CS 2750 Machine Learning
Lecture 21

Learning with hidden variables and missing values.

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

Learning probability distribution

Basic learning settings:

- A set of random variables $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$
- **A model of the distribution** over variables in ***X*** with parameters Θ
- **Data** $D = \{D_1, D_2, .., D_N\}$

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

Variables can be:

- – Hidden
- – Values can be missing

Hidden variables

Modeling assumption:

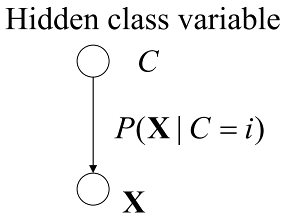
Variables $\mathbf{X} = \{X_1, X_2, \dots, X_n\}$ are related through hidden variables

Why to add hidden variables?

- **More flexibility in describing the distribution** $P(\mathbf{X})$
- **Smaller parameterization of** $P(\mathbf{X})$
  - **New independences can be introduced via hidden variables**

Example:

- Latent variable models
  - hidden classes (categories)



Missing values

A set of random variables $\mathbf{X} = \{X_1, X_2, \dots, X_n\}$

- **Data** $D = \{D_1, D_2, .., D_N\}$
- **But some values are missing**

$D_i = (x_1^i, x_3^i, \dots x_n^i)$

Missing value of x_2^i

$D_{i+1} = (x_3^i, \dots x_n^i)$

Missing values of x_1^i, x_2^i

Etc.

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

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 $\mathbf{\Theta}'$
2. **Maximization step**. Compute the new estimates of $\mathbf{\Theta}$ for the completed data

Stop when no improvement possible

EM algorithm

Algorithm (general formulation)

```
Initialize parameters $\Theta$
Repeat
Set    $\Theta' = \Theta$
1. Expectation step
       $Q(\Theta \mid \Theta') = E_{H|D,\Theta'} \log P(H, D \mid \Theta, \xi)$
2. Maximization step
       $\Theta = \arg\max_{\Theta} Q(\Theta \mid \Theta')$
   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 advantages

Key advantages:

- For Bayesian belief networks

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

  – **Q decomposes along variables** (has a nice form)

$$\log P(H, D | \Theta, \xi) = \log \prod_{l=1}^{N} P(H^{(l)}, D^{(l)} | \Theta, \xi) = \log \prod_{l=1}^{N} \prod_{i=1}^{n} \theta_{ijk}(l)$$

$$Q(\Theta, \Theta') = \sum_{l=1}^{N} \sum_{\{H\}} P(H^{(l)} | D^{(l)}, \Theta') \sum_{i=1}^{n} \log \theta_{ijk}(l)$$

$$= \sum_{l=1}^{N} \sum_{i=1}^{n} \sum_{\{H_i = X_i \cup pa(X_i)\}} P(H_i^{(l)} | D^{(l)}, \Theta') \log \theta_{ijk}(l)$$

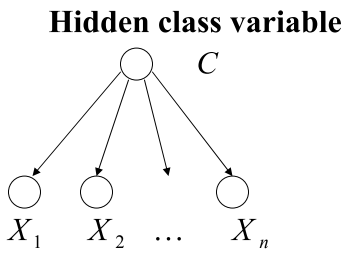
  – **The maximization of Q can be carried in the closed form**

    - 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:

- $P(\mathbf{X})$ is modeled using a Naïve Bayes model with hidden class variable
- Missing entries (values) for attributes in the dataset D



Attributes are independent given the class

EM for the Naïve Bayes

- **We can use EM to learn the parameters**

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

- **Parameters:**

  π_j prior on class j

  θ_{ijk} 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)

  $\delta_{ijk}^{\,l}$ 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

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

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

$$= \sum_{l=1}^{N} \sum_{j} (\delta_j^l \log \pi_j + \sum_{i} \sum_{k} \delta_{ijk}^l \log \theta_{ijk})$$

$$E_{H|D,\Theta'} \log P(H, D \mid \Theta, \xi) = \sum_{l=1}^{N} \sum_{j} (E_{H|D,\Theta'}(\delta_j^l) \log \pi_j + \sum_{i} \sum_{k} E_{H|D,\Theta'}(\delta_{ijk}^l) \log \theta_{ijk})$$

$$E_{H|D,\Theta'}(\delta_j^l) = p(C_l = j \mid D_l, \Theta')$$

$$E_{H|D,\Theta'}(\delta_{ijk}^l) = p(X_{il} = k, C_l = j \mid D_l, \Theta')$$

Substitutes 0,1 with expected value

EM for Naïve Bayes model

- Computing derivatives of $\boldsymbol{Q}$ for parameters and setting it to 0 we get:

$$\pi_j = \frac{\tilde{N}_j}{N} \qquad \theta_{ijk} = \frac{\tilde{N}_{ijk}}{\sum_{k=1}^{r_i} \tilde{N}_{ijk}}$$

$$\tilde{N}_j = \sum_{l=1}^{N} E_{H|D,\Theta'}(\delta_j^l) = \sum_{l=1}^{N} p(C_l = j \mid D_l, \Theta')$$

$$\tilde{N}_{ijk} = \sum_{l=1}^{N} E_{H|D,\Theta'}(\delta_{ijk}^l) = \sum_{l=1}^{N} p(X_{il} = k, C_l = j \mid D_l, \Theta')$$

- **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

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

$$\theta_{ijk} = \frac{\tilde{N}_{ijk}}{\sum_{k=1}^{r_i} \tilde{N}_{ijk}} \quad \longleftarrow \quad \text{Parameter value maximizing } \boldsymbol{Q}$$

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

may require inference

- **Again:**
  - Use expected counts instead of counts

Gaussian mixture model

Probability of occurrence of a data example $\boldsymbol{x}$ is modeled as

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

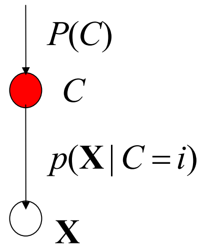
where

$p(C = i)$

= probability of a data point coming from class $C=i$

$p(\mathbf{x} \mid C = i) \approx N(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$

= class conditional density (modeled as a Gaussian) for class i



$P(C)$

C

$p(\mathbf{X} \mid C = i)$

$\mathbf{X}$

Remember: ***C* is hidden !!!!**

Generative Naïve Bayes classifier model

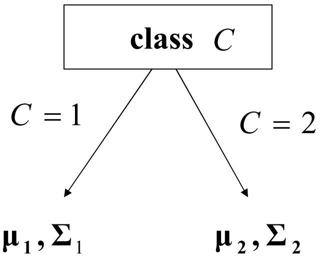
- Generative classifier model based on the Naïve Bayes
- Assume the class labels are known. The ML estimate is

$$N_i = \sum_{j:C_l = i} 1$$

$$\tilde{\pi}_i = \frac{N_i}{N}$$

$$\tilde{\boldsymbol{\mu}}_i = \frac{1}{N_i} \sum_{j:C_l = i} \mathbf{x}_j$$

$$\tilde{\boldsymbol{\Sigma}}_i = \frac{1}{N_i} \sum_{j:C_l = i} (\mathbf{x}_j - \boldsymbol{\mu}_i)(\mathbf{x}_j - \boldsymbol{\mu}_i)^T$$



class C

$C = 1$

$C = 2$

$\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1$

$\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2$

Gaussian mixture model

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

$$h_{il} = p(C_l = i \mid \mathbf{x}_l, \Theta') = \frac{p(C_l = i \mid \Theta')\, p(x_l \mid C_l = i, \Theta')}{\sum_{u=1}^{m} p(C_l = u \mid \Theta')\, p(x_l \mid C_l = u, \Theta')}$$

$$N_i = \sum_l h_{il}$$

Count replaced with the expected count

$$\tilde{\pi}_i = \frac{N_i}{N}$$

$$\tilde{\boldsymbol{\mu}}_i = \frac{1}{N_i} \sum_l h_{il} \mathbf{x}_j$$

$$\tilde{\boldsymbol{\Sigma}}_i = \frac{1}{N_i} \sum_l h_{il} (\mathbf{x}_j - \boldsymbol{\mu}_i)(\mathbf{x}_j - \boldsymbol{\mu}_i)^T$$

Gaussian mixture algorithm

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

Initialize means $\boldsymbol{\mu}_i$ 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)

Responsibility:

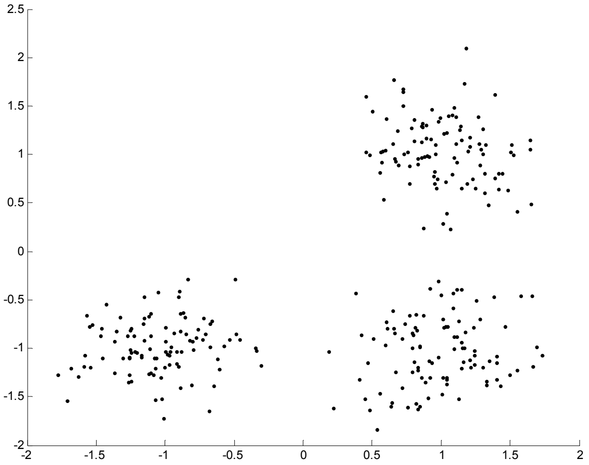
$$h_{il} = \frac{p(C_l = i \mid \Theta')\, p(x_l \mid C_l = i, \Theta')}{\sum_{u=1}^{m} p(C_l = u \mid \Theta')\, p(x_l \mid C_l = u, \Theta')}$$

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

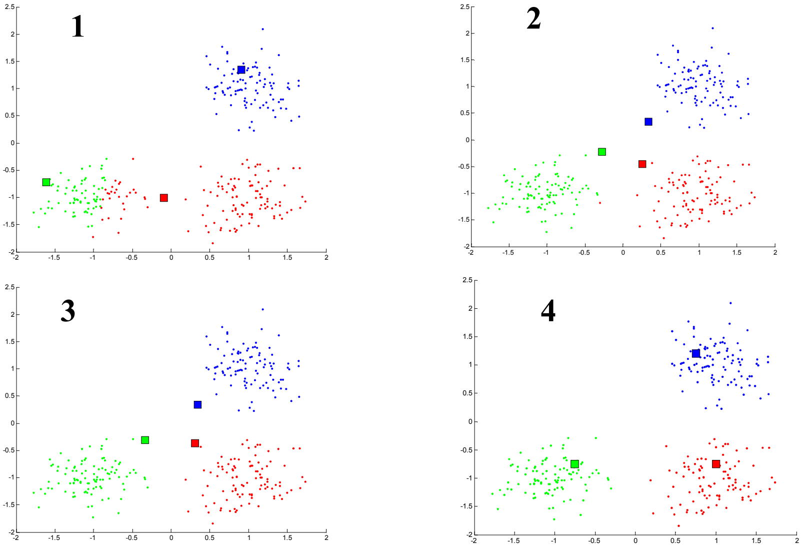
New mean:

$$\boldsymbol{\mu}_i = \frac{\sum_{l=1}^{N} h_{il} \mathbf{x}_l}{\sum_{l=1}^{N} h_{il}}$$

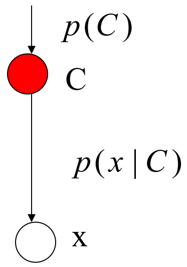
Gaussian mixture model - example



Gaussian mixture example



Gaussian mixture model. Gradient ascent.



- A set of parameters

$$\Theta = \{\pi_1, \pi_2, .. \pi_m, \mu_1, \mu_2, ... \mu_m\}$$

Assume unit variance terms and fixed priors

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

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

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

$$\frac{\partial l(\Theta)}{\partial \mu_i} = \sum_{l=1}^{N} h_{il}(x_l - \mu_i)$$

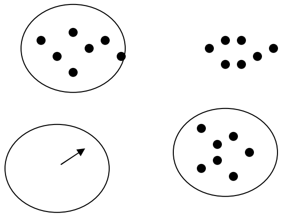
- easy on-line update

EM versus gradient ascent

Gradient ascent

$$\mu_i \leftarrow \mu_i + \alpha \sum_{l=1}^{N} h_{il}(x_l - \mu_i)$$

Learning rate

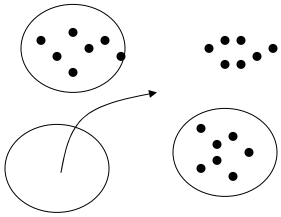


Small pull towards distant uncovered data

EM

$$\mu_i \leftarrow \frac{\sum_{l=1}^{N} h_{il} \mathbf{x}_l}{\sum_{l=1}^{N} h_{il}}$$

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

$$h_{il} = \frac{p(C_l = i \mid \Theta') p(x_l \mid C_l = i, \Theta')}{\sum_{u=1}^{m} p(C_l = u \mid \Theta') p(x_l \mid C_l = u, \Theta')}$$

K- Means

- Only the closest Gaussian is made responsible for a point

$h_{il} = 1$ If i is the closest Gaussian

$h_{il} = 0$ Otherwise

Re-estimation of means

$$\boldsymbol{\mu}_i = \frac{\sum_{l=1}^{N} h_{il} \mathbf{x}_l}{\sum_{l=1}^{N} h_{il}}$$

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

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

K-means algorithm

- **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)