Expectation Maximization (EM).
Mixtures of Gaussians.

Learning probability distribution

**Basic learning settings:**
- A set of random variables $X = \{X_1, X_2, \ldots, X_n\}$
- A model of the distribution over variables in $X$ with parameters $\Theta$
- Data $D = \{D_1, D_2, \ldots, D_N\}$
  s.t. $D_i = (x_1^i, x_2^i, \ldots, x_n^i)$

**Objective:** find parameters $\hat{\Theta}$ that describe the data

**Assumptions considered so far:**
- Known parameterizations
- No hidden variables
- No-missing values
General EM

The key idea of a method:
Compute the parameter estimates iteratively by performing the following two steps:

Two steps of the EM:
1. **Expectation step.** Complete all hidden and missing variables with expectations for the current set of parameters $\Theta'$
2. **Maximization step.** Compute the new estimates of $\Theta$ for the completed data

Stop when no improvement possible

---

EM

Let $H$ – be a set of hidden or missing values

**Derivation**

$$P(H, D \mid \Theta ', \xi ) = P(H \mid D, \Theta , \xi )P(D \mid \Theta , \xi )$$

$$\log P(H, D \mid \Theta , \xi ) = \log P(H \mid D, \Theta , \xi ) + \log P(D \mid \Theta , \xi )$$

$$\log P(D \mid \Theta , \xi ) = \log P(H, D \mid \Theta , \xi ) - \log P(H \mid D, \Theta , \xi )$$

Log-likelihood of data

**Average both sides** with $P(H \mid D, \Theta ', \xi )$ for some $\Theta'$

$$E_{H,D,\Theta'} \log P(D \mid \Theta , \xi ) = E_{H,D,\Theta'} \log P(H, D \mid \Theta , \xi ) - E_{H,D,\Theta'} \log P(H \mid \Theta , \xi )$$

$$\log P(D \mid \Theta , \xi ) = Q(\Theta \mid \Theta ') + H(\Theta \mid \Theta ')$$

Log-likelihood of data
EM algorithm

**Algorithm (general formulation)**

Initialize parameters $\Theta$

Repeat

Set $\Theta' = \Theta$

1. **Expectation step**
   \[ Q(\Theta | \Theta') = E_{H,D,\Theta} \log P(H,D | \Theta, \xi) \]

2. **Maximization step**
   \[ \hat{\Theta} = \arg \max_{\Theta} Q(\Theta | \Theta') \]
   until no or small improvement in $\Theta$ ($\Theta = \Theta'$)

**Questions:** Why this leads to the ML estimate?

What is the advantage of the algorithm?

---

EM algorithm

- Why is the EM algorithm correct?
- **Claim:** maximizing Q improves the log-likelihood
  \[ l(\Theta) = Q(\Theta | \Theta') + H(\Theta | \Theta') \]

**Difference in log-likelihoods (current and next step)**

\[ l(\Theta) - l(\Theta') = Q(\Theta | \Theta') - Q(\Theta' | \Theta') + H(\Theta | \Theta') - H(\Theta' | \Theta') \]

**Subexpression**

\[ H(\Theta | \Theta') - H(\Theta' | \Theta') \geq 0 \]

**Kullback-Leibler (KL) divergence** (distance between 2 distributions)

\[ KL(P \mid R) = \sum_i P_i \log \frac{P_i}{R_i} \geq 0 \quad \text{is always positive !!!} \]

\[ H(\Theta | \Theta') = -E_{H,D,\Theta} \log P(H | \Theta, D, \xi) = -\sum_i p(H | D, \Theta') \log P(H | \Theta, D, \xi) \]

\[ H(\Theta | \Theta') - H(\Theta' | \Theta') = \sum_i P(H | D, \Theta') \log \frac{P(H | \Theta', D, \xi)}{P(H | \Theta, D, \xi)} \geq 0 \]
**EM algorithm**

**Difference in log-likelihoods**

\[
l(\Theta) - l(\Theta') = Q(\Theta \mid \Theta') - Q(\Theta' \mid \Theta') + H(\Theta \mid \Theta') - H(\Theta' \mid \Theta')
\]

Thus

by maximizing Q we maximize the log-likelihood

\[
l(\Theta) = Q(\Theta \mid \Theta') + H(\Theta \mid \Theta')
\]

EM is a first-order optimization procedure

- **Climbs the gradient**
- **Automatic learning rate**

    No need to adjust the learning rate !!!!

---

**EM advantages**

**Key advantages:**

- In many problems (e.g. Bayesian belief networks)

  \[
  Q(\Theta \mid \Theta') = E_{H \mid D, \Theta'} \log P(H, D \mid \Theta, \xi)
  \]

  – has a nice form and the maximization of Q can be carried out in the closed form

- No need to compute Q before maximizing

- We directly optimize
  – using quantities corresponding to expected counts

---
Naïve Bayes with a hidden class and missing values

Assume:
1. $P(X)$ is modeled using a Naïve Bayes model with hidden class variable
2. Missing entries (values) for attributes in the dataset $D$

Hidden class variable

Attributes are independent given the class

EM for the Naïve Bayes

- We can use EM to learn the parameters
  \[ Q(\Theta | \Theta^{'}) = E_{H|D,\Theta^{'}} \log P(H, D | \Theta, \xi) \]
- Parameters:
  - $\pi_j$: prior on class $j$
  - $\theta_{jk}$: probability of an attribute $i$ having value $k$ given class $j$
- Indicator variables:
  - $\delta_{jl}^i$: for example $l$, the class is $j$; if true (=1) else false (=0)
  - $\delta_{jk}^i$: for example $l$, the class is $j$ and the value of attrib $i$ is $k$
- because the class is hidden and some attributes are missing, the values (0,1) of indicator variables are not known; they are hidden

$H$ – a collection of all indicator variables
EM for the Naïve Bayes model

- We can use EM to do the learning of parameters

\[ Q(\Theta \mid \Theta') = E_{H,D,\Theta'} \log P(H,D \mid \Theta, \xi) \]

\[ \log P(H,D \mid \Theta, \xi) = \log \prod_{l=1}^{N} \prod_{j} \pi_j^\delta_j \prod_{i} \prod_{k} \theta_{ij}^{\delta_{ik}} \]

\[ = \sum_{l=1}^{N} \sum_{j} (\delta_j^l \log \pi_j + \sum_{i} \sum_{k} \delta_{ik}^l \log \theta_{ij}) \]

\[ E_{H,D,\Theta} \log P(H,D \mid \Theta, \xi) = \sum_{l=1}^{N} \sum_{j} (E_{H,D,\Theta} (\delta_j^l) \log \pi_j + \sum_{i} \sum_{k} E_{H,D,\Theta} (\delta_{ik}^l) \log \theta_{ij}) \]

\[ E_{H,D,\Theta} (\delta_j^l) = p(C_i = j \mid D_i, \Theta') \]

\[ E_{H,D,\Theta} (\delta_{ik}^l) = p(X_{il} = k, C_i = j \mid D_i, \Theta') \]

Substitutes 0, 1 with expected value

- Computing derivatives of \( Q \) for parameters and setting it to 0 we get:

\[ \pi_j = \frac{\tilde{N}_j}{N} \]

\[ \theta_{ik} = \frac{\tilde{N}_{ik}}{\sum_{k=1}^{N} \tilde{N}_{jk}} \]

\[ \tilde{N}_j = \sum_{l=1}^{N} E_{H,D,\Theta} (\delta_j^l) = \sum_{l=1}^{N} p(C_i = j \mid D_i, \Theta') \]

\[ \tilde{N}_{ik} = \sum_{l=1}^{N} E_{H,D,\Theta} (\delta_{ik}^l) = \sum_{l=1}^{N} p(X_{il} = k, C_i = j \mid D_i, \Theta') \]

- Important:
  - Use expected counts instead of counts !!!!
  - Re-estimate the parameters using expected counts
EM for BBNs

- The same result applies to learning of parameters of any Bayesian belief network with discrete-valued variables

\[ Q(\Theta | \Theta') = E_{H|D, \Theta'} \log P(H, D | \Theta, \xi) \]

\[ \theta_{ijk} = \frac{\tilde{N}_{ijk}}{\sum_{k=1}^{N} \tilde{N}_{ijk}} \quad \text{Parameter value maximizing } Q \]

\[ \tilde{N}_{ijk} = \sum_{l=1}^{N} p(x_i^l = k, pa_i^l = j | D^l', \Theta') \]

- Again:
  - Use expected counts instead of counts

---

Gaussian mixture model
Mixture of Gaussians

- Density function for the Mixture of Gaussians model

Gaussian mixture model

Probability of occurrence of a data point $x$ is modeled as

$$p(x) = \sum_{i=1}^{k} p(C = i) p(x \mid C = i)$$

where

- $p(C = i)$ = probability of a data point coming from class $C=i$
- $p(x \mid C = i) \approx N(\mu_i, \Sigma_i)$ = class conditional density (modeled as a Gaussian) for class $i$

**Special feature:** $C$ is hidden !!!!
Generative Naïve Bayes classifier model

- Generative classifier model based on the Naïve Bayes
- Assume the class labels are known. The ML estimate is

\[
N_i = \sum_{j: C_j = i} 1
\]

\[
\tilde{\pi}_i = \frac{N_i}{N}
\]

\[
\tilde{\mu}_i = \frac{1}{N_i} \sum_{j: C_j = i} x_j
\]

\[
\tilde{\Sigma}_i = \frac{1}{N_i} \sum_{j: C_j = i} (x_j - \tilde{\mu}_i)(x_j - \tilde{\mu}_i)^T
\]

Gaussian mixture model

- In the Gaussian mixture Gaussians are not labeled
- We can apply **EM algorithm**: re-estimation based on the class posterior

\[
h_{il} = p(C_i = i \mid x_i, \Theta') = \frac{p(C_i = i \mid \Theta') p(x_i \mid C_i = i, \Theta')}{{\sum}_{u=1}^{m} p(C_i = u \mid \Theta') p(x_i \mid C_i = u, \Theta')}
\]

\[
N_i = \sum_{l} h_{il}
\]

\[
\tilde{\pi}_i = \frac{N_i}{N}
\]

\[
\tilde{\mu}_i = \frac{1}{N_i} \sum_{l} h_{il} x_j
\]

\[
\tilde{\Sigma}_i = \frac{1}{N_i} \sum_{l} h_{il} (x_j - \tilde{\mu}_i)(x_j - \tilde{\mu}_i)^T
\]
**Gaussian mixture algorithm**

- **Special case:** fixed covariance matrix for all hidden groups (classes) and uniform prior on classes

- **Algorithm:**
  
  Initialize means \( \mu_i \) for all classes \( i \)

  Repeat two steps until no change in the means:

  1. Compute the class posterior for each Gaussian and each point (a kind of responsibility for a Gaussian for a point)

     \[
     h_{il} = \frac{p(C_i = i \mid \Theta') p(x_i \mid C_i = i, \Theta')}{\sum_{u=1}^{m} p(C_j = u \mid \Theta') p(x_i \mid C_j = u, \Theta')}
     \]

  2. Move the means of the Gaussians to the center of the data, weighted by the responsibilities

     \[
     \mu_i = \frac{\sum_{i=1}^{N} h_{il} x_i}{\sum_{i=1}^{N} h_{il}}
     \]

---

**Gaussian mixture model. Gradient ascent.**

- A set of parameters

  \[
  \Theta = \{ \pi_1, \pi_2, \ldots, \pi_m, \mu_1, \mu_2, \ldots, \mu_m \}
  \]

  Assume unit variance terms and fixed priors

  \[
  P(x \mid C = i) = (2\pi)^{-1/2} \exp \left\{ -\frac{1}{2} \| x - \mu_i \|^2 \right\}
  \]

  \[
  P(D \mid \Theta) = \prod_{i=1}^{N} \sum_{i=1}^{m} \pi_i (2\pi)^{-1/2} \exp \left\{ -\frac{1}{2} \| x_i - \mu_i \|^2 \right\}
  \]

  \[
  l(\Theta) = \sum_{i=1}^{N} \log \sum_{i=1}^{m} \pi_i (2\pi)^{-1/2} \exp \left\{ -\frac{1}{2} \| x_i - \mu_i \|^2 \right\}
  \]

  \[
  \frac{\partial l(\Theta)}{\partial \mu_i} = \sum_{i=1}^{N} h_{il} (x_i - \mu_i) \quad \text{-- very easy on-line update}
  \]
**EM versus gradient ascent**

**Gradient ascent**

\[
\mu_j \leftarrow \mu_j + \alpha \sum_{i=1}^{N} h_{ij} (x_i - \mu_j)
\]

**Learning rate**

Small pull towards distant uncovered data

---

**EM**

\[
\mu_j \leftarrow \frac{\sum_{j=1}^{N} h_{ij} x_i}{\sum_{j=1}^{N} h_{ij}}
\]

**No learning rate**

Renormalized – big jump in the first step

---

**K-means approximation to EM**

**Mixture of Gaussians with the fixed covariance matrix:**

- posterior measures the responsibility of a Gaussian for every point
  \[
  h_{ij} = \frac{p(C_j = i \mid \Theta') p(x_i \mid C_j = i, \Theta')}{\sum_{u=1}^{N} p(C_j = u \mid \Theta') p(x_i \mid C_j = u, \Theta')}
  \]

- **Re-estimation of means:**
  \[
  \mu_j = \frac{\sum_{j=1}^{N} h_{ij} x_i}{\sum_{j=1}^{N} h_{ij}}
  \]

- **K-Means approximations**
  - Only the closest Gaussian is made responsible for a point
    
    \[
    h_{ij} = \begin{cases} 
    1 & \text{If } i \text{ is the closest Gaussian} \\
    0 & \text{Otherwise}
    \end{cases}
    \]
  - Results in moving the means of Gaussians to the center of the data points it covered in the previous step
K-means algorithm

K-Means algorithm:
Initialize k values of means (centers)
Repeat two steps until no change in the means:
– Partition the data according to the current means (using the similarity measure)
– Move the means to the center of the data in the current partition

• Used frequently for clustering data