Unsupervised Learning and Expectation Maximization

Shuiwang Ji
Department of Computer Science & Engineering
Texas A&M University

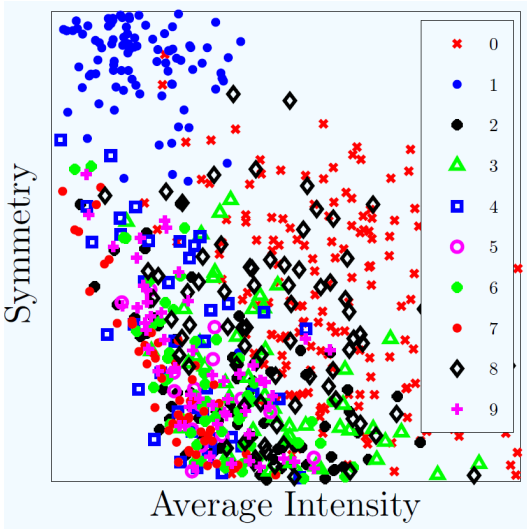
Based on Pattern Recognition and Machine Learning and Learning from Data

Outline

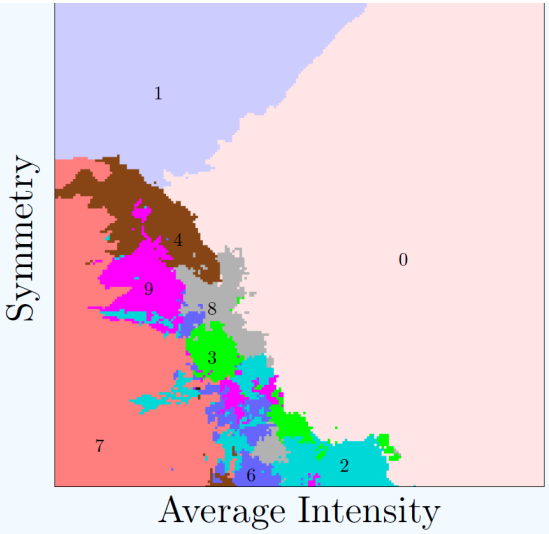
- k-means clustering
- Parzen window density estimation
- Gaussian mixture models
- Mixture models and expectation maximization

Supervised Learning versus Unsupervised Learning

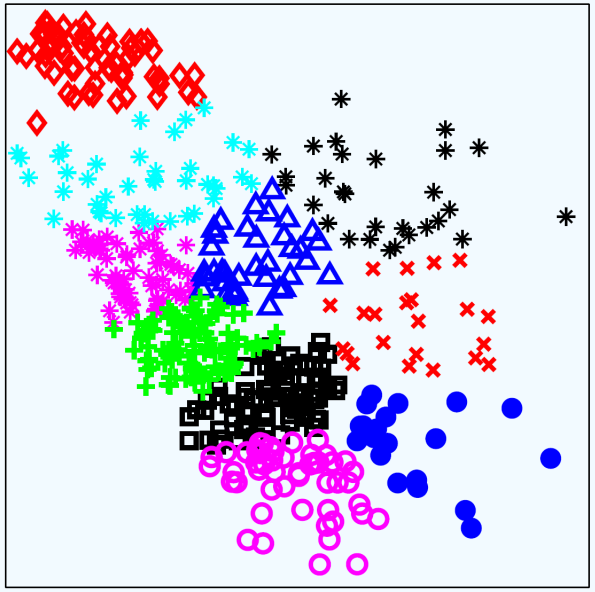
- Supervised Learning: Modeling a mapping from X to Y
- Unsupervised Learning: Modeling of X, including clustering, density estimation, etc.



(a) Multiclass digits data



(b) 21-NN decision regions



k-means Clustering

- The goal of k-means clustering is to partition the input data points $\mathrm{x}_1, \ldots, \mathrm{x}_N$ into k sets $S_1, \ldots, S_k$ and select centers $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k$ for each cluster.
- The centers are representative of the data if every data point in cluster S_j is close to its corresponding center $\boldsymbol{\mu}_j$.
- For cluster S_j with center $\boldsymbol{\mu}_j$, define the squared error measure E_j to quantify the quality of the cluster,

$$E_j = \sum_{\mathrm{x}_n \in S_j} \|\mathrm{x}_n - \boldsymbol{\mu}_j\|^2$$

- The error E_j measures how well the center $\boldsymbol{\mu}_j$ approximates the points in S_j.

k-means Clustering

- The k-means error function sums this cluster error over all clusters,

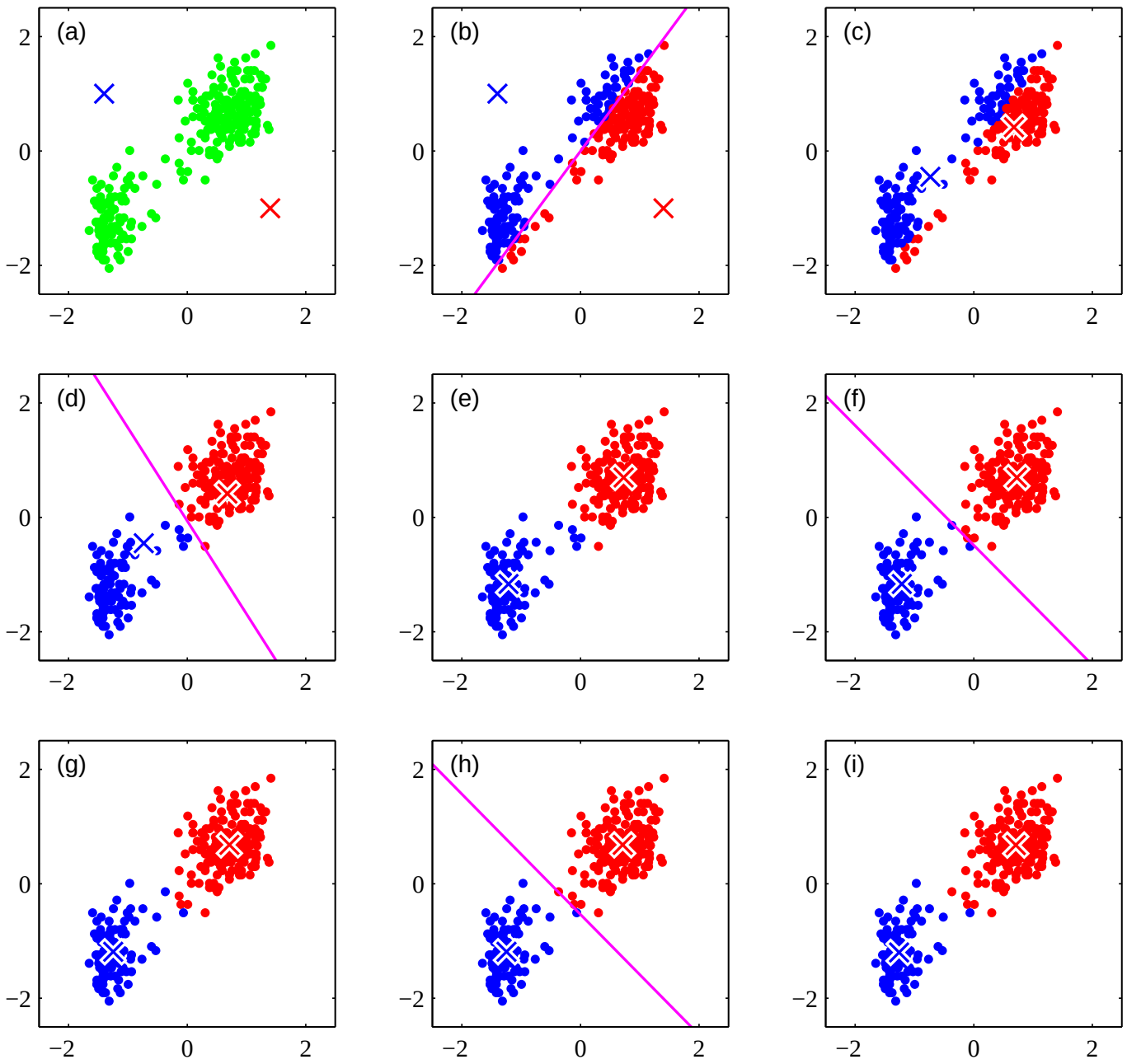
$$E_{\text{in}}\left(S_1, \ldots, S_k; \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k\right) = \sum_{j=1}^{k} E_j = \sum_{n=1}^{N} \left\|\mathrm{x}_n - \boldsymbol{\mu}\left(\mathrm{x}_n\right)\right\|^2$$

  where $\boldsymbol{\mu}\left(\mathrm{x}_n\right)$ is the center of the cluster to which x_n belongs.
- We seek the partition $S_1, \ldots, S_k$ and centers $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k$ that minimize the k-means error.

Computations of k-means Clustering

- Minimizing the k-means error is an NP-hard problem.
- However, if we fix a partition, then the optimal centers are easy to obtain.
- Similarly, if we fix the centers, then the optimal partition is easy to obtain.
- This suggests a very simple iterative algorithm which is known as Lloyd's algorithm.
  1. Initialize $\boldsymbol{\mu}_j$.
  2. Construct S_j to be all points closest to $\boldsymbol{\mu}_j$.
  3. Update each $\boldsymbol{\mu}_j$ to equal the centroid of S_j.
  4. Repeat steps 2 and 3 until E_{in} stops decreasing.

Example of k-means



Discussions

- Lloyd's algorithm falls into a class of algorithms known as E-M (expectation-maximization) algorithms.
- It minimizes a complex error function by separating the variables to be optimized into two sets.
- If one set is known, then it is easy to optimize the other set, which is the basis for an iterative algorithm, such as with Lloyd's algorithm.

Probability Density Estimation

- The probability density of x is a generalization of clustering to a finer representation. Clusters can be thought of as regions of high probability.
- The basic task in probability density estimation is to estimate:
  For a given x, how likely it is that you would generate inputs similar to x.
- To answer this question we need to look at what fraction of the inputs in the data are similar to x.

Parzen Window Density Estimation

- The most common density estimation technique is the Parzen window.
- The normalized Gaussian kernel is

$$\phi(z) = \frac{1}{(2\pi)^{d/2}} e^{-\frac{1}{2}z^2}$$

- One can verify that $\hat{P}(\mathrm{x}) = \phi(\|\mathrm{x}\|)$ is a probability density
- For any $r > 0$,

$$\hat{P}(\mathrm{x}) = \frac{1}{r^d} \cdot \phi\left(\frac{\|\mathrm{x}\|}{r}\right)$$

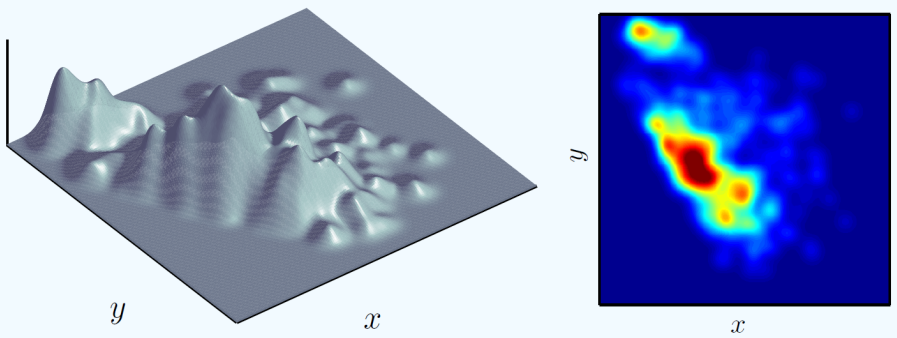
  is also a density (r is the width of the bump).

Parzen Window Density Estimation

- In Parzen window, you have a bump with weight $\frac{1}{N}$ on each data point, and $\hat{P}(\mathrm{x})$ is a sum of the bumps:

$$\hat{P}(\mathrm{x}) = \frac{1}{Nr^d} \sum_{i=1}^{N} \phi\left(\frac{\|\mathrm{x} - \mathrm{x}_i\|}{r}\right).$$

- Since each bump integrates to 1 , the scaling by $\frac{1}{N}$ ensures that $\hat{P}(\mathrm{x})$ integrates to 1.



(a) RBF density estimate (b) Density contours

Figure 6.14: Gaussian kernel Parzen window with $r = 0.02$ using 500 digits data. (a) The probability density surface is smooth but bumpy around the periphery of the data; the tail is exponential, dropping very quickly to zero away from the data. Compare this with the spiky more slowly decaying density produced by nearest neighbor in Figure 6.13. (b) Density contours (red is high, blue is low) highlighting the clusters in the data as regions of locally high density. Clusters are more readily visible with the smoother Parzen window than the nearest neighbor estimate.

Gaussian Mixture Model

Gaussian Distributions

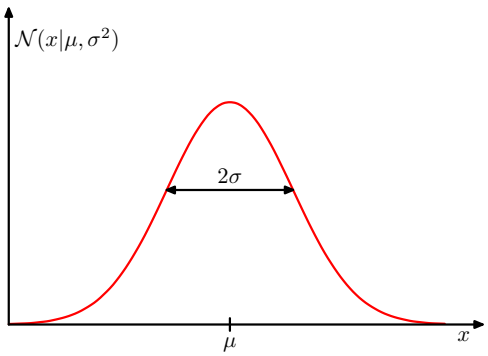
- Gaussian distribution, or *normal distribution*, is a widely used model for the distribution of continuous variables
- Gaussian distribution in 1-dimensional space

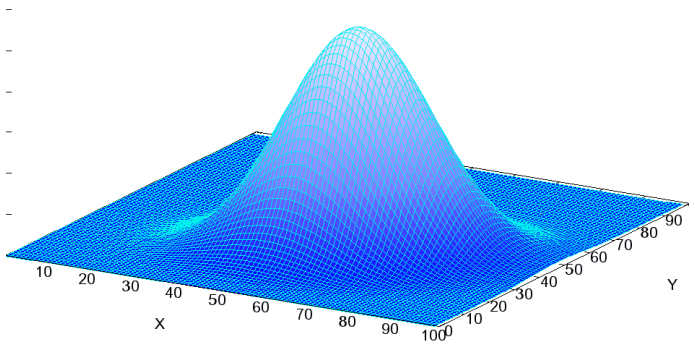
$$\mathcal{N}(x|\mu,\sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\} \tag{1}$$

  where μ is the mean, and σ^2 is the variance
- Gaussian distribution in d-dimensional space

$$\mathcal{N}(\mathrm{x}|\mu,\Sigma) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(\mathrm{x}-\mu)^T\Sigma^{-1}(\mathrm{x}-\mu)\right\} \tag{2}$$

  where μ is a d-dimensional mean vector, Σ is a $d \times d$ covariance matrix, and $|\Sigma|$ is the determinant of Σ





Maximum Likelihood for the Gaussian

- Given a data set $\mathrm{X} = (\mathrm{x}_1, \cdots, \mathrm{x}_n)^T$ in which the observations x_n are assumed to be drawn independently from a multivariate Gaussian distribution, we can estimate the parameters of the distribution by maximum likelihood
- The log likelihood function is given by

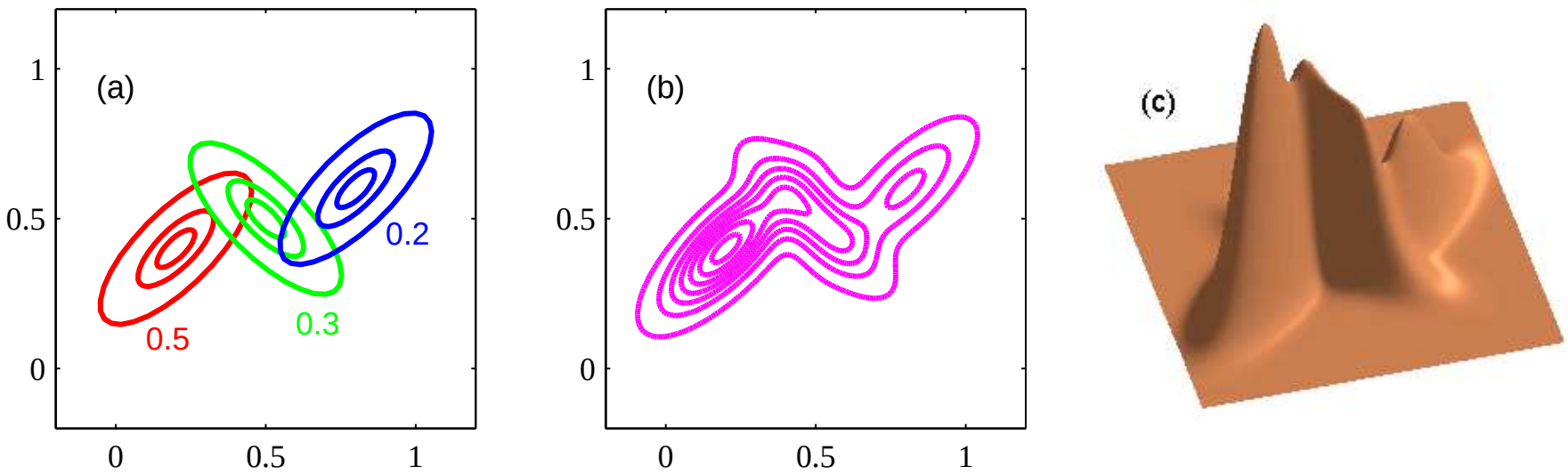
$$\ln p(\mathrm{X}|\mu, \Sigma) = -\frac{nd}{2}\ln(2\pi) - \frac{n}{2}\ln|\Sigma| - \frac{1}{2}\sum_{i=1}^{n}(\mathrm{x}_n - \mu)^T\Sigma^{-1}(\mathrm{x}_n - \mu)$$

- Taking the derivative of the log likelihood with respect to μ and Σ and setting to zero, we obtain

$$\mu_{\mathrm{ML}} = \frac{1}{n}\sum_{i=1}^{n}\mathrm{x}_i \tag{3}$$

$$\Sigma_{\mathrm{ML}} = \frac{1}{n}\sum_{i=1}^{n}(\mathrm{x}_i - \mu_{\mathrm{ML}})(\mathrm{x}_i - \mu_{\mathrm{ML}})^T \tag{4}$$

Modeling of Complex Distributions



Gaussian Mixture Model (GMM)

- Just as the RBF-network is the parametric version of the nonparametric RBF, the Gaussian mixture model (GMM) is the parametric version of the Parzen window density estimate.
- In Parzen window, you have a bump with weight $\frac{1}{N}$ on each data point, and $\hat{P}(\mathrm{x})$ is a sum of the bumps:

$$\hat{P}(\mathrm{x}) = \frac{1}{Nr^d} \sum_{i=1}^{N} \phi\left(\frac{\|\mathrm{x} - \mathrm{x}_i\|}{r}\right).$$

- The Parzen window estimate places a Gaussian bump at every data point; the GMM places just k bumps at centers $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k$.
- The Gaussian kernel is the most commonly used and easiest to handle.

Sampling

- There are k Gaussian distributions, with respective means $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k$ and covariance matrices $\Sigma_1, \ldots, \Sigma_k$.
- To generate a data point x, first pick a Gaussian $j \in \{1, \ldots, k\}$ according to probabilities $\{w_1, \ldots, w_k\}$, where $w_j > 0$ and $\sum_{j=1}^{k} w_j = 1$.
- After selecting Gaussian j, x is generated according to a Gaussian distribution with parameters $\boldsymbol{\mu}_j, \Sigma_j$.
- The probability density of x given that you picked Gaussian j is

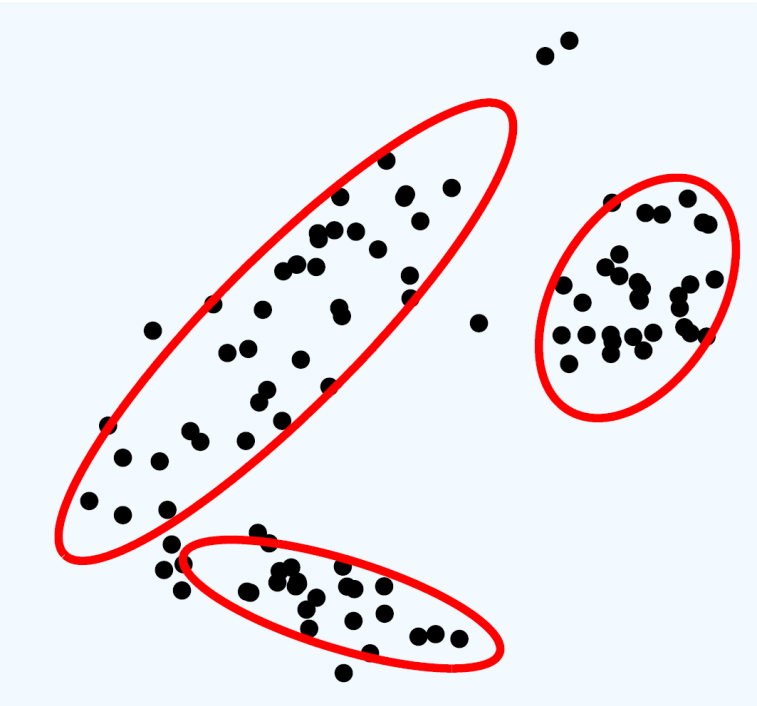
$$P(\mathrm{x} \mid j) = \mathcal{N}(\mathrm{x}; \boldsymbol{\mu}_j, \Sigma_j) = \frac{1}{(2\pi)^{d/2} \left|\Sigma_j\right|^{1/2}} e^{-\frac{1}{2}\left(\mathrm{x}-\boldsymbol{\mu}_j\right)^{\mathrm{T}} \Sigma_j^{-1} \left(\mathrm{x}-\boldsymbol{\mu}_j\right)}.$$

- By the law of total probability, the unconditional probability density for x is

$$P(\mathrm{x}) = \sum_{j=1}^{k} P(\mathrm{x} \mid j) \mathbb{P}[j] = \sum_{j=1}^{k} w_j \mathcal{N}(\mathrm{x}; \boldsymbol{\mu}_j, \Sigma_j)$$

Example

- Each of the k Gaussian bumps represents a cluster of the data. The probability density puts a bump (Gaussian) of total probability (weight) w_j at the center $\boldsymbol{\mu}_j$; Σ_j determines the shape of the bump.
- The Gaussians are illustrated by a contour of constant probability. The different shapes of the clusters are controlled by the covariances Σ_j.



Parameter Estimation

- The parameters of the model need to be learned from the actual data.
- These parameters are the mixture weights w_j, the centers $\boldsymbol{\mu}_j$ and the covariance matrices Σ_j.
- To determine the unknown parameters in the GMM, we will minimize an in-sample error called the likelihood.
- Our criterion for choosing is that, for the chosen density estimate $\hat{P}$, the data should have a high probability of being generated.
- Since the data points are independent, the probability (density) for the data $\mathrm{x}_1, \ldots, \mathrm{x}_N$ if the data were generated according to $\hat{P}(\mathrm{x})$ is

$$\hat{P}\left(\mathrm{x}_1, \ldots, \mathrm{x}_N\right) = \prod_{n=1}^{N} \hat{P}\left(\mathrm{x}_n\right) = \prod_{n=1}^{N} \left(\sum_{j=1}^{k} w_j \mathcal{N}\left(\mathrm{x}_n; \boldsymbol{\mu}_j, \Sigma_j\right) \right)$$

Maximum Likelihood

- This is the likelihood of a particular GMM specified by parameters $\{w_j, \boldsymbol{\mu}_j \Sigma_j\}$.
- The method of maximum likelihood selects the parameters which maximize $\hat{P}(\mathrm{x}_1, \ldots, \mathrm{x}_N)$, or equivalently which minimize $-\ln \hat{P}(\mathrm{x}_1, \ldots, \mathrm{x}_N)$.
- Thus, we may minimize the in-sample error:

$$\begin{aligned} E_{\text{in}}\ (w_j, \boldsymbol{\mu}_j, \Sigma_j) &= -\ln \hat{P}(\mathrm{x}_1, \ldots, \mathrm{x}_N) \\ &= -\sum_{n=1}^{N} \ln \left(\sum_{j=1}^{k} w_j \mathcal{N}(\mathrm{x}_n; \boldsymbol{\mu}_j, \Sigma_j) \right). \end{aligned}$$

- Even for the friendly Gaussian mixture model, the summation inside the logarithm makes it very difficult to minimize the in-sample error.
- Need the Expectation Maximization (E-M) algorithm.

Expectation Maximization

- EM is based on the notion of a latent (hidden, unmeasured, missing) piece of data that would make the optimization much easier.
- In the context of the Gaussian Mixture Model, suppose we knew which data points came from bump 1 , bump 2, . . ., bump k. The problem would suddenly become much easier, because we can estimate the center and covariance matrix of each bump using the data from that bump alone; further we can estimate the probabilities w_j by the fraction of data points in bump j.
- Unfortunately we do not know which data came from which bump, so we start with a guess, and iteratively improve this guess. The general algorithm is
  1. Start with estimates for the bump membership of each x_n.
  2. Estimates of $w_j, \boldsymbol{\mu}_j, \Sigma_j$ given the bump memberships.
  3. Update the bump memberships given $w_j, \boldsymbol{\mu}_j, \Sigma_j$; iterate to step 2 until convergence.

Soft Assignment

- The bump memberships need not be all or nothing. Specifically, at iteration t, let $\gamma_{nj}(t) \geq 0$ be the 'fraction' of data point x_n that belongs to bump j, with $\sum_{j=1}^{k} \gamma_{nj} = 1$ (the entire point is allocated among all the bumps); you can view γ_{nj} as the probability that x_n was generated by bump j.
- The 'number' of data points belonging to bump j is given by

$$N_j = \sum_{n=1}^{N} \gamma_{nj}.$$

If Soft Assignments are Known: Maximization Step

- The γ_{nj} are the hidden variables that we do not know, but if we did know the γ_{nj}, then we could compute estimates of $w_j, \boldsymbol{\mu}_j, \Sigma_j$:

$$
\begin{aligned}
w_j &= \frac{N_j}{N}; \\
\boldsymbol{\mu}_j &= \frac{1}{N_j} \sum_{n=1}^{N} \gamma_{nj} \mathrm{x}_n; \\
\Sigma_j &= \frac{1}{N_j} \sum_{n=1}^{N} \gamma_{nj} \mathrm{x}_n \mathrm{x}_n^{\mathrm{T}} - \boldsymbol{\mu}_j \boldsymbol{\mu}_j^{\mathrm{T}}.
\end{aligned}
$$

- Intuitively, the weights are the fraction of data belonging to bump j;
- The means are the average data point belonging to bump j where we take into account that only a fraction of a data point may belong to a bump;
- The covariance matrix is the weighted covariance of the data belonging to the bump.

Expectation Step

- Once we have these new estimates of the parameters, we can update the bump memberships γ_{nj}.
- To get $\gamma_{nj}(t+1)$, we compute the probability that data point x_n came from bump j given the parameters $w_j, \boldsymbol{\mu}_j, \Sigma_j$.
- We want

$$\gamma_{nj}(t+1) = \mathbb{P}\left[j \mid \mathrm{x}_n\right]$$

By an application of Bayes rule,

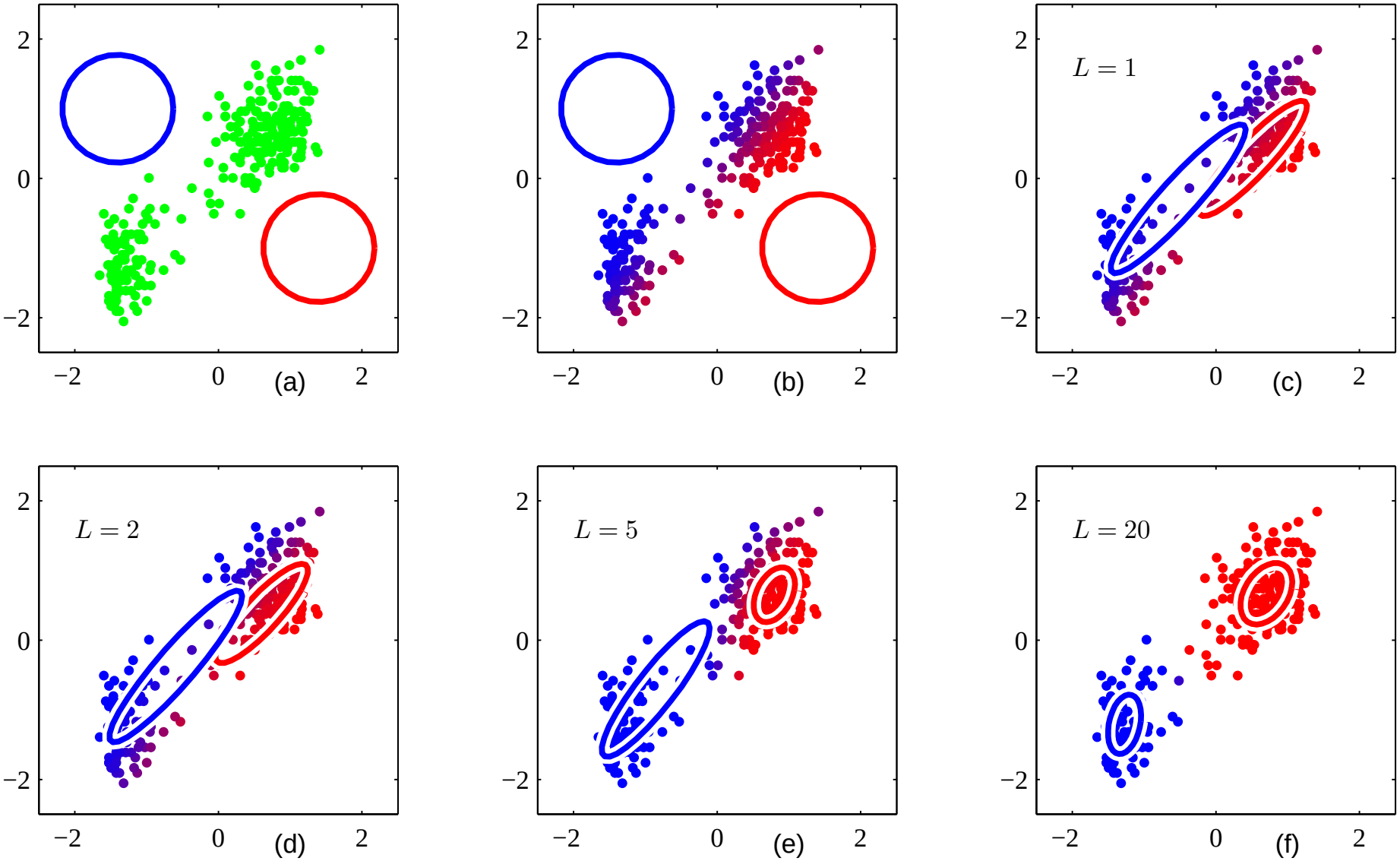
$$\mathbb{P}\left[j \mid \mathrm{x}_n\right] = \frac{P\left(\mathrm{x}_n \mid j\right)\mathbb{P}[j]}{P\left(\mathrm{x}_n\right)} = \frac{\mathcal{N}\left(\mathrm{x}_n; \boldsymbol{\mu}_j, \Sigma_j\right) \cdot w_j}{P\left(\mathrm{x}_n\right)}.$$

- We won't have to compute $P\left(\mathrm{x}_n\right)$ in the denominator because it is independent of j and can be fixed by the normalization condition $\sum_{j=1}^{k} \gamma_{nj} = 1$.
- We thus arrive at the update for the bump memberships,

$$\gamma_{nj}(t+1) = \frac{w_j \mathcal{N}\left(\mathrm{x}_n; \boldsymbol{\mu}_j, \Sigma_j\right)}{\sum_{\ell=1}^{k} w_\ell \mathcal{N}\left(\mathrm{x}_n; \boldsymbol{\mu}_\ell, \Sigma_\ell\right)}$$

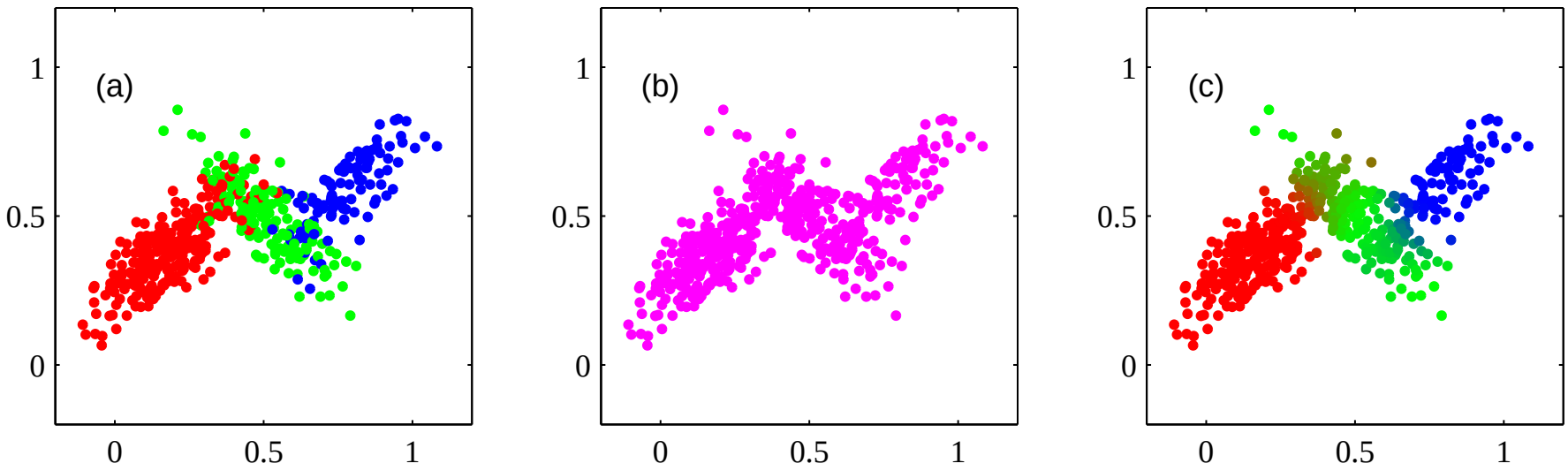
Illustration of EM Algorithm

- One standard-deviation contours for the two Gaussian components are shown as blue and red circles



Sampling from GMM

- We can use the technique of ancestral sampling to generate random samples distributed according to the Gaussian mixture model
- Procedure of sampling
  - First generate a value for z, denoted as $\hat{z}$, from the marginal distribution $p(z)$
  - Then generate a value for x from the conditional distribution $p(x|\hat{z})$

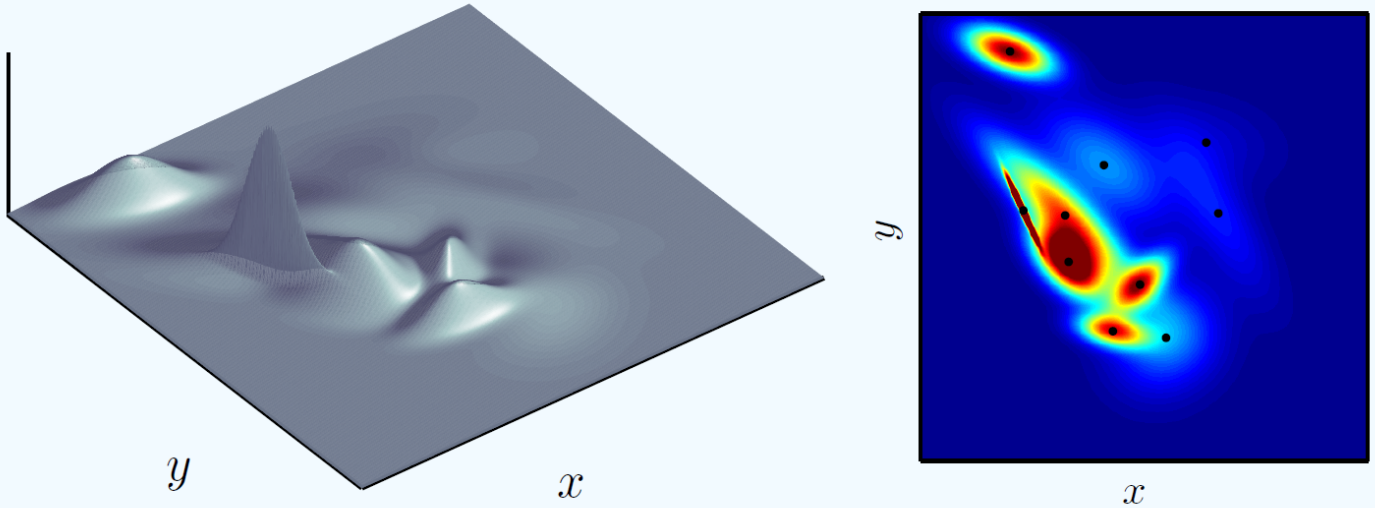


$p(z)p(x|z)$ $\quad$ $p(x)$ $\quad$ Color with $\gamma(z_{nk})$

Discussions

- The E-M algorithm is a remarkable example of a learning algorithm that 'bootstraps' itself to a solution.
- The algorithm starts by guessing some values for unknown quantities that you would like to know.
- The guess is probably quite wrong, but nevertheless the algorithm makes inferences based on the incorrect guess.
- These inferences are used to slightly improve the guess.
- The guess and inferences slightly improve each other in this way until at the end you have bootstrapped yourself to a decent guess at the unknowns, as well as a good inference based on those unknowns.

Example



(a) 10-center GMM (b) Density contours

Figure 6.15: The 10-center GMM for 500 digits data. (a) The probability density surface. The bumps are no longer spherical as in the Parzen window. The bump shapes are determined by the covariance matrices. (b) Density contours (red is high, blue is low) with the centers as black dots. The centers identify the 'clusters' (compare with the k-means clusters on page 31).

Comparison of K-means and GMM

- EM algorithm makes a *soft* assignment based on the posterior probabilities
- K-means gives a *hard* assignment of data points to clusters
- K-means algorithm can be considered as a particular limit of EM for Gaussian mixtures

Comparison of K-means and GMM

- Consider a Gaussian mixture model in which the covariance matrices of the mixture components are given by ϵI, i.e., $\Sigma_k = \epsilon I$, where $\epsilon \in \mathbb{R}$ is a fixed constant, and it need not to be estimated
- Each component is given by

$$p(\mathrm{x}|\mu_k, \Sigma_k) = \frac{1}{(2\pi)^{d/2}\epsilon^{1/2}} \exp\left\{-\frac{1}{2\epsilon}\|\mathrm{x}_n - \mu_k\|^2\right\} \tag{5}$$

- The posterior probability of z_k given observation x_n is

$$\gamma(z_{nk}) = \frac{\pi_k \exp(-\|\mathrm{x}_n - \mu_k\|^2/2\epsilon)}{\sum_j \pi_j \exp(-\|\mathrm{x}_n - \mu_j\|^2/2\epsilon)} \tag{6}$$

- Suppose $j = \arg\min_k \|\mathrm{x}_n - \mu_k\|^2$, if $\epsilon \to 0$ and $\pi_k \neq 0$ for all k, then

$$\gamma(z_{nk}) \quad \to \quad \begin{cases} 1 & \text{if } j = k \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

Comparison of K-means and GMM

- (7) implies that $\gamma(z_{nk}) \to r_{nk}$ when $\epsilon \to 0$
- For parameter $\{\mu_k\}$, the formula in EM is

$$\mu_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) x_n \to \frac{\sum_n r_{nk} x_n}{\sum_n r_{nk}} \tag{8}$$

- For parameter $\{\pi_k\}$, the formula in EM is

$$\pi_k = \frac{N_k}{N} \to \frac{\sum_{n=1}^{N} r_{nk}}{N} \tag{9}$$

- It means that the limit of the EM algorithm for this particular Gaussian mixture is the exact K-means

Mixture Models in General

Mixture Models

- Let $P_k(\mathrm{x};\theta_k)$ be a density for $k = 1, \ldots, K$, where θ_k are the parameters specifying P_k. We will refer to each P_k as a bump.
- In the GMM setting, all the P_k are Gaussians, and $\theta_k = \{\boldsymbol{\mu}_k, \Sigma_k\}$
- A mixture model is a weighted sum of these K bumps,

$$P(\mathrm{x};\Theta) = \sum_{k=1}^{K} w_k P_k(\mathrm{x};\theta_k),$$

  where the weights satisfy $w_k \geq 0$ and $\sum_{k=1}^{K} w_k = 1$ and we have collected all the parameters into a single grand parameter, $\Theta = \{w_1, \ldots, w_K; \theta_1, \ldots, \theta_K\}$
- Intuitively, to generate a random point x, you first pick a bump according to the probabilities $w_1, \ldots, w_K$. Suppose you pick bump k. You then generate a random point from the bump density P_k

Parameter Estimation

- Given data $X = \mathrm{x}_1, \ldots, \mathrm{x}_N$ generated independently, we wish to estimate the parameters of the mixture which maximize the log-likelihood,

$$\begin{aligned}
\ln P(X \mid \Theta) &= \ln \prod_{n=1}^{N} P\left(\mathrm{x}_n \mid \Theta\right) \\
&= \ln \prod_{n=1}^{N} \left(\sum_{k=1}^{K} w_k P_k\left(\mathrm{x}_n; \theta_k\right) \right) \\
&= \sum_{n=1}^{N} \ln \left(\sum_{k=1}^{K} w_k P\left(\mathrm{x}_n \mid \theta_k\right) \right)
\end{aligned}$$

- Note that X is known and fixed. What is not known is which particular bump was used to generate data point x_n
- Denote by $j_n \in \{1, \ldots, K\}$ the bump that generated x_n (we say x_n is a 'member' of bump j_n). Collect all bump memberships into a set $J = \{j_1, \ldots, j_N\}$.

Complete and Incomplete Data

- If we knew which data belonged to which bump, we can estimate each bump density's parameters separately, using only the data belonging to that bump.
- We call (X, J) the complete data. If we know the complete data, we can easily optimize the log-likelihood.
- We call X the incomplete data. Though X is all we can measure, it is still called the 'incomplete' data because it does not contain enough information to easily determine the optimal parameters Θ^* which minimize $E_{\text{in}}(\Theta)$.

Likelihood of Complete Data

- To get the likelihood of the complete data, we need the joint probability $\mathbb{P}[\mathrm{x}_n, j_n \mid \Theta]$. Using Bayes' theorem,

$$\begin{aligned}\mathbb{P}[\mathrm{x}_n, j_n \mid \Theta] &= \mathbb{P}[j_n \mid \Theta]\,\mathbb{P}[\mathrm{x}_n \mid j_n, \Theta] \\ &= w_{j_n} P_{j_n}(\mathrm{x}_n; \theta_{j_n}).\end{aligned}$$

- Since the data are independent,

$$\begin{aligned}P(X, J \mid \Theta) &= \prod_{n=1}^{N} \mathbb{P}[\mathrm{x}_n, j_n \mid \Theta] \\ &= \prod_{n=1}^{N} w_{j_n} P_{j_n}(\mathrm{x}_n; \theta_{j_n}).\end{aligned}$$

Likelihood of Complete Data

- Let N_k be the number of occurrences of bump k in J, and let X_k be those data points corresponding to the bump k, so $X_k = \{\mathrm{x}_n \in X : j_n = k\}$. We compute the log-likelihood for the complete data as follows:

$$
\begin{aligned}
\ln P(X, J \mid \Theta) &= \sum_{n=1}^{N} \ln w_{j_n} + \sum_{n=1}^{N} \ln P_{j_n}\left(\mathrm{x}_n; \theta_{j_n}\right) \\
&= \sum_{k=1}^{K} N_k \ln w_k + \sum_{k=1}^{K} \underbrace{\sum_{\mathrm{x}_n \in X_k} \ln P_k\left(\mathrm{x}_n; \theta_k\right)}_{L_k(X_k, \theta_k)} \\
&= \sum_{k=1}^{K} N_k \ln w_k + \sum_{k=1}^{K} L_k\left(X_k; \theta_k\right).
\end{aligned}
$$

Likelihood of Complete Data

- The w_k (in the first term) are separated from the θ_k (in the second term)
- The second term is the sum of K non-interacting log-likelihoods $L_k(X_k, \theta_k)$ corresponding to the data belonging to X_k and only involving bump k 's parameters θ_k.
- Each log-likelihood L_k can be optimized independently of the others.
- For many choices of P_k, $L_k(X_k; \theta_k)$ can be optimized analytically, even though the log-likelihood for the incomplete data is intractable

Results

- Maximize the first term in complete data log likelihood subject to $\sum_k w_k = 1$, and the optimal weights are shown to be

$$w_k^* = N_k / N$$

- For the GMM,

$$P_k\left(\mathrm{x}; \boldsymbol{\mu}_k, \Sigma_k\right) = \frac{1}{(2\pi)^{d/2} \left|\Sigma_k\right|^{1/2}} \exp\left(-\frac{1}{2}\left(\mathrm{x} - \boldsymbol{\mu}_k\right)^{\mathrm{T}} \Sigma_k^{-1} \left(\mathrm{x} - \boldsymbol{\mu}_k\right)\right).$$

Maximizing $L_k\left(X_k; \boldsymbol{\mu}_k, \Sigma_k\right)$ gives the optimal parameters:

$$\boldsymbol{\mu}_k^* = \frac{1}{N_k} \sum_{\mathrm{x}_n \in X_k} \mathrm{x}_n;$$

$$\Sigma_k^* = \frac{1}{N_k} \sum_{\mathrm{x}_n \in X_k} \left(\mathrm{x}_n - \boldsymbol{\mu}_k\right)\left(\mathrm{x}_n - \boldsymbol{\mu}_k\right)^{\mathrm{T}}.$$

- $\boldsymbol{\mu}_k^*$ is the insample mean for the data belonging to bump k
- Σ_k^* is the in-sample covariance matrix.

When J is not observed

- In reality, we do not have access to J, and hence it is called a 'hidden variable'
- One approach is to guess J and maximize the resulting complete likelihood.
- This almost works. Instead of maximizing the complete likelihood for a single guess, we consider an average of the complete likelihood over all possible guesses.
- Specifically, we treat J as an unknown random variable and maximize the expected value (with respect to J) of the complete log-likelihood.
- This expected value is as easy to minimize as the complete likelihood.

Example

- You have two opaque bags.
  - Bag 1 has red and green balls, with μ_1 being the fraction of red balls.
  - Bag 2 has red and blue balls with μ_2 being the fraction of red.
- You pick four balls in independent trials as follows.
  - First pick one of the bags at random, each with probability $\frac{1}{2}$;
  - then, pick a ball at random from the bag.
- Here is the sample of four balls you got: 
- The task is to estimate μ_1 and μ_2.
- It would be much easier if we knew which bag each ball came from.

Let's Try

- Half the balls will come from Bag 1 and the other half from Bag 2.
- The blue balls come from Bag 2, so the other two should come from Bag 1: 🟢🔴 | 🔵🔵
- Using in-sample estimates, $\hat{\mu}_1 = \frac{1}{2}$ and $\hat{\mu}_2 = 0$.

Alternative Reasoning

- It seems a little counter-intuitive that we would estimate $\hat{\mu}_2 = 0$, for isn't there a positive probability that the red ball came from Bag 2?
- Here is another way to reason. 'Half' of each red ball came from Bag 1 and the other 'half' from Bag 2.
- So,

$$\begin{aligned} \hat{\mu}_1 &= \frac{1}{2} / \left(1 + \frac{1}{2}\right) = \frac{1}{3} \\ \hat{\mu}_2 &= \frac{1}{2} / \left(2 + \frac{1}{2}\right) = \frac{1}{5} \end{aligned}$$

A More Generalized Case

- A proportion p_1 of red ball came from Bag 1 and the other $p_2 = 1 - p_1$ from Bag 2.
- So,

$$\begin{aligned} \hat{\mu}_1 &= p_1 / (1 + p_1) \\ \hat{\mu}_2 &= p_2 / (2 + p_2) \end{aligned}$$

Formally

- The red ball is from either Bag 1 or Bag 2.
- We can compute the likelihood for each of these two cases:

$$\begin{aligned}
&\tfrac{1}{2}(1-\mu_1)\times\tfrac{1}{2}(\mu_1)\times\tfrac{1}{2}(1-\mu_2)\times\tfrac{1}{2}(1-\mu_2) \quad \text{(Bag 1)};\\
&\tfrac{1}{2}(1-\mu_1)\times\tfrac{1}{2}(\mu_2)\times\tfrac{1}{2}(1-\mu_2)\times\tfrac{1}{2}(1-\mu_2) \quad \text{(Bag 2)}.
\end{aligned}$$

- The log-likelihood for these two cases:

$$\begin{aligned}
\mathrm{LLH}_1 &= \ln(1-\mu_1)+\ln(\mu_1)+2\ln(1-\mu_2)-4\ln 2 \quad \text{(Bag 1)};\\
\mathrm{LLH}_2 &= \ln(1-\mu_1)+\ln(\mu_2)+2\ln(1-\mu_2)-4\ln 2 \quad \text{(Bag 2)}.
\end{aligned}$$

Expectation Step 2/2

- Compute the expected log-likelihood using p_1 and $p_2 = 1 - p_1$:

$$\begin{aligned} & p_1 \times \mathrm{LLH}_1 + p_2 \times \mathrm{LLH}_2 \\ = \quad & \ln(1-\mu_1) + p_1 \ln(\mu_1) + p_2 \ln(\mu_2) + 2\ln(1-\mu_2) - 4\ln 2. \end{aligned}$$

Maximization Step

- Next comes the maximization step. Treating p_1, p_2 as constants, maximize the expected log-likelihood with respect to μ_1, μ_2 and update $\hat{\mu}_1, \hat{\mu}_2$ to these optimal values as

$$\hat{\mu}_1 \leftarrow \frac{p_1}{1 + p_1} \quad \text{and} \quad \hat{\mu}_2 \leftarrow \frac{p_2}{2 + p_2}$$

Expectation Step 1/2

- We have new estimates $\hat{\mu}_1$ and $\hat{\mu}_2$.
- Using Bayes theorem, we can compute updated p_1 and p_2 as

$$p_1 = \frac{\hat{\mu}_1}{\hat{\mu}_1 + \hat{\mu}_2}, \quad p_2 = \frac{\hat{\mu}_2}{\hat{\mu}_1 + \hat{\mu}_2}.$$

Expectation Maximization Steps

$$p_1 = \frac{\hat{\mu}_1}{\hat{\mu}_1 + \hat{\mu}_2}, \quad p_2 = \frac{\hat{\mu}_2}{\hat{\mu}_1 + \hat{\mu}_2}.$$

- Altogether, we have

$$\hat{\mu}_1 \leftarrow \frac{p_1}{1 + p_1} = \frac{\hat{\mu}_1}{2\hat{\mu}_1 + \hat{\mu}_2} \quad \text{and} \quad \hat{\mu}_2 \leftarrow \frac{p_2}{2 + p_2} = \frac{\hat{\mu}_2}{2\hat{\mu}_1 + 3\hat{\mu}_2}.$$

Numerical Results

- The full algorithm just iterates this update process with the new estimates. Let's see what happens if we start (arbitrarily) with estimates $\hat{\mu}_1 = \hat{\mu}_2 = \frac{1}{2}$:

	Iteration number									
	0	1	2	3	4	5	6	7	...	1000
$\hat{\mu}_1$	$\frac{1}{2}$	$\frac{1}{3}$	0.38	0.41	0.43	0.45	0.45	0.46	...	0.49975
$\hat{\mu}_2$	$\frac{1}{2}$	$\frac{1}{5}$	0.16	0.13	0.10	0.09	0.07	0.07	...	0.0005

- If we continued this table, $\hat{\mu}_1 \to \frac{1}{2}$ and $\hat{\mu}_2 \to 0$.

Example Revisited

- You have two opaque bags.
  - Bag 1 has red and green balls, with μ_1 being the fraction of red balls.
  - Bag 2 has red and blue balls with μ_2 being the fraction of red.
- You pick four balls in independent trials as follows.
  - First pick one of the bags at random, each with probability $\frac{1}{2}$;
  - then, pick a ball at random from the bag.
- Here is the sample of four balls you got: 
- The likelihood of the data is

$$\frac{1}{2}(1-\mu_1) \times \frac{1}{2}(\mu_1+\mu_2) \times \frac{1}{2}(1-\mu_2) \times \frac{1}{2}(1-\mu_2).$$

- The log-likelihood is

$$\ln(1-\mu_1) + \ln(\mu_1+\mu_2) + 2\ln(1-\mu_2) - 4\ln 2.$$

- For this simple example, we can maximize the log-likelihood and obtain $\hat{\mu}_1 = \frac{1}{2}$ and $\hat{\mu}_2 = 0$

Discussions

- It's miraculous that by maximizing an expected log-likelihood using a guess for the parameters, we end up converging to the true maximum likelihood solution.
- Why is this useful? Because the maximizations for μ_1 and μ_2 are decoupled. We trade a maximization of a complicated likelihood of the incomplete data for a bunch of simpler maximizations that we iterate.

THANKS!