Expectation maximization (EM)

Milos Hauskrecht
milos@cs.pitt.edu
5329 Sennott Square

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
**Hidden variables**

**Modeling assumption:** Variables \( X = \{X_1, X_2, \ldots, X_n\} \) are related through hidden variables

**Why to add hidden variables?**
- More flexibility in describing the distribution \( P(X) \)
- Smaller parameterization of \( P(X) \)
  - New independences can be introduced via hidden variables

**Example:**
- Latent variable models
  - hidden classes (categories)

```
\begin{align*}
\text{Hidden class variable} & \quad C \\
& \quad P(X \mid C = i) \\
& \quad X
\end{align*}
```

**Hidden variable model. Example.**
- We want to represent the probability model of a population in a two dimensional space \( X = \{X_1, X_2\} \)

![Observed data](image)
Hidden variable model

- We want to represent the probability model of a population in a two dimensional space \( X = \{X_1, X_2\} \)

**Observed data**

![Observed data diagram](image-url)
**Hidden variable model**

- We want to represent the probability model of a population in a two-dimensional space \( \mathbf{X} = \{X_1, X_2\} \).

**Model**: 3 Gaussians with a hidden class variable.

**Observed data**

![Observed data diagram](image)

**Mixture of Gaussians**

Probability of the occurrence of a data point \( \mathbf{x} \) is modeled as

\[
p(\mathbf{x}) = \sum_{i=1}^{k} p(C = i) p(\mathbf{x} | C = i)
\]

where

\[
p(C = i)
\]

= probability of a data point coming from class \( C=i \)

\[
p(\mathbf{x} | C = i) \approx N(\mathbf{\mu}_i, \mathbf{\Sigma}_i)
\]

= class-conditional density (modeled as Gaussian) for class \( i \)
Mixture of Gaussians

- Density function for the Mixture of Gaussians model

Naïve Bayes with a hidden class variable

Introduction of a hidden variable can reduce the number of parameters defining $P(X)$

Example:
- Naïve Bayes model with a hidden class variable

  Hidden class variable
  
  Attributes are independent given the class

  • Useful in customer profiles
    - Class value = type of customers
Missing values

A set of random variables \( X = \{X_1, X_2, \ldots, X_n\} \)

- **Data** \( D = \{D_1, D_2, \ldots, D_N\} \)
- **But some values are missing**
  \[
  D_i = (x_1^i, x_2^i, \ldots, x_n^i)
  \]
- Missing value of \( x_2^i \)
  \[
  D_{i+1} = (x_1^i, \ldots, x_n^i)
  \]
- Missing values of \( x_1^i, x_2^i \)
  Etc.

- **Example:** medical records
- **We still want to estimate parameters of** \( P(X) \)

Density estimation

**Goal:** Find the set of parameters \( \hat{\Theta} \)

**Estimation criteria:**
- ML \( \max_\Theta p(D \mid \Theta, \xi) \)
- Bayesian \( p(\Theta \mid D, \xi) \)

**Optimization methods for ML:** gradient-ascent, conjugate gradient, Newton-Rhapson, etc.

- **Problem:** No or very small advantage from the structure of the corresponding belief network

**Expectation-maximization (EM) method**
- An alternative optimization method
- Suitable when there are missing or hidden values
- **Takes advantage of the structure of the belief network**
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 all variables with hidden or missing values

Derivation

$$P(H, D | \Theta, \xi) = P(H | D, \Theta, \xi)P(D | \Theta, \xi)$$

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

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

Log-likelihood of data

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

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

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

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**
   
   $\Theta = \arg \max_{\Theta} Q(\Theta | \Theta')$

   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)

$KL(P \mid R) = \sum_i P_i \log \frac{P_i}{R_i} \geq 0$  

Is always positive !!!

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

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

Difference in log-likelihoods

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

\[ l(\Theta) - l(\Theta') \geq Q(\Theta | \Theta') - Q(\Theta' | \Theta') \]

Thus

by maximizing Q we maximize the log-likelihood

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

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)

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

– has a nice form and the maximization of Q can be carried in the closed form
• No need to compute Q before maximizing
• We directly optimize
  – use quantities corresponding to expected counts
**Naïve Bayes with a hidden class and missing values**

**Assume:**
- \( P(X) \) is modeled using a Naïve Bayes model with hidden class variable
- Missing entries (values) for attributes in the dataset \( D \)

**Hidden class variable**

![Diagram showing a hidden class variable and attributes](image)

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:**
  - \( \delta_j^l \): for example \( l \), the class is \( j \); if true (=1) else false (=0)
  - \( \delta_{jk}^l \): 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
  \[ Q(\Theta | \Theta') = E_{H,D,\Theta'} \log P(H, D | \Theta, \xi) \]
  \[
  \log P(H, D | \Theta, \xi) = \log \prod_{l=1}^{N} \prod_{j} \pi_{j}^{\delta_{j}^l} \prod_{i} \prod_{k} \theta_{ik}^{\xi_{ik}} \\
  = \sum_{l=1}^{N} \sum_{j} (\delta_{j}^l \log \pi_{j} + \sum_{i} \sum_{k} \delta_{ik}^{\xi_{ik}} \log \theta_{ik})
  \]

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

\[
E_{H,D,\Theta}(\delta_{j}^l) = p(C_i = j | D_l, \Theta')
\]

Substitutes 0,1 with expected value

EM for Naïve Bayes model

• Computing derivatives of \( Q \) for parameters and setting it to 0 we get:
  \[ \pi_{j} = \frac{\tilde{N}_{j}}{N} \quad \theta_{ik} = \frac{\tilde{N}_{ik}}{\sum_{k=1}^{l} \tilde{N}_{jk}} \]

\[
\tilde{N}_{j} = \sum_{l=1}^{N} E_{H,D,\Theta}(\delta_{j}^l) = \sum_{l=1}^{N} p(C_i = j | D_l, \Theta')
\]

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

• 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

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

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

\[ \tilde{N}_{ijk} = \sum_{l=1}^{N} p(x_i^l = k, pa_i^l = j \mid D', \Theta') \quad \text{may require inference} \]

• Again:
  – Use expected counts instead of counts

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 \( i \)

**Remember:** \( C \) 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

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

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

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

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

Gaussian mixture model

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

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

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

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

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

\[
\tilde{\Sigma}_i = \frac{1}{N_i} \sum_{l} h_{il} (x_i - \mu_i)(x_i - \mu_i)^T
\]
**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)

  \[
  h_d = \frac{p(C_i = i | \Theta^*) p(x_i | C_i = i, \Theta^*)}{\sum_{u=1}^{m} p(C_j = u | \Theta^*) p(x_i | C_j = u, \Theta^*)}
  \]

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

  \[
  \text{New mean: } \mu_i = \frac{\sum_{j=1}^{N} h_{ij} x_j}{\sum_{j=1}^{N} h_{ij}}
  \]
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

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

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

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

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

**Gradient ascent**

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

**EM**

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

**Learning rate**

- Small pull towards distant uncovered data
- No learning rate

Renormalized – big jump in the first step

---

K-means approximation to EM

**Expectation-Maximization:**

- Posterior measures the responsibility of a Gaussian for every point

\[ h_{ij} = \frac{p(C_i = i \mid i, \Theta^*) p(x_j \mid C_i = i, \Theta^*)}{\sum_{u=1}^{k} p(C_j = u \mid \Theta^*) p(x_j \mid C_j = u, \Theta^*)} \]

**K- Means**

- Only the closest Gaussian is made responsible for a point

\[ h_{ij} = \begin{cases} 1 & \text{if } i \text{ is the closest Gaussian} \\ 0 & \text{otherwise} \end{cases} \]

**Re-estimation of means**

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

- Results in moving the means of Gaussians to the center of the data points it covered in the previous step

---

CS 3750 Machine Learning
K-means algorithm

Useful for clustering data:
• Assume we want to distribute data into \(k\) different groups
  – Similarity between data points is measured in terms of the distance
  – Groups are defined in terms of centers (also called means)

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

Properties
• converges to centers minimizing the sum of center-point distances (local optima)
• The result may be sensitive to the initial means’ values

Advantages:
• Simplicity
• Generality – can work for an arbitrary distance measure

Drawbacks:
• Can perform poorly on overlapping regions
• Lack of robustness to outliers (outliers are not covered)