Fundamentals of Machine Learning:

Unsupervised Learning I: K-Means Clustering and Gaussian Mixture Models

Prof. Andrew D. Bagdanov (andrew.bagdanov AT unifi.it)

UNIVERSITÀ
DEGLI STUDI
FIRENZE

DINFO
DIPARTIMENTO DI
INGEGNERIA
DELL'INFORMAZIONE

Outline

Introduction

Motivations

- In the previous lecture we saw how kernel density estimators (or Parzen density estimation) can be used to infer a marginal data distribution.
- This is already an example of an unsupervised learning problem.
- We are given data without any targets and we should learn something "useful" from this data.
- In this lecture we will see two more standard approaches to unsupervised learning:
  1. The K-Means algorithm, which learns K clusters of data in Euclidean spaces.
  2. The Gaussian Mixture Model, which fits a generative parametric mixture of K Gaussians to the data.

Lecture objectives

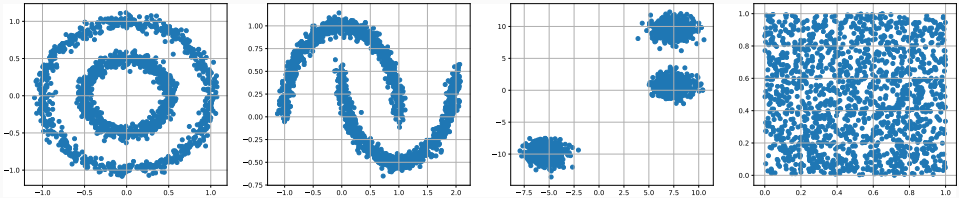
After this lecture you will:

- Understand how the K-Means algorithm partitions the input data by assigning each point to exactly one cluster.
- Understand how Gaussian Mixture Models represent the data likelihood as superposition of Gaussian distributions.
- Understand how latent variables can be entangled with observed variables to render tractable certain optimization problems.
- Understand how Expectation Maximization (EM) is used to learn the parameters of a GMM from data.

K-means Clustering

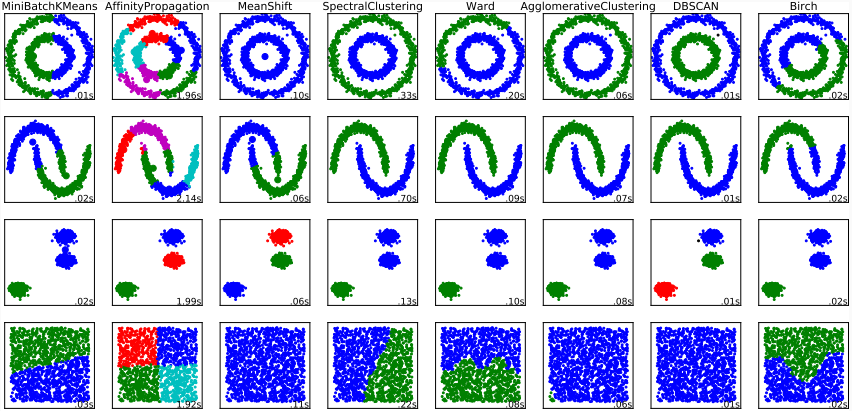
Clustering: Motivation

- What if you have data with structure, but you don't know what this structure is or have any a priori model for it?



Clustering: Motivation (continued)

- Clustering refers to techniques that learn groups of related data points in feature space:



Clustering: The k-Means Algorithm

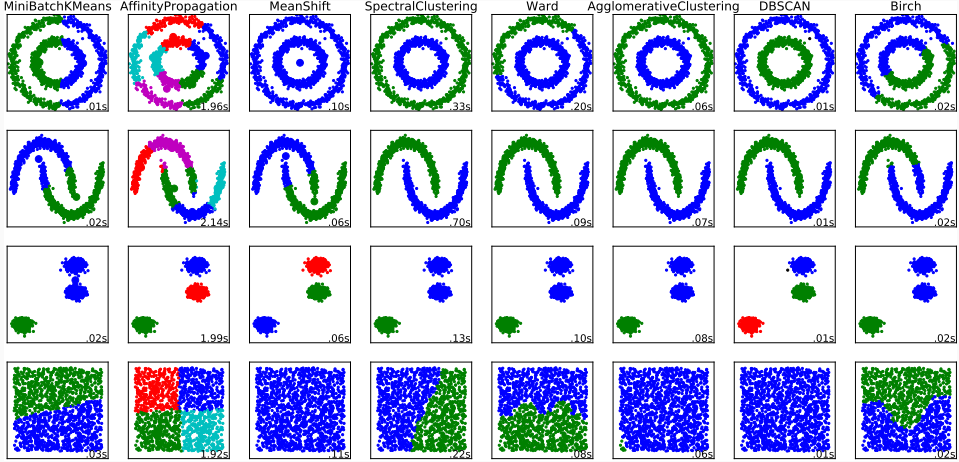
- k-Means is a very simple algorithm that associates each data point with one of k means or centroids in the data space:
- Given an initial set of k means $\mathbf{m}_1^1, \ldots, \mathbf{m}_k^1$, the alternates between two steps:
  1. Assignment: $S_i^t = \{ \mathbf{x}_p \mid ||\mathbf{x}_p - \mathbf{m}_i^t|| \leq ||\mathbf{x}_p - \mathbf{m}_j^t|| \forall j \}$.
  2. Update: $\mathbf{m}_i^{t+1} = \frac{1}{|S_i^t|} \sum_{\mathbf{x}_i \in S_i^t} \mathbf{x}_j$.
- The algorithm converges when the cluster assignments no longer change.
- Note: this algorithm is not guaranteed to find the global optimum clusters.

Clustering: Analysis

- Clustering is a robust technique (in that it never fails).
- Its usefulness depends on the data distribution and which type of clustering you perform.
- In sklearn:
  - `cluster.AffinityPropagation`
  - `cluster.AgglomerativeClustering`
  - `cluster.Birch`
  - `cluster.DBSCAN`
  - `cluster.KMeans`
  - `cluster.MeanShift`
  - `cluster.SpectralClustering`
- See `sklearn.cluster` for more details.

Clustering: Motivation (continued)

- It's useful to revisit this plot:



The K-Means Algorithm

- Assume we have a dataset $\mathcal{D} = \{ \mathbf{x}_1, \ldots, \mathbf{x}_N \}$ of N samples of a D-dimensional Euclidean variable $\mathbf{x}$.
- Goal: partition $\mathcal{D}$ into K clusters.
- What is a cluster? Intuitively, a subset of $\mathcal{D}$ is a cluster if the sum of inter-point distances in it are small.
- To capture this intuition, we introduce a set of K D-dimensional prototype vectors $\boldsymbol{\mu}_k$ for $k = 1, \ldots, K$.
- Then, for each sample $\mathbf{x}_n$ we introduce a binary indicator variable r_{nk}:

$$r_{nk} = \begin{cases} 1 & \text{if } \mathbf{x}_n \text{ is assigned to cluster } k \\ 0 & \text{otherwise} \end{cases}$$

The K-Means Algorithm

- Now we can define an objective function:

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} ||\mathbf{x}_n - \boldsymbol{\mu}_k||^2$$

- This results in an optimization problem in r_{nk} and $\boldsymbol{\mu}_k$.
- We can approach this with an alternating iterative algorithm which:
  1. (E-Step): Minimizes J with respect to r_{nk} keeping $\boldsymbol{\mu}_k$ fixed.
  2. (M-Step): Minimizes J with respect to $\boldsymbol{\mu}_k$ keeping r_{nk} fixed.

K-Means: The E-Step

- Consider holding the $\boldsymbol{\mu}_k$ fixed and finding the optimal r_{nk}.
- J is a linear function of r_{nk}, and we can optimize for each n independently.
- So our rule is to just assign $\mathbf{x}_n$ to the closest $\boldsymbol{\mu}_k$:

$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg\min_j ||\mathbf{x}_n - \boldsymbol{\mu}_j||^2 \\ 0 & \text{otherwise} \end{cases}$$

K-Means: The M-Step

- Now consider holding the r_{nk} fixed and minimizing J in $\boldsymbol{\mu}_k$.
- J is quadratic in $\boldsymbol{\mu}_k$, so we set the gradient wrt them to zero.

$$2\sum_{n=1}^{N} r_{nk}(\mathbf{x}_n - \boldsymbol{\mu}_k) = 0 \quad \forall k$$

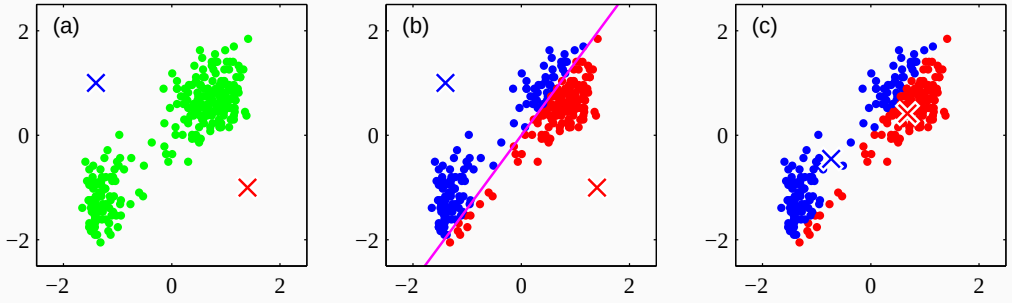
- Which is easy to solve for $\boldsymbol{\mu}_k$:

$$\boldsymbol{\mu}_k = \frac{\sum_{n=1}^{N} r_{nk}\mathbf{x}_n}{\sum_{n=1}^{N} r_{nk}}$$

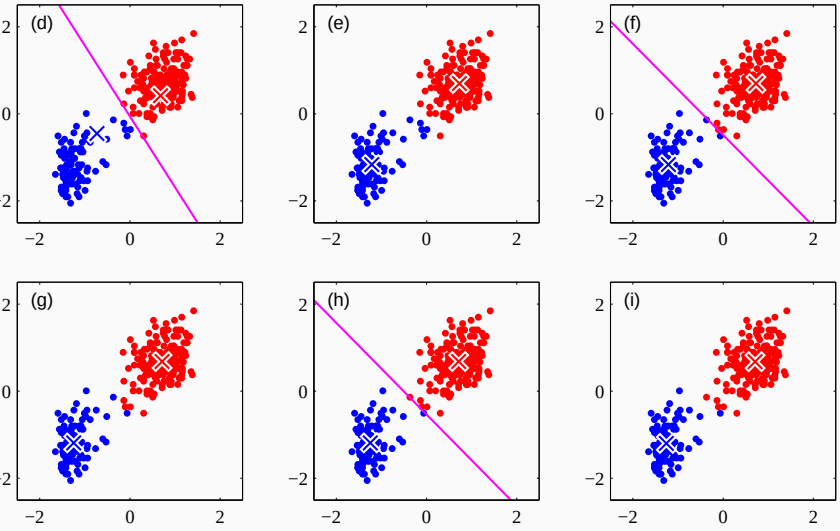
- And even easier to interpret.

K-Means: Analysis

- The two steps of the K-Means algorithm are alternated until the assignments do not change in the E-Step.
- Each step of K-Means reduces the value of J, convergence is guaranteed.
- Depending on initialization, however, K-Means can converge to a bad local minimum.



K-Means: Analysis

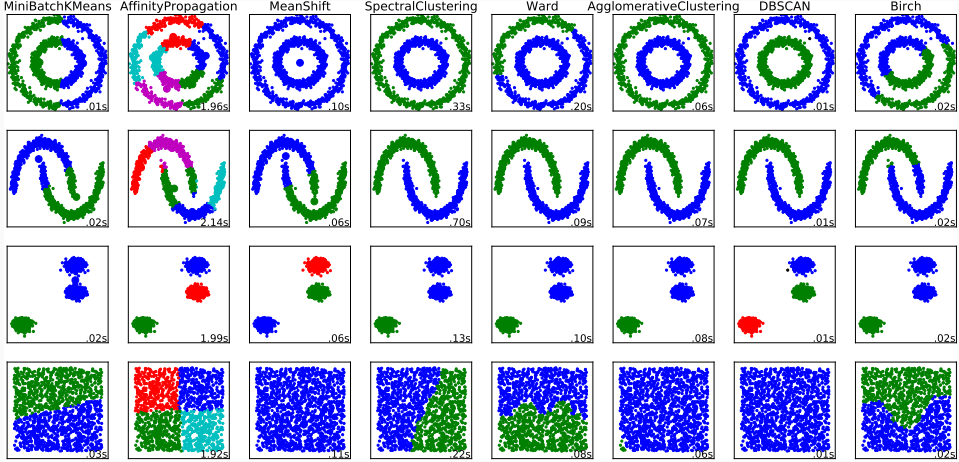


Clustering: Analysis

- Clustering is a robust technique (in that it never fails).
- Its usefulness depends on the data distribution and which type of clustering you perform.
- In sklearn:
  - `cluster.AffinityPropagation`
  - `cluster.AgglomerativeClustering`
  - `cluster.Birch`
  - `cluster.DBSCAN`
  - `cluster.KMeans`
  - `cluster.MeanShift`
  - `cluster.SpectralClustering`
- See `sklearn.cluster` for more details.

Clustering: Analysis

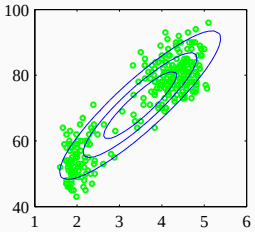
- It's useful to revisit this plot:

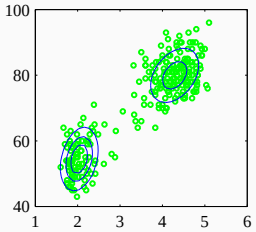


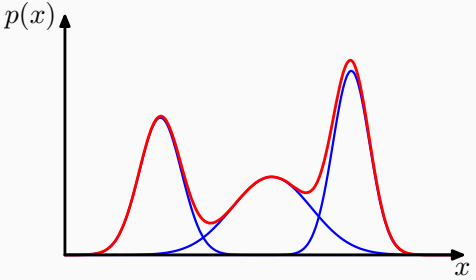
SKIPPABLE Gaussian Mixture Models

Gaussian Mixtures Models (GMMs) and Expectation Maximization (EM)

- Modeling distributions with a single Gaussian can work well in practice.
- However, sometimes it is a terrible approximation of reality.







GMMs: a Latent Variable Model

- We have already seen the kernel density estimation approach – slap a Gaussian around every point and normalize.
- In these examples, though, it seems clear that the data are generated from some finite and fixed number of generating distributions.
- We can instead fit a parametric Mixture of Gaussians – a Gaussian Mixture Model (GMM) to data:

$$p(\mathbf{x}; \boldsymbol{\theta}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- This is a superposition of K Gaussians, each of which is called a component of the mixture with mixing coefficient π_k.
- We require $0 \leq \pi_k \leq 1$ and $\sum_k \pi_k = 1$. Why?

GMMs: a Latent Variable Model

- OK, but if we try to use Maximum Likelihood to estimate $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$:

$$\begin{aligned} p(\mathcal{D}; \boldsymbol{\theta}) &= \prod_{n=1}^{N} p(\mathbf{x}_n \mid \boldsymbol{\theta}) \\ &= \prod_{n=1}^{N} \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \end{aligned}$$

- Taking the logarithm (which won't change the maximum):

$$\ln p(\mathcal{D}; \boldsymbol{\theta}) = \sum_{n=1}^{N} \ln \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}_k)$$

- And we run into a dead end... Why?

GMMs: a Latent Variable Model

- Let's look at the mixture equation again:

$$p(\mathbf{x}; \boldsymbol{\theta}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- This type of superposition is difficult to fit to data without extra information about which components are "responsible" for specific points.
- So, we introduce a K-dimensional discrete latent random variable $\mathbf{z}$ having a one-hot representation.
- A latent variable is one that is never actually observed, but that we introduce to simplify the optimization problem (eventually).

GMMs: a Latent Variable Model

- Define $\mathbf{z}$ to be a one-hot random variable of dimension K: $z_k \in \{0, 1\}$ and $\sum_k z_k = 1$.
- We will now entangle this variable with $\mathbf{x}$ by defining the joint distribution $p(\mathbf{x}, \mathbf{z})$ in terms of the marginal $p(\mathbf{z})$ and the conditional distributions $p(\mathbf{x} \mid \mathbf{z})$.
- The marginal distribution $p(\mathbf{z})$ is specified in terms of the mixing coefficients:

$$p(z_k = 1) = \pi_k \quad p(\mathbf{z}) = \prod_{k=1}^{K} \pi_k^{z_k}$$

- The conditional distribution of $\mathbf{x}$ given a specific $\mathbf{z}$ is similarly:

$$p(\mathbf{x} \mid \mathbf{z}) = \prod_{k=1}^{K} \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)^{z_k}$$

GMMs: a Latent Variable Model

- The joint distribution over $\mathbf{x}$ and $\mathbf{z}$ is $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{z})p(\mathbf{x} \mid \mathbf{z})$.
- If we marginalize $\mathbf{z}$ away, we get our mixture back:

$$\begin{aligned} p(\mathbf{x}) &= \sum_{\mathbf{z}} p(\mathbf{z})p(\mathbf{x} \mid \mathbf{z}) \\ &= \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \end{aligned}$$

- One might rightly ask: so what?
- The point is that this allows us to work with the joint distribution instead of the unstructured marginal.
- Because of this representation of $p(\mathbf{x})$, if we have N observations $\mathbf{x}_1, \ldots, \mathbf{x}_N$, we will have a latent variable $\mathbf{z}_n$ for each $\mathbf{x}_n$.

GMMs: a Latent Variable Model

- This is an important quantity in what follows:

$$\begin{aligned}\gamma(z_k) \equiv p(z_k = 1 \mid \mathbf{x}) &= \frac{p(z_k = 1)p(\mathbf{x} \mid z_k = 1)}{\sum_{k=1}^{K} p(z_k = 1)p(\mathbf{x} \mid z_k = 1)} \\ &= \frac{\pi_k \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}\end{aligned}$$

- π_k is the prior probability of $z_k = 1$, and $\gamma(z_k)$ is the corresponding posterior.
- We can think of $\gamma(z_k)$ as the responsibility component k takes for explaining observation $\mathbf{x}$.

GMMs: Fitting the Model

- Again, we have a dataset $\mathcal{D} = \{ \mathbf{x}_1, \ldots, \mathbf{x}_N \}$ that we want to fit a GMM to.
- We can represent $\mathcal{D}$ as an $N \times D$ matrix $\mathbf{X}$.
- The corresponding latent variables will be an $N \times K$ matrix $\mathbf{Z}$.
- Assuming the $\mathbf{x}_n$ are i.i.d. we can "just" optimize:

$$\ln p(\mathbf{X} \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

- We know how to do this: take derivatives wrt $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$, and $\boldsymbol{\pi}$ and set them to $\mathbf{0}$.

GMMs: Fitting the Model

- Our data likelihood:

$$\ln p(\mathbf{X} \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

- The gradient wrt $\boldsymbol{\mu}_k$:

$$0 = \sum_{n=1}^{N} \underbrace{\frac{\pi_k \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}}_{=\gamma(z_{nk})} \Sigma_k^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_k)$$

- Defining $N_k = \sum_{n=1}^{N} \gamma(z_{nk})$ (how to interpret this?):

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n$$

GMMs: Fitting the Model

- By a similar (but longer) line of reasoning:

$$\boldsymbol{\Sigma}_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})(\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^T$$

- For the mixing coefficients we must constrain them to sum to 1:

$$\ln p(\mathbf{X} \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) + \lambda \left(\sum_{k=1}^{K} \pi_k - 1 \right)$$

- The partials wrt π_k are:

$$\begin{aligned} 0 &= \sum_{n=1}^{N} \frac{\mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_j \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} + \lambda \\ &= \frac{1}{\pi_k} \sum_{n=1}^{N} \frac{\pi_k \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_j \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} + \lambda \end{aligned}$$

GMMs: Fitting the Model

- Again, our responsibilities appear (after multiplying by 1!):

$$\begin{aligned} 0 &= \frac{1}{\pi_k} \sum_{n=1}^{N} \frac{\pi_k \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_j \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} + \lambda \\ &= \frac{N_k}{\pi_k} + \lambda \end{aligned}$$

- The partials wrt our Lagrange multiplier:

$$0 = \sum_{k=1}^{k} \pi_k - 1 \implies \sum_{k=1}^{K} \pi_k = 1 \quad \text{(duh)}$$

- Combining, we have:

$$\pi_k = \frac{N_k}{N}$$

Expectation Maximization (EM) for GMMs

1. Initialize $\boldsymbol{\pi}$, $\boldsymbol{\mu}$, and $\boldsymbol{\Sigma}$.
2. E step: Compute responsibilities for current parameters.

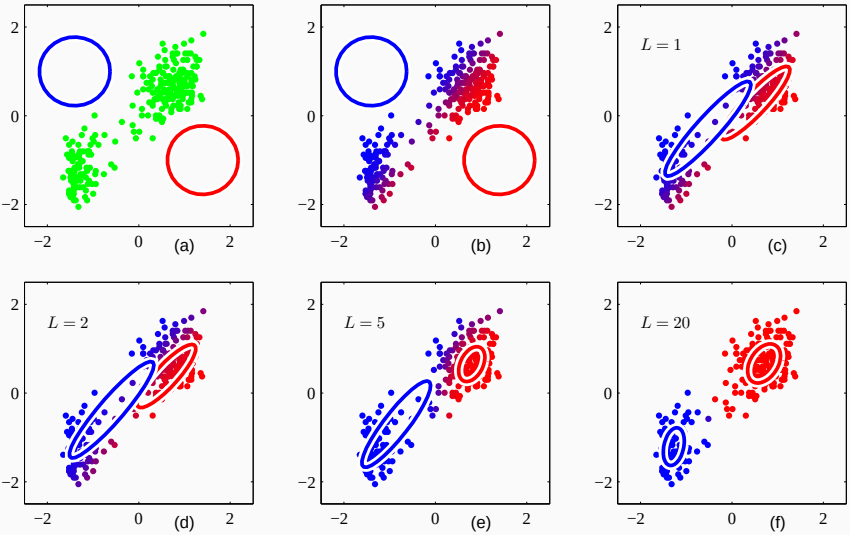
$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}, \quad N_k = \sum_{n=1}^{N} \gamma(z_{nk})$$

3. M step: Re-estimate parameters with current $\gamma(z_{nk})$ (to maximize likelihood).

$$\begin{aligned}
\boldsymbol{\mu}_k^{\text{new}} &= \frac{1}{N_K} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n \\
\boldsymbol{\Sigma}_k^{\text{new}} &= \frac{1}{N_K} \sum_{n=1}^{N} (\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}})(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}})^T \\
\pi_k^{\text{new}} &= \frac{N_k}{N}
\end{aligned}$$

4. Check for convergence and (maybe) repeat.

Visualizing EM



Concluding Remarks

Clustering

- Clustering in general – and K-Means in particular – are robust methods for unsupervised learning.
- Clustering is used in a broad variety of applications.
- It is also useful to understand what, if any, structure exists in your data.
- There are many types of clustering algorithms, each of which has its own affordances.

Gaussian Mixtures and the EM Algorithm

- Gaussian Mixture Models can be thought of as a type of soft K-Means.
- In this lecture we have seen how introducing latent variables can (eventually) simplify optimization of otherwise intractable objectives.
- GMMs also have a natural generative interpretation:
  1. Sample a mixture component k according to $p(\pi)$.
  2. Then sample a point from $\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$.
- Note that the Expectation Maximization (EM) algorithm is broadly applicable to estimation problems of joint distributions over observed and latent variables.
- Here we have only seen one such application of EM.

Reading and Homework Assignments

Reading Assignment:

- Bishop: Chapter 2 (2.3.9), Chapter 9 (9.1, 9.2)
- Scikit-learn User Guide (sections 2.1, 2.3)