \overset{def}{=} p(\mathbf{X}|\theta) = \int_{\mathbf{Z}} p(\mathbf{X},\mathbf{Z}|\theta)d\mathbf{Z}$$

However, this quantity is often intractable (e.g., if $\mathbf{Z} = (z_1, z_2, \ldots, z_n)$ is a sequence of events, so that the number of values grows exponentially with the sequence length, the exact calculation of the sum will be extremely difficult).

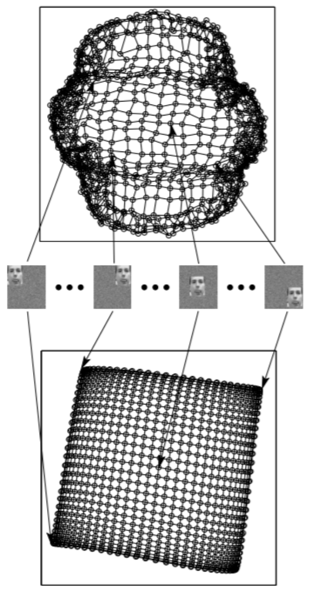

Figure 3: The results of PCA (top) and LLE (bottom), applied to images of a single face translated across a two-dimensional background of noise. Note how LLE maps the images with corner faces to the corners of its two dimensional embedding, while PCA fails to preserve the neighborhood structure of nearby images.

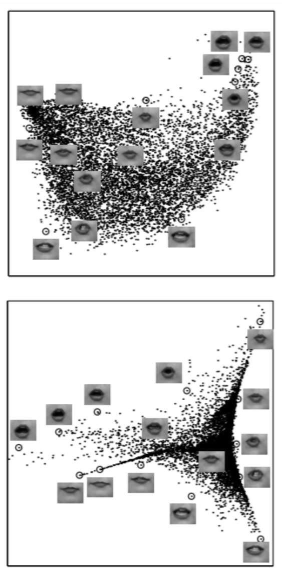

Figure 4: Images of lips mapped into the embedding space described by the first two coordinates of PCA (top) and LLE (bottom). Representative lips are shown next to circled points in different parts of each space. The differences between the two embeddings indicate the presence of nonlinear structure in the data.

Figure 5: PCA vs. local linear embedding. https://cs.nyu.edu/ roweis/lle/papers/lleintro.pdf

An example: K-means clustering

- $X = (x_1, x_2, \dots, x_n)$ denotes features of many data points.
- $\theta = (\mu_1, \sigma_1^2, \mu_2, \sigma_2^2, \dots, \mu_k, \sigma_k^2)$ denotes the mean and variance of k clusters.
- $Z = (z_1, z_2, \dots, z_n)$, $z_i \in \{1, 2, \dots, k\}$. $z_i = j$ indicates that the algorithm assigns the data point x_i to the j-th cluster.
- $p(Z|X, \theta) = \prod_{i=1}^{n} p(z_i|x_i, \theta)$ indicates the joint probability of assigning each x_i to the z_i-th cluster.

The EM algorithm seeks to find the MLE of the marginal likelihood by iteratively applying these two steps:

- **Expectation step (E step):** Define $Q(\theta|\theta^{(t)})$ as the expected value of the log-likelihood function of θ, with respect to the current conditional distribution of $\mathbf{Z}$ given $\mathbf{X}$ and the current estimates of the parameters $\theta^{(t)}$:
$$Q(\theta|\theta^{(t)}) = E_{\mathbf{Z} \sim p(\mathbf{Z}|\mathbf{X}, \theta^{(t)})}[\log p(\mathbf{X}, \mathbf{Z}|\theta)]$$

Where does the above equation come from?
Let us learn parameters to maximize the log-likelihood

$$\max_{\theta} \log p(\mathbf{X}|\theta),$$

where is a standard equation for MLE.

$$\begin{aligned}
&\max_{\theta} \log p(\mathbf{X}|\theta) \\
&\implies \Delta\theta = \eta \frac{\partial}{\partial\theta} \log p(\mathbf{X}|\theta) \\
&\qquad = \eta \frac{1}{p(\mathbf{X}|\theta)} \frac{\partial}{\partial\theta} p(\mathbf{X}|\theta) \\
&\qquad = \eta \frac{1}{p(\mathbf{X}|\theta)} \frac{\partial}{\partial\theta} \sum_{\mathbf{Z}} p(\mathbf{X},\mathbf{Z}|\theta) \\
&\qquad = \eta \frac{1}{p(\mathbf{X}|\theta)} \sum_{\mathbf{Z}} \frac{\partial}{\partial\theta} p(\mathbf{X},\mathbf{Z}|\theta) \\
&\qquad = \eta \frac{1}{p(\mathbf{X}|\theta)} \sum_{\mathbf{Z}} p(\mathbf{X},\mathbf{Z}|\theta) \frac{\partial}{\partial\theta} \log p(\mathbf{X},\mathbf{Z}|\theta) \\
&\qquad = \eta \sum_{\mathbf{Z}} \frac{p(\mathbf{X},\mathbf{Z}|\theta)}{p(\mathbf{X}|\theta)} \frac{\partial}{\partial\theta} \log p(\mathbf{X},\mathbf{Z}|\theta) \\
&\qquad = \eta \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X},\theta) \frac{\partial}{\partial\theta} \log p(\mathbf{X},\mathbf{Z}|\theta)
\end{aligned}$$

$$\begin{aligned}
&\approx \eta \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X},\theta^{(t)}) \frac{\partial}{\partial\theta} \log p(\mathbf{X},\mathbf{Z}|\theta), \qquad \text{here we use } \theta^{(t)} \text{ instead of } \theta. \\
&= \eta \frac{\partial}{\partial\theta} \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X},\theta^{(t)}) \log p(\mathbf{X},\mathbf{Z}|\theta) \\
&= \eta \frac{\partial}{\partial\theta} E_{\mathbf{Z}\sim p(\mathbf{Z}|\mathbf{X},\theta^{(t)})} [\log p(\mathbf{X},\mathbf{Z}|\theta)]
\end{aligned}$$

Therefore, learning θ to maximize $Q(\theta|\theta^{(t)})$ is highly close to maximizing $\log p(\mathbf{X}|\theta)$.

- **Maximization step (M step):** Find the parameters that maximize this quantity:

$$\theta^{(t+1)} = \underset{\theta}{\operatorname{argmax}}\, Q(\theta|\theta^{(t)})$$

The motive is as follows. If the value of the parameters θ is known, usually the value of the latent variables $\mathbf{Z}$ can be found by maximizing the log-likelihood over all possible values of $\mathbf{Z}$. Conversely, if we know the value of the latent variables $\mathbf{Z}$, we can find an estimate of the parameters θ fairly easily, typically by simply grouping the observed data points according to the value of the associated latent variable and averaging the values, or some function of the values, of the points in each group. This suggests an iterative algorithm, in the case where both θ and $\mathbf{Z}$ are unknown:

- **Step 1:** First, initialize the parameters θ to some random values.

- **Step 2:** Compute the probability of each possible value of $\mathbf{Z}$, given θ.
- **Step 3:** Then, use the just-computed values of $\mathbf{Z}$ to compute a better estimate for the parameters θ.
- **Step 4:** Iterate steps 2 and 3 until convergence.

The algorithm as just described monotonically approaches a local minimum of the cost function.

3.1 Proof of the correctness

Expectation-maximization works to improve $Q(\theta|\theta^{(t)})$ rather than directly improving $\log p(\mathbf{X}|\theta)$. Here is shown that improvements to the former imply improvements to the latter.

For any $\mathbf{Z}$ with non-zero probability $p(\mathbf{Z}|\mathbf{X},\theta)$, we can write

$$\log p(\mathbf{X}|\theta) = \log p(\mathbf{X},\mathbf{Z}|\theta) - \log p(\mathbf{Z}|\mathbf{X},\theta)$$

We take the expectation over possible values of the unknown data $\mathbf{Z}$ under the current parameter estimate $\theta^{(t)}$ by multiplying both sides by $p(\mathbf{Z}|\mathbf{X},\theta^{(t)})$ and summing (or integrating) over $\mathbf{Z}$. The left-hand side is the expectation of a constant, so we get:

$$\begin{aligned}\log p(\mathbf{X}|\theta) &= \left[\sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X},\theta^{(t)})\right] \log p(\mathbf{X},\mathbf{Z}|\theta) - \left[\sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X},\theta^{(t)})\right] \log p(\mathbf{Z}|\mathbf{X},\theta), \qquad \text{because } \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X},\theta^{(t)}) = 1 \\ &= Q(\theta|\theta^{(t)}) + H(\theta|\theta^{(t)})\end{aligned}$$

where $H(\theta|\theta^{(t)})$ is defined by the negated sum it is replacing. This last equation holds for every value of θ including $\theta = \theta^{(t)}$,

$$\log p(\mathbf{X}|\theta^{(t)}) = Q(\theta^{(t)}|\theta^{(t)}) + H(\theta^{(t)}|\theta^{(t)})$$

and subtracting this last equation from the previous equation gives

$$\log p(\mathbf{X}|\theta) - \log p(\mathbf{X}|\theta^{(t)}) = Q(\theta|\theta^{(t)}) - Q(\theta^{(t)}|\theta^{(t)}) + H(\theta|\theta^{(t)}) - H(\theta^{(t)}|\theta^{(t)})$$

However, Gibbs' inequality tells us that $H(\theta|\theta^{(t)}) \geq H(\theta^{(t)}|\theta^{(t)})$, so we can conclude that

$$\log p(\mathbf{X}|\theta) - \log p(\mathbf{X}|\theta^{(t)}) \geq Q(\theta|\theta^{(t)}) - Q(\theta^{(t)}|\theta^{(t)}).$$

In words, choosing θ to improve $Q(\theta|\theta^{(t)})$ causes $\log p(\mathbf{X}|\theta)$ to improve at least as much.

3.2 Mixture of Gaussians, EM, K-means

We assume an unobserved $z \in \{1,2,\dots,K\}$ has a causal relationship with x (it generates x): $z \to x$.

K refers to the total number of available categories. The general idea is that each exemplar will fall within a particular cluster or category: for example, if x is in the first cluster, $p(z=1|x,\theta) = 1$. And the distribution of any x is affected by the cluster it falls into: $p(x|z=k,\theta) \equiv f_k(x) \sim N(\mu_k, \sigma^2 I)$. μ_k is assumed to be unknown, while σ^2 is known. Here, $\theta = \{\mu_1,\mu_2,\dots,\mu_K\}$.

Furthermore, we denote the unconditional probability that an observation falls within cluster k as $p_k = p(z=k|\theta)$, with $\sum_{i=1}^{K} p_i = 1$. The probability associated with each exemplar/observation is then $p(x|\theta) = \sum_{k=1}^{K} p(x,z=k|\theta) = \sum_{k=1}^{K} p(z=k)p(x|z=k,\theta) = \sum_{k=1}^{K} p_k f_k(x)$.

To estimate this generative model, we need to find parameters μ_k and p_k, for $k \in \{1,\dots,K\}$. This can be done by solving the following maximum likelihood problem:

$$max_{\{\mu_k,p_k\}} \frac{1}{n}\sum_{i=1}^{n} log\left(f_k(x_i)\right)$$

In the discussion that follows, we present different approaches that can be used to estimate such parameters (all of which are related to the EM algorithm). The EM version is also shown below.

K-Means (related to, but exactly the same to the EM algorithm)

Our version of the K-means clustering algorithm can be implemented as follows: first, for each observation $x_i = (x_{i1}, x_{i2}, ..., x_{ip})$, pick a guess for $z_i \in \{1, 2, \ldots, K\}$.

Here, we refer to this classifier as a "hard" classifier, i.e., when $p(z_i|x_i, \theta)$ usually approximates either 1 or 0, we can roughly consider $p(z_i|x_i, \theta) = 1$ or 0 for simplicity.

- **Step 1:** Updating Parameters: Given $\{z_i\}$, compute $\hat{p}_k = \sum_{i=1}^{n} p(z_i = k|x_i, \theta)p(x_i|\theta)$, and $\hat{\mu}_k = \frac{\sum_{i=1}^{n} p(z_i=k|x_i,\theta)x_i}{\sum_{i=1}^{n} p(z_i=k|x_i,\theta)}$. Note that when we compute $\hat{p}_k$, we are simply counting the proportion of observations that fall within each cluster; $\hat{\mu}_k$ is simply the average value of x within cluster k.
- **Step 2:** Inference of Latent Variables: Let $z_i = \text{arg}min_k |x_i - \mu_k|$. Put simply, we assign x_i here to the cluster whose means/centers are closest to x_i.
- **Step 3:** Iterate.

EM - Soft Classifier (more closer to the EM algorithm)

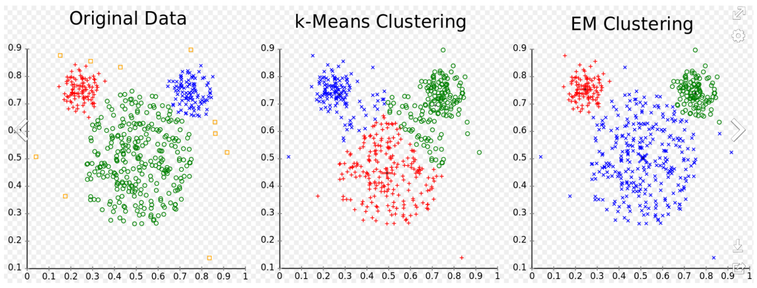

Figure 6: Clustering. https://en.wikipedia.org/wiki/K-means_clustering

Another approach is to use a "soft" classifier. In this approach, we allow $p(z_i|x_i, \theta)$ to be a soft probability. This is accomplished by changing Step (2) in the K-means procedure. Step (1) is left unaltered, and can be called the M step here. Step (2) is now the E step (it computes $E(z_i|x_i, \hat{\mu}, \hat{p})$) in the EM algorithm.

To implement the soft or fractional assignment, instead of letting $z_i = \text{argmin}_k |x_i - \mu_k|$ in Step (2), we compute instead:

$$p(z_i = k|x_i, \theta) = \frac{p(z_i = k|\theta)p(x_i|z_i = k, \theta)}{\sum_{k'=1}^{K} p(z_i = k'|\theta)p(x_i|z_i = k', \theta)} = \frac{\hat{p}_k f_k(x)}{\sum_{k'=1}^{K} \hat{p}_{k'} f_{k'}(x)},$$

where $f_k(x) = \frac{1}{(2\pi\sigma^2)^{p/2}} e^{-\frac{1}{2\sigma^2}|x-\mu_k|^2}$. In other words, we let $p(z_i = k|x_i, \theta)$ reflect the probability that observation x_i falls within cluster k. Note that the farther x_i is from μ_k, the lower $f_k(x)$ will be, and thus the lower $p(z_i = k|x_i, \theta)$ will be.