CSCI-567: Machine Learning (Fall 2019)

Prof. Victor Adamchik

U of Southern California

Nov. 5, 2019

Top 10 Algorithms in Machine Learning ...

(circa 2019):

- k-Nearest Neighbors
- Decision Tree and Random Forests
- Naive Bayes
- Linear and Logistic Regression
- Artificial Neural Networks
- SVM
- Boosting
- Dimensionality Reduction Algorithms
- Clustering
- Markov Chains and Decision Processes

Outline

Outline

Supervised learning v.s unsupervised learning

There are different types of machine learning problems

- **supervised learning** (what we have discussed by now)
  All data is labeled
  Aim to predict, e.g. classification and regression

- **unsupervised learning**
  All data is unlabeled
  Aim to discover hidden and latent patterns and explore data

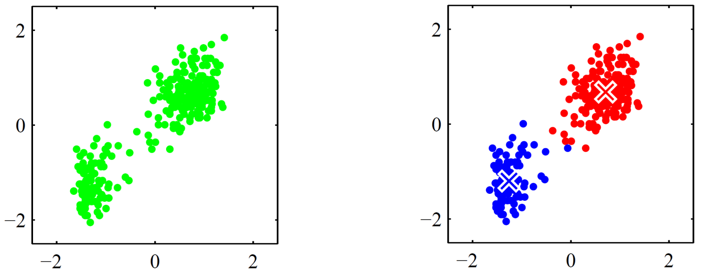
Today's focus: one important unsupervised learning problem: **clustering**

Clustering: informal definition

Given: a set of data points (feature vectors), *without labels*

Output: group the data into some clusters, which means

- assign each point to a specific cluster
- find the center (representative/prototype/...) of each cluster

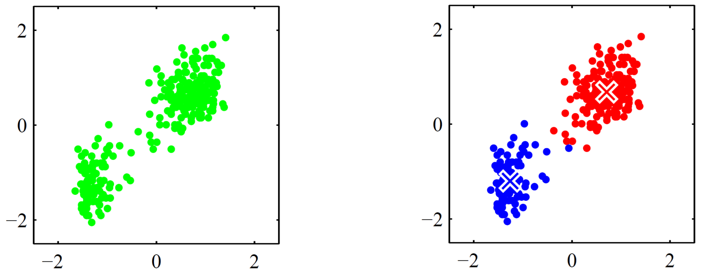


Clustering: formal definition

Given: data points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N \in \mathbb{R}^{\mathsf{D}}$ and #clusters K we want to find

Output: group the data into K clusters, which means

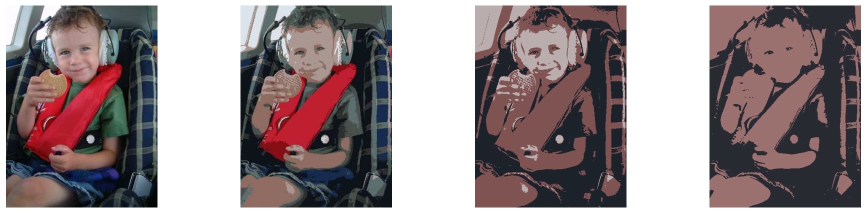
- find an assignment $\gamma_{nk} \in \{0, 1\}$ s.t. if a data point $n \in [N]$ belongs to a cluster $k \in [K]$ then $\gamma_{nk} = 1$ and $\sum_{k \in [K]} \gamma_{nk} = 1$.
- find the cluster centers $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K \in \mathbb{R}^{\mathsf{D}}$



Many applications

One example: **image compression** (vector quantization)

- each pixel is a point
- perform clustering over these points
- replace each point by the center of the cluster it belongs to



Original image Large $K \longrightarrow$ Small K

Formal Objective

Key difference from supervised learning problems: no labels given, which means *no ground-truth to even measure the quality of your answer!*

Still, we can turn it into an optimization problem, e.g. through the popular **"K-means" objective**: find γ_{nk} and $\boldsymbol{\mu}_k$ to minimize

$$F(\{\gamma_{nk}\}, \{\boldsymbol{\mu}_k\}) = \sum_{n=1}^{N}\sum_{k=1}^{K} \gamma_{nk}\|\boldsymbol{x}_n - \boldsymbol{\mu}_k\|_2^2$$

i.e. the **sum of distances of each point to its center**.

Unfortunately, finding the exact minimizer is *NP-hard!*

Alternating minimization

Instead, use a heuristic that alternatively minimizes over $\{\gamma_{nk}\}$ and $\{\boldsymbol{\mu}_k\}$:

Initialize $\{\gamma_{nk}^{(1)}\}$ and $\{\boldsymbol{\mu}_k^{(1)}\}$

For $t = 1, 2, \ldots$

- fix centers $\{\boldsymbol{\mu}_k^{(t)}\}$, find assignments $\{\gamma_{nk}^{(t+1)}\}$

$$\{\gamma_{nk}^{(t+1)}\} = \operatorname*{argmin}_{\{\gamma_{nk}\}} F\left(\{\gamma_{nk}\}, \{\boldsymbol{\mu}_k^{(t)}\}\right)$$

- fix assignments$\{\gamma_{nk}^{(t+1)}\}$, find new centers $\{\boldsymbol{\mu}_k^{(t+1)}\}$

$$\{\boldsymbol{\mu}_k^{(t+1)}\} = \operatorname*{argmin}_{\{\boldsymbol{\mu}_k\}} F\left(\{\gamma_{nk}^{(t+1)}\}, \{\boldsymbol{\mu}_k\}\right)$$

A closer look

The first step (fixed centers, find assignments)

$$\operatorname*{argmin}_{\{\gamma_{nk}\}} F(\{\gamma_{nk}\}, \{\boldsymbol{\mu}_k\}) = \operatorname*{argmin}_{\{\gamma_{nk}\}} \sum_n \sum_k \gamma_{nk}\|\boldsymbol{x}_n - \boldsymbol{\mu}_k\|_2^2$$

is simply to **assign each $\boldsymbol{x}_n$ to the closest $\boldsymbol{\mu}_k$**, i.e.

$$\gamma_{nk} = \mathbb{I}\left[k == \operatorname*{argmin}_{c} \|\boldsymbol{x}_n - \boldsymbol{\mu}_c\|_2^2\right]$$

for all $k \in [K]$ and $n \in [N]$.

A closer look

The second step (fixed assignments, find centers)

$$\operatorname*{argmin}_{\{\boldsymbol{\mu}_k\}} F(\{\gamma_{nk}\}, \{\boldsymbol{\mu}_k\}) = \operatorname*{argmin}_{\{\gamma_{nk}\}} \sum_n \sum_k \gamma_{nk}\|\boldsymbol{x}_n - \boldsymbol{\mu}_k\|_2^2$$

We will do it for each cluster.

The center is simply **an average of the points in that cluster** (hence the name)

$$\boldsymbol{\mu}_k = \frac{\sum_n \gamma_{nk}\boldsymbol{x}_n}{\sum_n \gamma_{nk}}$$

for each $k \in [K]$.

The K-means algorithm, S. Lloyd (1957)

Step 0 Initialization (choose K centers)

Step 1 Fix the centers $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$, assign each point to the closest center:

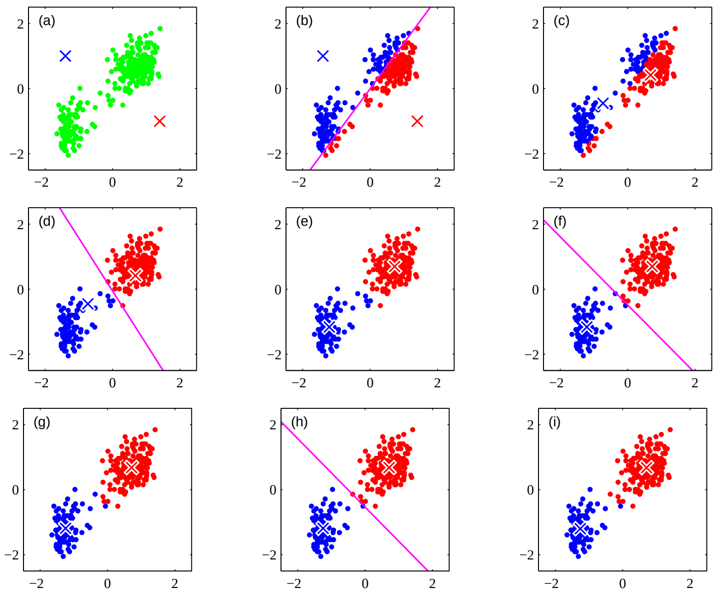
$$\gamma_{nk} = \mathbb{I}\left[k == \underset{c}{\operatorname{argmin}} \|\boldsymbol{x}_n - \boldsymbol{\mu}_c\|_2^2\right]$$

Step 2 Fix the assignment $\{\gamma_{nk}\}$, update the centers

$$\boldsymbol{\mu}_k = \frac{\sum_n \gamma_{nk}\boldsymbol{x}_n}{\sum_n \gamma_{nk}}$$

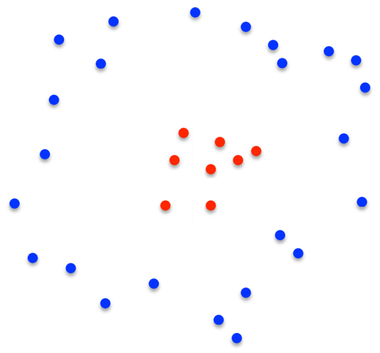
Step 3 Repeat Steps 1 and 2 until the centers no longer change.

An example



K-means algorithm is a heuristic!

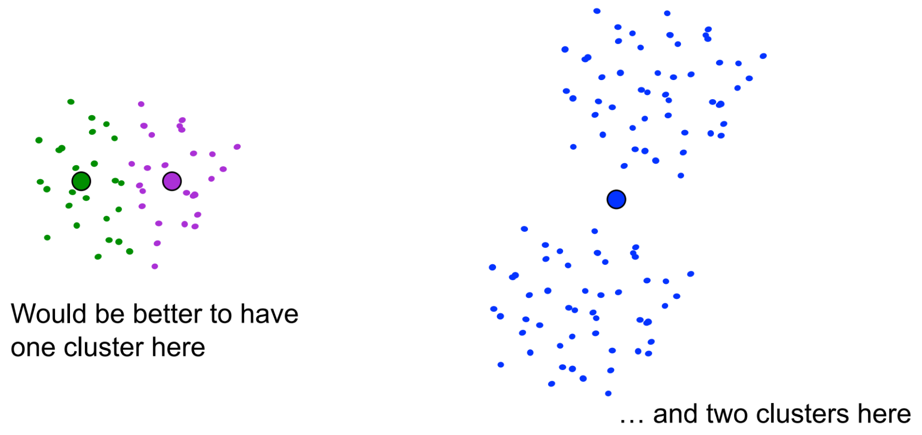
K-means is not always able to properly cluster:



K-means algorithm is a heuristic!

It does matter how you initialize the centers!

In the following example $K = 3$:



How to initialize?

A bad selection for the initial centers can lead to a very poor clustering of data.

It also may lead a very long to converge.

There are different ways to initialize:

- randomly pick K points as initial centers
- as it turns out, good initial centers are ones that aren't close to each other. (e.g. K-means++, 2007)

How to initialize?

The K-means++ algorithm.

The algorithm selects initial centers that aren't close to each other, then uses K-means algorithm for clustering.

The high-level pseudo-code for the K-means++:

- select a data point at random as the first center
- loop K-1 times
  - compute distance squared $d(x)^2$ from each point to the nearest cluster center
  - select a point that has largest probability $\frac{d(x)^2}{\sum_x d(x)^2}$ as the next center

Convergence

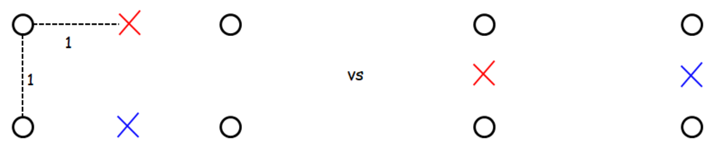
It will converge in a finite number of iterations to a local minimum.

- objective decreases at each step
- objective is lower bounded by 0
- #possible_assignments is finite (K^N, exponentially large though)
- it may take *exponentially* many iterations to converge
- it might not converge to the *global* minimum

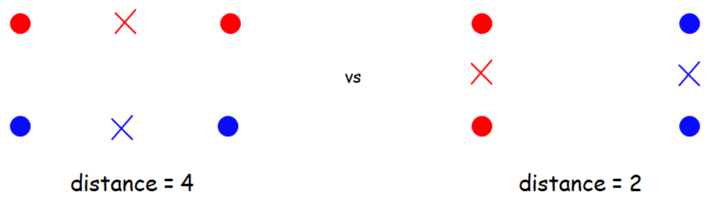
Local minimum v.s global minimum

Simple example: 4 data points, 2 clusters, 2 different initializations.

We initialize the centers by the mean of two points.



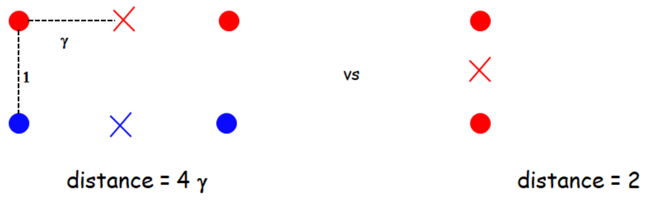
K-means converges immediately in both cases.



Local minimum v.s global minimum

In the left picture we get a local minimum, but in the right - a global minimum!

Moreover, local minimum can be *arbitrarily worse* if we increase the width of this "rectangle" to 2γ.



So, we get stuck at a local minimum.
Initialization matters a lot!

Cluster Quality Measures

We need to define a measure of cluster quality Q and then try different values of K until we get an optimal value for Q

There are different metrics for evaluating clustering algorithms, depending on what types of clusters we want

K-means emphasizes similarity of data within clusters:

$$Q = \sum_{k=1}^{K} \frac{1}{C_k} \sum_{\boldsymbol{x} \in C_k} \|\boldsymbol{x} - \boldsymbol{\mu}_k\|_2^2$$

where C_k is the number of data points in cluster k.

Cluster Quality Measures

Other Quality measures:

The aim is to identify sets of clusters that are compact and at the same time are well separated

- Dunn Index
- Davies-Bouldin Index
- Silhouette Index

Outline

Taxonomy of ML Models

There are two kinds of classification models in machine learning — generative models and discriminative models.

Discriminative models:

- nearest neighbor, k-means clustering, traditional neural networks, SVM.
- we learn $f()$ on data set $(\boldsymbol{x}_i, y_i)$ to output the most likely y on unseen x.
- having $f()$ we know how to discriminate unseen x's from different classes.
- we learn the decision boundary between the classes.
- we have no idea how the data is generated.

Taxonomy of ML Models

There are two kinds of classification models in machine learning — generative models and discriminative models.

Generative models:

- Naïve Bayes, Gaussian mixture model, Hidden Markov model, Adversarial Network (GAN).
- it's used widely in unsupervised machine learning.
- it's a probabilistic way to think about how the data might have been generated.
- learn the joint probability distribution $P(\boldsymbol{x}, y)$ and predict $P(y|\boldsymbol{x})$ with the help of Bayes Theorem.

Gaussian mixture models

Gaussian mixture models (GMM) is a probabilistic approach for clustering

- more explanatory than minimizing the K-means objective
- can be seen as a soft version of K-means

To solve GMM, we will introduce a powerful method for learning probabilistic model: **Expectation–Maximization (EM) algorithm**

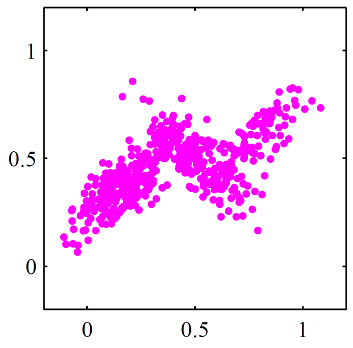
A generative model

For classification, we discussed the sigmoid model to "explain" how the labels are generated.

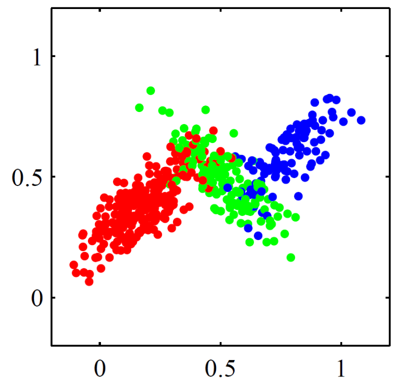
Similarly, for clustering, we want to come up with a probabilistic model p to **"explain" how the data is generated**.

That is, each point is an independent sample of $\boldsymbol{x} \sim p$.

What probabilistic model generates data like this?



Gaussian mixture models: intuition



We will model each region with a Gaussian distribution. This leads to the idea of Gaussian **mixture** models (GMMs).

The problem we are now facing is that i) we do not know which (color) region a data point comes from; ii) the parameters of Gaussian distributions in each region. We need to find all of them from *unsupervised* data $\mathcal{D} = \{\boldsymbol{x}_n\}_{n=1}^N$.

GMM: formal definition

A GMM has the following density function:

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \omega_k N(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \sum_{k=1}^{K} \omega_k \frac{1}{\sqrt{(2\pi)^D |\boldsymbol{\Sigma}_k|}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^{\mathrm{T}}\boldsymbol{\Sigma}_k^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k)}$$

where

- K: the number of Gaussian components (same as #clusters we want)
- $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$: mean and covariance matrix of the k-th Gaussian
- $\omega_1, \ldots, \omega_K$: mixture weights, they represent how much each component contributes to the final distribution. It satisfies two properties:

$$\forall\ k,\ \omega_k > 0, \quad \text{and} \quad \sum_k \omega_k = 1$$

Another view

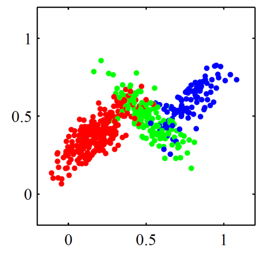
By introducing a **latent variable** $z \in [K]$, which indicates cluster membership, we can see p as a **marginal distribution**

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} p(\boldsymbol{x}, z = k) = \sum_{k=1}^{K} p(z = k)p(\boldsymbol{x}|z = k) = \sum_{k=1}^{K} \omega_k N(\boldsymbol{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

$\boldsymbol{x}$ and z are both random variables drawn from the model

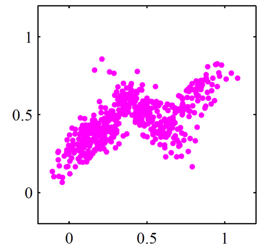
- $\boldsymbol{x}$ is observed
- z is unobserved/latent

An example



The conditional distributions are

$$p(\boldsymbol{x} \mid z = \text{red}) = N(\boldsymbol{x} \mid \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$$
$$p(\boldsymbol{x} \mid z = \text{blue}) = N(\boldsymbol{x} \mid \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$$
$$p(\boldsymbol{x} \mid z = \text{green}) = N(\boldsymbol{x} \mid \boldsymbol{\mu}_3, \boldsymbol{\Sigma}_3)$$



The marginal distribution is

$$p(\boldsymbol{x}) = p(\text{red})N(\boldsymbol{x} \mid \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) + p(\text{blue})N(\boldsymbol{x} \mid \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2) + p(\text{green})N(\boldsymbol{x} \mid \boldsymbol{\mu}_3, \boldsymbol{\Sigma}_3)$$

Learning GMMs

Learning a GMM means finding all the parameters $\boldsymbol{\theta} = \{\omega_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$.

In the process, we will learn the latent variable z_n as well:

$$p(z_n = k \mid \boldsymbol{x}_n) \triangleq \gamma_{nk} \in [0, 1]$$

i.e. "soft assignment" of each point to each cluster, as opposed to "hard assignment" by K-means.

GMM is more explanatory than K-means

- both learn the cluster centers $\boldsymbol{\mu}_k$'s
- in addition, GMM learns cluster weight ω_k and covariance $\boldsymbol{\Sigma}_k$, thus
  - we can *predict probability of seeing a new point*
  - we can *generate synthetic data*

How to learn these parameters?

An obvious attempt is **maximum-likelihood estimation (MLE)**: find

$$\underset{\boldsymbol{\theta}}{\operatorname{argmax}} \ \ln \prod_{n=1}^{N} p(\boldsymbol{x}_n \,; \boldsymbol{\theta}) = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \sum_{n=1}^{N} \ln p(\boldsymbol{x}_n \,; \boldsymbol{\theta}) \triangleq \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \, P(\boldsymbol{\theta})$$

This is called incomplete likelihood (since z_n's are unobserved), and is *intractable in general* (non-concave problem).

One solution is to still apply GD/SGD, but a much more effective approach is the **Expectation–Maximization (EM) algorithm**.

Preview of EM for learning GMMs

Step 0 Initialize $\omega_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$ for each $k \in [K]$

Step 1 (E-Step) **update the "soft assignment"** (fixing parameters)

$$\gamma_{nk} = p(z_n = k \mid \boldsymbol{x}_n) \propto \omega_k N\left(\boldsymbol{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)$$

Step 2 (M-Step) **update the model parameter** (fixing assignments)

$$\omega_k = \frac{\sum_n \gamma_{nk}}{N} \qquad \boldsymbol{\mu}_k = \frac{\sum_n \gamma_{nk} \boldsymbol{x}_n}{\sum_n \gamma_{nk}}$$

$$\boldsymbol{\Sigma}_k = \frac{1}{\sum_n \gamma_{nk}} \sum_n \gamma_{nk} (\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^{\mathrm{T}}$$

Step 3 return to Step 1 if not converged

Demo

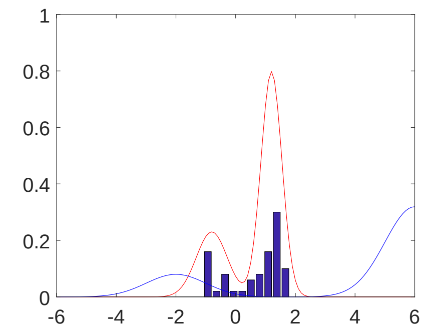
Generate 50 data points from a mixture of 2 Gaussians with

- $\omega_1 = 0.3, \mu_1 = -0.8, \Sigma_1 = 0.52$
- $\omega_2 = 0.7, \mu_2 = 1.2, \Sigma_2 = 0.35$

histogram represents the data

red curve represents the ground-truth density $p(\boldsymbol{x}) = \sum_{k=1}^K \omega_k N(\boldsymbol{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$

blue curve represents the learned density for a specific round



EM_demo.pdf shows how the blue curve moves towards red curve quickly via EM

EM algorithm

In general EM is **a heuristic to solve MLE with latent variables** (not just GMM), i.e. find the maximizer of

$$P(\boldsymbol{\theta}) = \sum_{n=1}^{N} \ln p(\boldsymbol{x}_n \,; \boldsymbol{\theta})$$

- $\boldsymbol{\theta}$ is the parameters for a general probabilistic model
- $\boldsymbol{x}_n$'s are observed random variables
- z_n's are latent variables

Again, directly solving the objective is intractable.

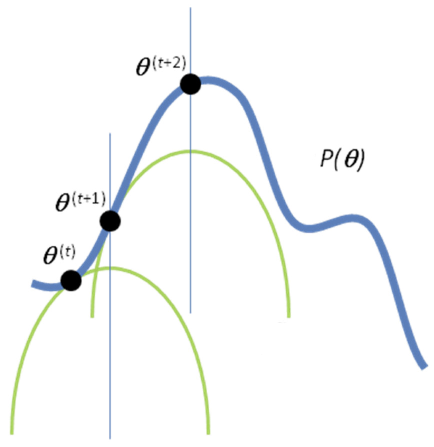
EM algorithm

A general algorithm for dealing with hidden data.

- EM is an optimization strategy for objective functions that can be interpreted as likelihoods in the presence of missing data.
- EM is much simpler than gradient methods: no need to choose step size.
- EM is an iterative algorithm with two steps:
  - E-step: fill-in hidden values using inference
  - M-step: apply standard MLE method to completed data
- We will prove that EM always converges to a local optimum of the likelihood.

High level idea

Keep maximizing **a lower bound of P** that is more manageable


$\boldsymbol{\theta}^{(t+2)}$
$P(\boldsymbol{\theta})$
$\boldsymbol{\theta}^{(t+1)}$
$\boldsymbol{\theta}^{(t)}$

Derivation of EM

Finding the lower bound of P:

$$\ln p(\boldsymbol{x} \,; \boldsymbol{\theta}) = \ln \frac{p(\boldsymbol{x}, z \,; \boldsymbol{\theta})}{p(z|\boldsymbol{x} \,; \boldsymbol{\theta})} \qquad \text{(true for any } z\text{)}$$

$$= \mathbb{E}_{z\sim q}\left[\ln \frac{p(\boldsymbol{x}, z \,; \boldsymbol{\theta})}{p(z|\boldsymbol{x} \,; \boldsymbol{\theta})}\right] \qquad \text{(true for any dist. } q\text{)}$$

Let us recall the definition of expectation

$$\mathbb{E}_{z\sim q}\left[f(z)\right] = \sum_z q(z) f(z)$$

and entropy

$$H(z) = -\mathbb{E}_{z\sim q}\left[\ln q(z)\right] = -\sum_z q(z) \ln q(z)$$

Derivation of EM

Finding the lower bound of P:

$$\ln p(\boldsymbol{x}\,;\boldsymbol{\theta}) = \ln \frac{p(\boldsymbol{x}, z\,;\boldsymbol{\theta})}{p(z|\boldsymbol{x}\,;\boldsymbol{\theta})} \qquad \text{(true for any } z\text{)}$$

$$= \mathbb{E}_{z\sim q}\left[\ln \frac{p(\boldsymbol{x}, z\,;\boldsymbol{\theta})}{p(z|\boldsymbol{x}\,;\boldsymbol{\theta})}\right] \qquad \text{(true for any dist. } q\text{)}$$

$$= \mathbb{E}_{z\sim q}\left[\ln p(\boldsymbol{x}, z\,;\boldsymbol{\theta})\right] - \mathbb{E}_{z\sim q}\left[\ln q(z)\right] - \mathbb{E}_{z\sim q}\left[\ln \frac{p(z|\boldsymbol{x}\,;\boldsymbol{\theta})}{q(z)}\right]$$

$$= \mathbb{E}_{z\sim q}\left[\ln p(\boldsymbol{x}, z\,;\boldsymbol{\theta})\right] + H(q) - \mathbb{E}_{z\sim q}\left[\ln \frac{p(z|\boldsymbol{x}\,;\boldsymbol{\theta})}{q(z)}\right] \qquad (H \text{ is entropy})$$

$$\geq \mathbb{E}_{z\sim q}\left[\ln p(\boldsymbol{x}, z\,;\boldsymbol{\theta})\right] + H(q) - \ln \mathbb{E}_{z\sim q}\left[\frac{p(z|\boldsymbol{x}\,;\boldsymbol{\theta})}{q(z)}\right]$$

(Jensen's inequality)

Jensen's inequality

Claim: $\mathbb{E}\left[\ln X\right] \leq \ln\left(\mathbb{E}[X]\right)$

Proof. By the definition of $\mathbb{E}[X] = \frac{1}{N}\left(x_1 + x_2 + \ldots + x_n\right)$, then

$$\mathbb{E}\left[\ln X\right] = \frac{1}{N}\left(\ln x_1 + \ln x_2 + \ldots + \ln x_n\right) = \frac{1}{N} \ln \prod_{n=1}^{N} x_n$$

It follows, that the above claim can be rewritten as

$$\frac{1}{N} \ln \prod_{n=1}^{N} x_n \leq \ln \frac{1}{N} \sum_{n=1}^{N} x_n$$

$$\sqrt[N]{\prod_{n=1}^{N} x_n} \leq \frac{1}{N} \sum_{n=1}^{N} x_n$$

This is the AGM inequality. For $N = 2$, it is just $(x_1 - x_2)^2 \geq 0$.

Derivation of EM

After applying Jensen's inequality, we obtain

$$\ln p(\boldsymbol{x}\,;\boldsymbol{\theta}) \geq \mathbb{E}_{z\sim q}\left[\ln p(\boldsymbol{x}, z\,;\boldsymbol{\theta})\right] + H(q) - \ln \mathbb{E}_{z\sim q}\left[\frac{p(z|\boldsymbol{x}\,;\boldsymbol{\theta})}{q(z)}\right]$$

Next, we observe that

$$\mathbb{E}_{z\sim q}\left[\frac{p(z|\boldsymbol{x}\,;\boldsymbol{\theta})}{q(z)}\right] = \sum_z q(z)\left(\frac{p(z|\boldsymbol{x}\,;\boldsymbol{\theta})}{q(z)}\right) = \sum_z p(z|\boldsymbol{x}\,;\boldsymbol{\theta}) = 1$$

It follows,

$$\ln p(\boldsymbol{x}\,;\boldsymbol{\theta}) \geq \mathbb{E}_{z\sim q}\left[\ln p(\boldsymbol{x}, z\,;\boldsymbol{\theta})\right] + H(q)$$

Alternatively maximize the lower bound

We have found a lower bound for the log-likelihood function

$$P(\boldsymbol{\theta}) = \sum_{n=1}^{N} \ln p(\boldsymbol{x}_n\,;\boldsymbol{\theta})$$

$$\geq \sum_{n=1}^{N}\left(\mathbb{E}_{z_n\sim q_n}\left[\ln p(\boldsymbol{x}_n, z_n\,;\boldsymbol{\theta})\right] + H(q_n)\right) = F(\boldsymbol{\theta}, \{q_n\})$$

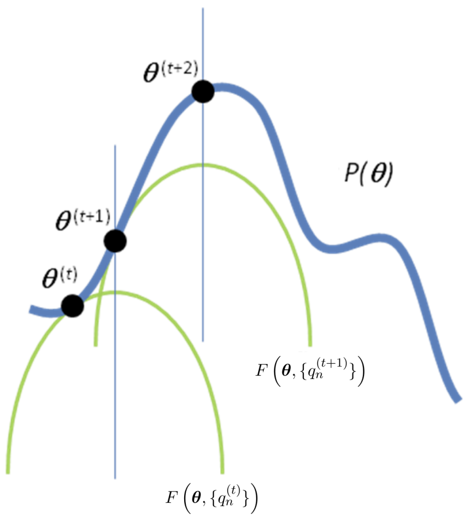
This holds for *any* $\{q_n\}$, so how do we choose?

Naturally, *the one that maximizes the lower bound* (i.e. the tightest lower bound)!

This is similar to K-means: we will alternatively maximizing F over $\{q_n\}$ and $\boldsymbol{\theta}$.

Pictorial explanation

$P(\boldsymbol{\theta})$ is non-concave, but $F\left(\boldsymbol{\theta}, \{q_n^{(t)}\}\right)$ often is concave and easy to maximize.



$\boldsymbol{\theta}^{(t+2)}$
$P(\boldsymbol{\theta})$
$\boldsymbol{\theta}^{(t+1)}$
$\boldsymbol{\theta}^{(t)}$
$F\left(\boldsymbol{\theta}, \{q_n^{(t+1)}\}\right)$
$F\left(\boldsymbol{\theta}, \{q_n^{(t)}\}\right)$

Maximizing over $\{q_n\}$

Fix $\boldsymbol{\theta}^{(t)}$, and maximize F over $\{q_n\}$

$$\operatorname*{argmax}_{q_n} F(\boldsymbol{\theta}, \{q_n\}) = \operatorname*{argmax}_{q_n} \left(\mathbb{E}_{z_n \sim q_n}\left[\ln p(\boldsymbol{x}_n, z_n\,; \boldsymbol{\theta}^{(t)})\right] + H(q_n)\right)$$

$$= \operatorname*{argmax}_{q_n} \sum_{k=1}^{K} \left(q_n(k) \ln p(\boldsymbol{x}_n, z_n = k\,; \boldsymbol{\theta}^{(t)}) - q_n(k) \ln q_n(k)\right)$$

subject to conditions:

$$q_n(k) \geq 0 \quad \text{and} \quad \sum_k q_n(k) = 1$$

Next, write down the Lagrangian and then apply KKT conditions.

Maximizing over $\{q_n\}$

The solution to

$$\operatorname*{argmax}_{q_n} F(\boldsymbol{\theta}, \{q_n\}) = \operatorname*{argmax}_{q_n} \mathbb{E}_{z_n \sim q_n}\left[\ln p(\boldsymbol{x}_n, z_n\,; \boldsymbol{\theta}^{(t)})\right] + H(q_n)$$

is (you have to verify it by yourself)

$$q_n^{(t)}(z_n) = p(z_n = k \mid \boldsymbol{x}_n\,; \boldsymbol{\theta}^{(t)})$$

i.e., the *posterior distribution of z_n* given $\boldsymbol{x}_n$ and $\boldsymbol{\theta}^{(t)}$.

So at $\boldsymbol{\theta}^{(t)}$, we found the tightest lower bound $F\left(\boldsymbol{\theta}, \{q_n^{(t)}\}\right)$:

- $F\left(\boldsymbol{\theta}, \{q_n^{(t)}\}\right) \leq P(\boldsymbol{\theta})$ for all $\boldsymbol{\theta}$.
- $F\left(\boldsymbol{\theta}^{(t)}, \{q_n^{(t)}\}\right) = P(\boldsymbol{\theta}^{(t)})$

Maximizing over $\boldsymbol{\theta}$

Fix $\{q_n^{(t)}\}$, maximize over $\boldsymbol{\theta}$ (note, $H(q_n^{(t)})$ is independent of $\boldsymbol{\theta}$):

$$\operatorname*{argmax}_{\boldsymbol{\theta}} F\left(\boldsymbol{\theta}, \{q_n^{(t)}\}\right)$$

$$= \operatorname*{argmax}_{\boldsymbol{\theta}} \sum_{n=1}^{N} \mathbb{E}_{z_n \sim q_n^{(t)}} \left[\ln p(\boldsymbol{x}_n, z_n\,; \boldsymbol{\theta})\right]$$

$$\triangleq \operatorname*{argmax}_{\boldsymbol{\theta}}\ Q(\boldsymbol{\theta}\,; \boldsymbol{\theta}^{(t)}) \qquad (\{q_n^{(t)}\} \text{ are computed via } \boldsymbol{\theta}^{(t)})$$

Q is called a **complete likelihood** and is usually more tractable, since z_n are not latent variables anymore.