CS 2750 Machine Learning
Lecture 20

Learning with hidden variables and missing values. Expectation maximization (EM)

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** $D = \{D_1, D_2, .., D_N\}$
  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 $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$ are related through hidden variables

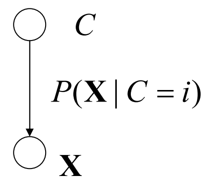
Why to add hidden variables?

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

Example:

- Latent variable models
  - hidden classes (categories)

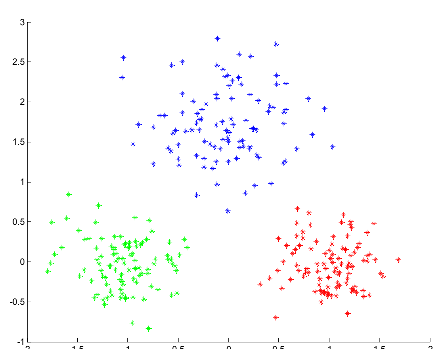
Hidden class variable



Hidden variable model. Example.

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

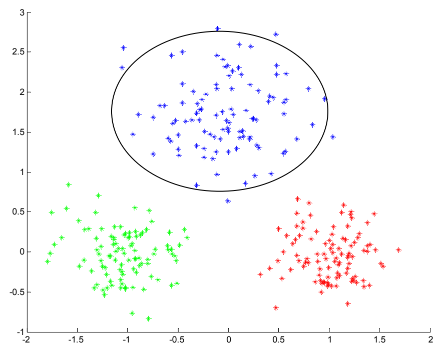
Observed data



Hidden variable model

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

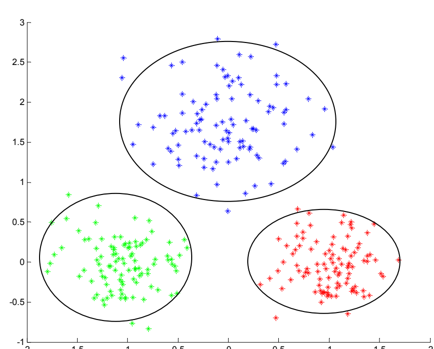
Observed data



Hidden variable model

- We want to represent a model of a population in a two dimensional space $\mathbf{X} = \{X_1, X_2\}$

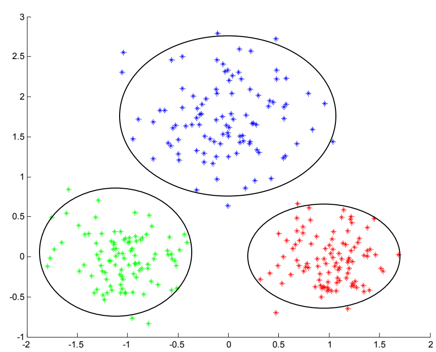
Observed data



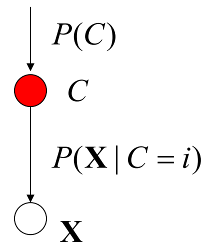
Hidden variable model

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

Observed data



Model : 3 Gaussians with a hidden class variable



Mixture of Gaussians

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

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

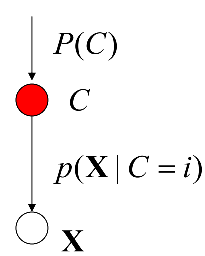
where

$p(C = i)$

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

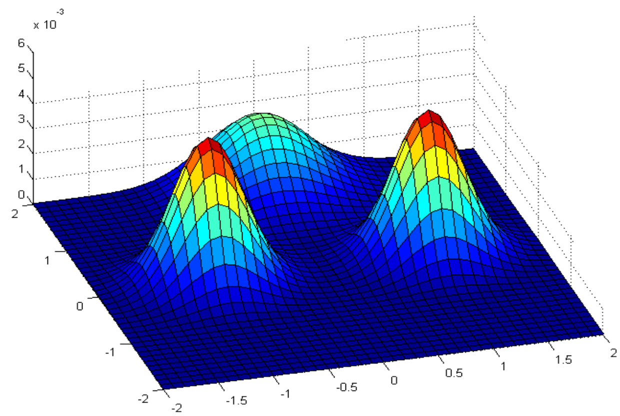
$p(\mathbf{x} \mid C = i) \approx N(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$

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



Mixture of Gaussians

- Density function for the Mixture of Gaussians model



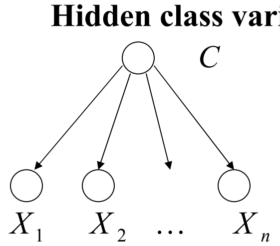
Naïve Bayes with a hidden class variable

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

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 $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$

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

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

Missing value of x_2^i

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

Missing values of x_1^i, x_2^i

Etc.

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

Density estimation

Goal: Find the set of parameters $\hat{\Theta}$

Estimation criteria:

- **ML** $\max_{\mathbf{\Theta}} p(D \mid \mathbf{\Theta}, \xi)$
- **Bayesian** $p(\mathbf{\Theta} \mid D, \xi)$

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

Stop when no improvement possible

EM

Let H – be a set of all variables with hidden or missing values

Derivation

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

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

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

Log-likelihood of data

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

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

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

Log-likelihood of data

EM algorithm

Algorithm (general formulation)

Initialize parameters Θ

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 Θ $(\Theta = \Theta')$

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**

$$l(\Theta) = Q(\Theta \mid \Theta') + H(\Theta \mid \Theta')$$

Difference in log-likelihoods (current and next step)

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

Subexpression $H(\Theta \mid \Theta') - H(\Theta' \mid \Theta') \geq 0$

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 !!!

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

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

EM algorithm

Difference in log-likelihoods

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

$$l(\Theta) - l(\Theta') \geq Q(\Theta \mid \Theta') - Q(\Theta' \mid \Theta')$$

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:

- For Bayesian belief networks

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

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

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

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

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

  - **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

EM for BBNs

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

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

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