CSCI567 Machine Learning (Fall 2018)

Prof. Haipeng Luo

U of Southern California

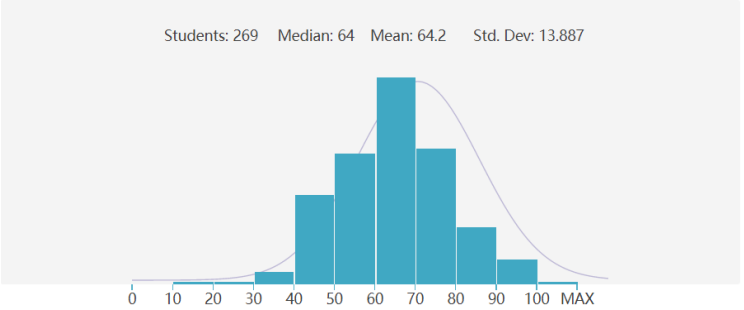
Oct 17, 2018

Administration

HW3 is due this Sunday. See Piazza announcements for two typos in P3.

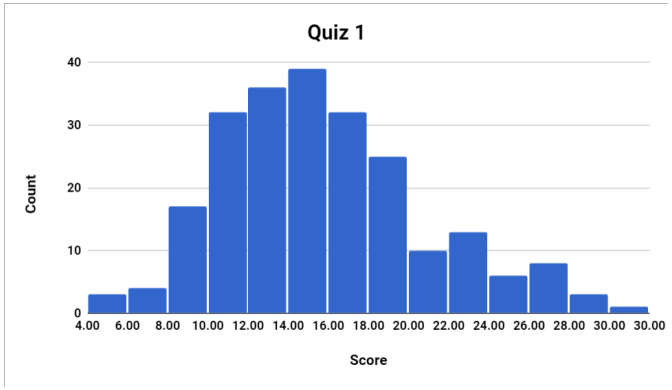
Midterm:

- Regrade requests later than 11:59PM 10/21 will not be considered
- Score distribution:



Data from last fall

Quiz1 mean and median were about 15/30. So similar to this semester.



Moving forward

Top 95% of the class will get at least B-

There will be a practice exam for the final

Final will be either shorter in terms of contents or longer in terms of time

Outline

Outline

Q1.1 (a)

Key: kernel methods are non-parametric

For example: kernelized linear regression (Lec 5) computes and stores

$$\boldsymbol{\alpha} = (\boldsymbol{K} + \lambda \boldsymbol{I})^{-1} \boldsymbol{y} \in \mathbb{R}^{\mathsf{N}}$$

Same for SVM dual (Lec 6), need to find α_n for each data point.

So for Q4.3 you should maintain α_n too.

Q3.2 (a)(b)

It provides a probabilistic interpretation of least square regression, similar to the probabilistic interpretation of logistic regression (Lec 3):

$$\begin{aligned}
\boldsymbol{w}^* &= \operatorname*{argmax}_{\boldsymbol{w}} \prod_{n=1}^{N} \mathbb{P}(y_n \mid \boldsymbol{x_n}; \boldsymbol{w}) \\
&= \operatorname*{argmax}_{\boldsymbol{w}} \sum_{n=1}^{N} \ln \mathbb{P}(y_n \mid \boldsymbol{x_n}; \boldsymbol{w}) = \operatorname*{argmin}_{\boldsymbol{w}} \sum_{n=1}^{N} - \ln \mathbb{P}(y_n \mid \boldsymbol{x_n}; \boldsymbol{w}) \\
&= \operatorname*{argmin}_{\boldsymbol{w}} \sum_{n=1}^{N} \ln(1 + e^{-y_n \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x_n}}) = \operatorname*{argmin}_{\boldsymbol{w}} \sum_{n=1}^{N} \ell_{\text{logistic}}(y_n \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_n)
\end{aligned}$$

i.e. *minimizing logistic loss is exactly doing MLE for the sigmoid model!*

Q4.1 (a)

Perceptron: repeat

- Pick a data point $\boldsymbol{x}_n$ uniformly at random
- If $\text{sgn}(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n) \neq y_n$

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + y_n\boldsymbol{x}_n$$

Perceptron converges in the separable case (as proven in W1), with zero training error, but does not guarantee to converge to a hyperplane with max-margin (SVM does).

Simple counterexample:

- $\boldsymbol{x}_1 = (1, 1), \boldsymbol{x}_2 = (-0.5, 1), \boldsymbol{x}_3 = (0, -1), y_1 = y_2 = +1, y_3 = -1$
- suppose $\boldsymbol{w}$ starts from $(0, 0)$ and we pick $(\boldsymbol{x}_1, y_1)$ first
- $\boldsymbol{w}$ converges to $(1, 1)$ (after one round)
- max-margin hyperplane is $(0, 1)$ instead

Q4.2

Perceptron: repeat

- Pick a data point $\boldsymbol{x}_n$ uniformly at random
- If $\text{sgn}(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_n) \neq y_n$

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + y_n\boldsymbol{x}_n$$

$\boldsymbol{w}$ is always a linear combination of the data

Without knowing the order of the updates, we can still determine the final weight vector.

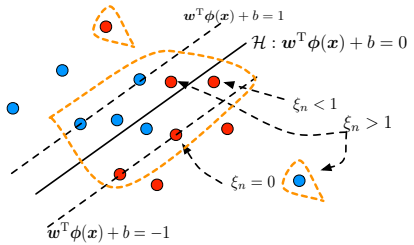
Q4.1 (c)

A support vector satisfies $\alpha_n^* \neq 0$ and

$$1 - \xi_n^* - y_n(\boldsymbol{w}^{*\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b^*) = 0$$

When

- $\xi_n^* = 0$, $y_n(\boldsymbol{w}^{*\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b^*) = 1$ and thus the point is $1/\|\boldsymbol{w}^*\|_2$ away from the hyperplane.
- $\xi_n^* < 1$, the point is classified correctly but does not satisfy the large margin constraint.
- $\xi_n^* > 1$, the point is misclassified.



Support vectors (circled with the orange line) are *the only points that matter!*

Q4.4

Key: can you calculate the gradient of

$$f(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{\mathsf{C}}) = \max\left\{0, \max_{k \neq y_n} \boldsymbol{w}_k^{\mathrm{T}}\boldsymbol{x}_n - w_{y_n}^{\mathrm{T}}\boldsymbol{x}_n\right\}?$$

With $\hat{y}_n = \text{argmax}_k\, \boldsymbol{w}_k^{\mathrm{T}}\boldsymbol{x}_n$,

$$\frac{\partial f}{\partial \boldsymbol{w}_k} = \begin{cases} \boldsymbol{0} & \text{if } y_n = \hat{y}_n \\ \boldsymbol{x}_n & \text{else if } k = \hat{y}_n \\ -\boldsymbol{x}_n & \text{else if } k = y_n \\ \boldsymbol{0} & \text{else} \end{cases}$$

Q4.5

(a) compare with W1 Q2.2

(b) compare with W1 Q2.1

(c) compare with W1 Q2.3

(d) tests whether you understand the concept of risk (Lec 1), and is also the second result you have seen in this class about the actual test performance guarantee of a learning algorithm (the first one is NNC)

(e) makes you think about why multiple passes of data is not necessarily good

5.1 (b)

(A) $\boldsymbol{W}_1\boldsymbol{W}_2\boldsymbol{x} = \boldsymbol{W}\boldsymbol{x}$ for $\boldsymbol{W} = \boldsymbol{W}_1\boldsymbol{W}_2$

(B) neural nets are non-convex in general, see discussions on Piazza

(C) more complicated function needs more neurons to represent

(D) max-pooling layer has no parameters to be learned

5.1 (d)(e)

Input: a volume of size $W_1 \times H_1 \times D_1$

Hyperparameters:

- K filters of size $F \times F$
- stride S
- amount of zero padding P (for one side)

Output: a volume of size $W_2 \times H_2 \times D_2$ where

- $W_2 = (W_1 + 2P - F)/S + 1$
- $H_2 = (H_1 + 2P - F)/S + 1$
- $D_2 = K$

#parameters: $(F \times F \times D_1 + 1) \times K$ weights

Outline

Supervised learning v.s unsupervised learning

Recall there are different types of machine learning problems

- **supervised learning** (what we have discussed by now)
  Aim to predict, e.g. classification and regression
- **unsupervised learning** (main focus from now on)
  Aim to discover hidden and latent patterns and explore data

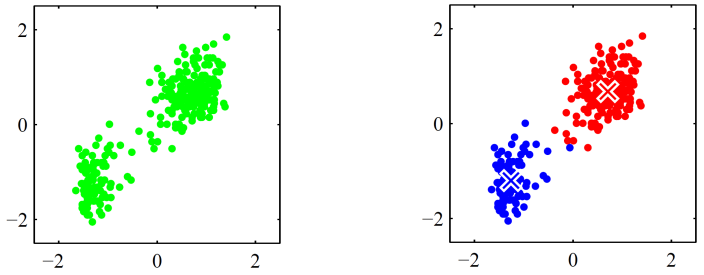
Today's focus: one important unsupervised learning problem: **clustering**

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

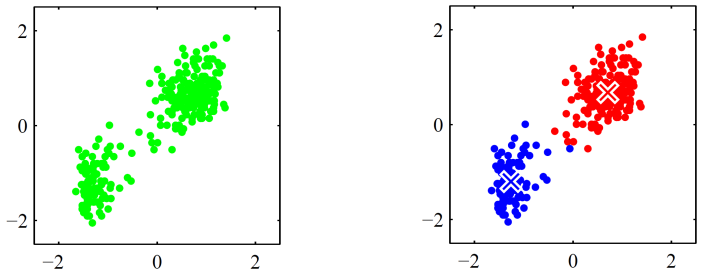


Clustering: formal definition

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

Output: group the data into K clusters, which means

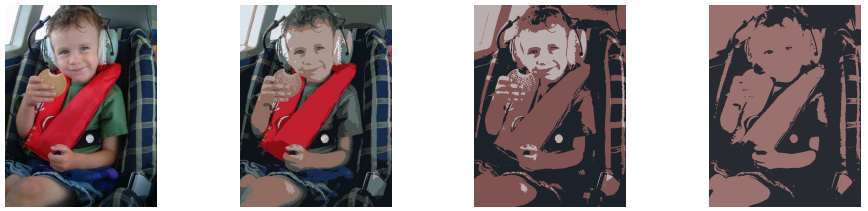
- find assignment $\gamma_{nk} \in \{0, 1\}$ for each data point $n \in [N]$ and $k \in [K]$ s.t. $\sum_{k \in [K]} \gamma_{nk} = 1$ for any fixed n
- find the cluster centers $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K \in \mathbb{R}^{\mathsf{D}}$



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

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

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 $t = 1, 2, \ldots$

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

A closer look

The first step

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

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

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

is simply **to average the points of each cluster** (hence the name)

$$\boldsymbol{\mu}_k = \frac{\sum_{n:\gamma_{nk}=1} \boldsymbol{x}_n}{|\{n : \gamma_{nk} = 1\}|} = \frac{\sum_n \gamma_{nk} \boldsymbol{x}_n}{\sum_n \gamma_{nk}}$$

for each $k \in [K]$.

The K-means algorithm

Step 0 Initialization

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

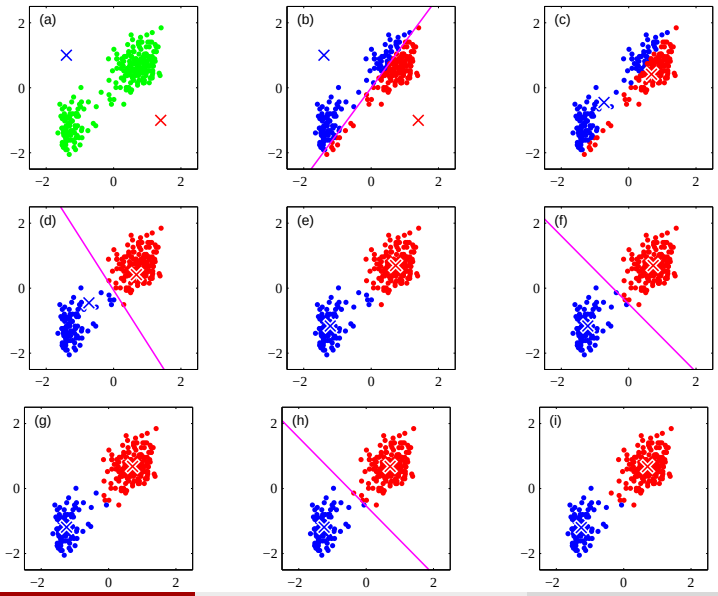
$$\gamma_{nk} = \mathbb{I}\left[k == \operatorname*{argmin}_c \|\boldsymbol{x}_n - \boldsymbol{\mu}_c\|_2^2\right]$$

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 Return to Step 1 if not converged

An example



How to initialize?

There are different ways to initialize:

- randomly pick K points as initial centers
- or randomly assign each point to a cluster
- or more sophisticated approaches (e.g. K-means++)

Convergence

It will converge in a finite number of iterations to a local minimum, why?

- objective decreases at each step
- objective is lower bounded by 0
- #possible_assignments is finite (K^N, exponentially large though)

However

- it could take *exponentially many iterations* to converge
- and it *might not converge to the global minimum*

Local minimum v.s global minimum

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

K-means converges immediately in both cases, but

- **left is local minimum, right is global minimum!**
- moreover, local minimum can be *arbitrarily worse* if we increase the width of this "rectangle"
- so *initialization matters a lot!*

Outline

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 mode: **Expectation–Maximization (EM) algorithm**

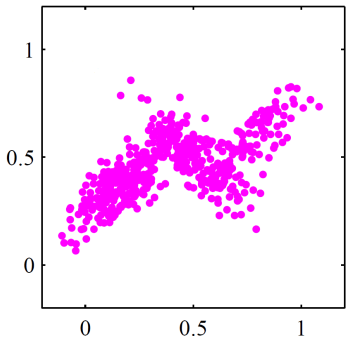
A generative model

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?

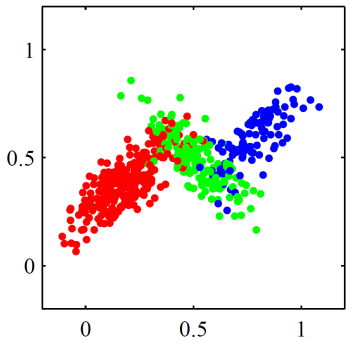


GMM: intuition

GMM is a natural model to explain such data

Assume there are 3 ground-truth Gaussian models. To generate a point, we

- first **randomly pick one of the Gaussian models**,
- then **draw a point according this Gaussian**.



Hence the name "**Gaussian mixture model**".

GMM: formal definition

A GMM has the following density function:

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

where

- K: the number of Gaussian components (same as #clusters we want)
- $\omega_1, \ldots, \omega_K$: mixture weights, a distribution over K components
- $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$: mean and covariance matrix of the k-th Gaussian
- N: the density function for a Gaussian

Another view

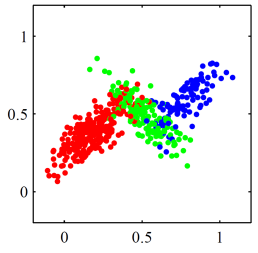
By introducing a **latent variable** $z \in [K]$, which indicates cluster membership, we can see p as a **marginal distribution**

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

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

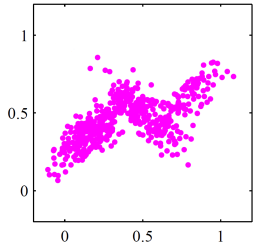
- $\boldsymbol{x}$ is observed
- z is unobserved/latent

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)

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

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

We will see how this is a special case of EM.

Demo

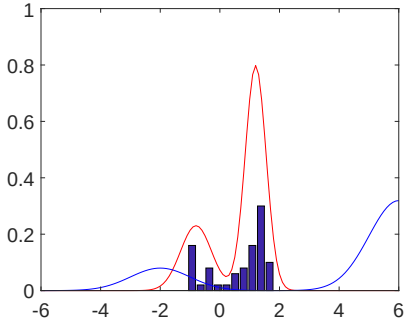
Generate 50 data points from a mixture of 2 Gaussians with

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

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

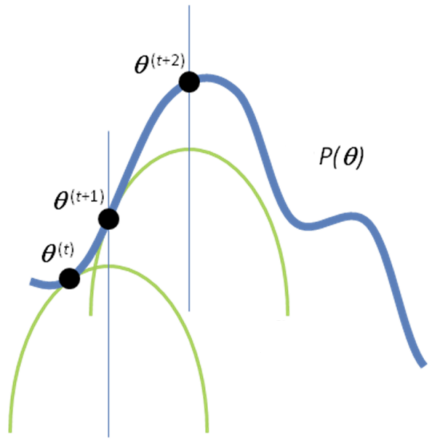
$$P(\boldsymbol{\theta}) = \sum_{n=1}^{N} \ln p(\boldsymbol{x}_n \,;\boldsymbol{\theta}) = \sum_{n=1}^{N} \ln \int_{z_n} p(\boldsymbol{x}_n, z_n \,;\boldsymbol{\theta}) dz_n$$

- $\boldsymbol{\theta}$ 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.

High level idea

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



Derivation of EM

Finding the lower bound of P:

$$\begin{aligned}
\ln p(\boldsymbol{x} \,;\boldsymbol{\theta}) &= \ln \frac{p(\boldsymbol{x}, z \,;\boldsymbol{\theta})}{p(z|\boldsymbol{x} \,;\boldsymbol{\theta})} && \text{(true for any } z\text{)} \\
&= \mathbb{E}_{z\sim q}\left[\ln \frac{p(\boldsymbol{x}, z \,;\boldsymbol{\theta})}{p(z|\boldsymbol{x} \,;\boldsymbol{\theta})}\right] && \text{(true for any dist. } q\text{)} \\
&= \mathbb{E}_{z\sim q}\left[\ln p(\boldsymbol{x}, z \,;\boldsymbol{\theta})\right] - \mathbb{E}_{z\sim q}\left[\ln q(z)\right] - \mathbb{E}_{z\sim q}\left[\ln \frac{p(z|\boldsymbol{x} \,;\boldsymbol{\theta})}{q(z)}\right] \\
&= \mathbb{E}_{z\sim q}\left[\ln p(\boldsymbol{x}, z \,;\boldsymbol{\theta})\right] + H(q) - \mathbb{E}_{z\sim q}\left[\ln \frac{p(z|\boldsymbol{x} \,;\boldsymbol{\theta})}{q(z)}\right] && (H \text{ is entropy}) \\
&\geq \mathbb{E}_{z\sim q}\left[\ln p(\boldsymbol{x}, z \,;\boldsymbol{\theta})\right] + H(q) - \ln \mathbb{E}_{z\sim q}\left[\frac{p(z|\boldsymbol{x} \,;\boldsymbol{\theta})}{q(z)}\right] \\
& && \text{(Jensen's inequality)} \\
&= \mathbb{E}_{z\sim q}\left[\ln p(\boldsymbol{x}, z \,;\boldsymbol{\theta})\right] + H(q)
\end{aligned}$$

Alternatively maximize the lower bound

Therefore, we obtain a lower bound for the log-likelihood function

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

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)!

Equivalently, this is the same as alternatively maximizing F over $\{q_n\}$ and $\boldsymbol{\theta}$ (similar to K-means).

Maximizing over $\{q_n\}$

Fix $\boldsymbol{\theta}^{(t)}$, the solution to

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

is $q_n^{(t)}$ s.t.

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

i.e., the *posterior distribution of* z_n given $\boldsymbol{x}_n$ and $\boldsymbol{\theta}^{(t)}$. (Verified in W4)

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

- $F\left(\boldsymbol{\theta}, \{q_n^{(t)}\}\right) \le P(\boldsymbol{\theta})$ for all $\boldsymbol{\theta}$.
- $F\left(\boldsymbol{\theta}^{(t)}, \{q_n^{(t)}\}\right) = P(\boldsymbol{\theta}^{(t)})$ (verify yourself by going through Slide 43)

Maximizing over $\boldsymbol{\theta}$

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

$$\begin{aligned}
&\operatorname*{argmax}_{\boldsymbol{\theta}} F\left(\boldsymbol{\theta}, \{q_n^{(t)}\}\right) \\
&= \operatorname*{argmax}_{\boldsymbol{\theta}} \sum_{n=1}^{N} \mathbb{E}_{z_n \sim q_n^{(t)}} \left[\ln p(\boldsymbol{x}_n, z_n \,;\boldsymbol{\theta})\right] \quad (H(q_n^{(t)}) \text{ is independent of } \boldsymbol{\theta}) \\
&\triangleq \operatorname*{argmax}_{\boldsymbol{\theta}} Q(\boldsymbol{\theta} \,;\boldsymbol{\theta}^{(t)}) \quad (\{q_n^{(t)}\} \text{ are computed via } \boldsymbol{\theta}^{(t)})
\end{aligned}$$

Q is the (expected) **complete likelihood** and is usually more tractable.

General EM algorithm

Step 0 Initialize $\boldsymbol{\theta}^{(1)}$, $t = 1$

Step 1 (E-Step) **update the posterior of latent variables**

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

and obtain **Expectation** of complete likelihood

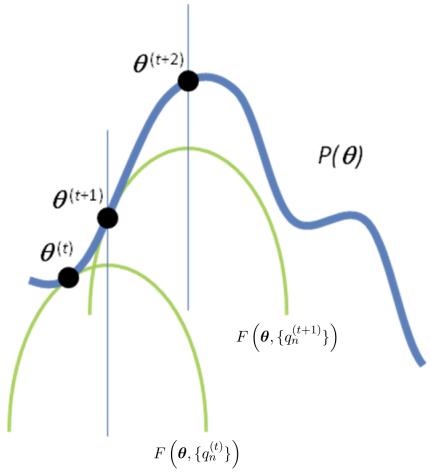
$$Q(\boldsymbol{\theta} \,;\boldsymbol{\theta}^{(t)}) = \sum_{n=1}^{N} \mathbb{E}_{z_n \sim q_n^{(t)}} \left[\ln p(\boldsymbol{x}_n, z_n \,;\boldsymbol{\theta})\right]$$

Step 2 (M-Step) **update the model parameter** via **Maximization**

$$\boldsymbol{\theta}^{(t+1)} \leftarrow \operatorname*{argmax}_{\boldsymbol{\theta}} Q(\boldsymbol{\theta} \,;\boldsymbol{\theta}^{(t)})$$

Step 3 $t \leftarrow t + 1$ and return to Step 1 if not converged

Pictorial explanation



$P(\boldsymbol{\theta})$ is non-concave, but $Q(\boldsymbol{\theta}; \boldsymbol{\theta}^{(t)})$ often is concave and easy to maximize.

$$\begin{aligned}
P(\boldsymbol{\theta}^{(t+1)}) &\ge F\left(\boldsymbol{\theta}^{(t+1)} \,;\{q_n^{(t)}\}\right) \\
&\ge F\left(\boldsymbol{\theta}^{(t)} \,;\{q_n^{(t)}\}\right) \\
&= P(\boldsymbol{\theta}^{(t)})
\end{aligned}$$

So EM always increases the objective value and will converge to some local maximum (similar to K-means).

Apply EM to learn GMMs

E-Step:

$$\begin{aligned} q_n^{(t)}(z_n = k) &= p\left(z_n = k \mid \boldsymbol{x}_n \,; \boldsymbol{\theta}^{(t)}\right) \\ &\propto p\left(\boldsymbol{x}_n, z_n = k \,; \boldsymbol{\theta}^{(t)}\right) \\ &= p\left(z_n = k \,; \boldsymbol{\theta}^{(t)}\right) p(\boldsymbol{x}_n \mid z_n = k \,; \boldsymbol{\theta}^{(t)}\right) \\ &= \omega_k^{(t)} N\left(\boldsymbol{x}_n \mid \boldsymbol{\mu}_k^{(t)}, \boldsymbol{\Sigma}_k^{(t)}\right) \end{aligned}$$

This computes the "soft assignment" $\gamma_{nk} = q_n^{(t)}(z_n = k)$, i.e. conditional probability of $\boldsymbol{x}_n$ belonging to cluster k.

Apply EM to learn GMMs

M-Step:

$$\begin{aligned} \operatorname*{argmax}_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)}) &= \operatorname*{argmax}_{\boldsymbol{\theta}} \sum_{n=1}^{N} \mathbb{E}_{z_n \sim q_n^{(t)}} \left[\ln p(\boldsymbol{x}_n, z_n \,; \boldsymbol{\theta})\right] \\ &= \operatorname*{argmax}_{\boldsymbol{\theta}} \sum_{n=1}^{N} \mathbb{E}_{z_n \sim q_n^{(t)}} \left[\ln p(z_n \,; \boldsymbol{\theta}) + \ln p(\boldsymbol{x}_n | z_n \,; \boldsymbol{\theta})\right] \\ &= \operatorname*{argmax}_{\{\omega_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}} \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma_{nk} \left(\ln \omega_k + \ln N(\boldsymbol{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right) \end{aligned}$$

To find $\omega_1, \ldots, \omega_K$, solve

$$\operatorname*{argmax}_{\boldsymbol{\omega}} \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma_{nk} \ln \omega_k$$

To find each $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$, solve

$$\operatorname*{argmax}_{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k} \sum_{n=1}^{N} \gamma_{nk} \ln N(\boldsymbol{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

M-Step (continued)

Solutions to previous two problems are very natural, for each k

$$\omega_k = \frac{\sum_n \gamma_{nk}}{N}$$

i.e. (weighted) fraction of examples belonging to cluster k

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

i.e. (weighted) average of examples belonging to cluster k

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

i.e (weighted) covariance of examples belonging to cluster k

You will verify some of these in W4.

Putting it together

EM for clustering:

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

Connection to K-means

K-means is in fact a special case of EM for (a simplified) GMM:

- assume $\boldsymbol{\Sigma}_k = \sigma^2 \boldsymbol{I}$ for some fixed σ so only ω_k and $\boldsymbol{\mu}_k$ are parameters
- when $\sigma \to 0$, EM becomes K-means

GMM is a soft version of K-means and it provides a probabilistic interpretation of the data, which means we can predict and generate data after learning.