

CS 2750 Machine Learning Lecture 18

Learning BBNs with hidden variables and missing values. Expectation maximization.

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

CS 2750 Machine Learning

Learning probability distribution

Basic learning settings:

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

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

Assumptions considered so far:

- Known parameterizations
- No hidden variables
- No-missing values

CS 2750 Machine Learning

Hidden variables

Modeling assumption:

Variables $\mathbf{X} = \{X_1, X_2, \dots, X_n\}$

- Additional variables are hidden – never observed in data

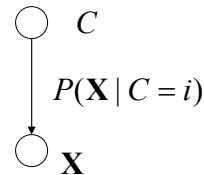
Why to add hidden variables?

- **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



CS 2750 Machine Learning

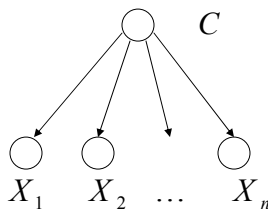
Naïve Bayes with a hidden class variable

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

CS 2750 Machine Learning

Missing values

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

- **Data** $D = \{D_1, D_2, \dots, D_N\}$

- **But some values are missing**

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

Missing value of x_2^i

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

Missing values of x_1^{i+1}, x_2^{i+1}

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** $\max_{\Theta} p(D | \Theta, \xi)$
- **Bayesian** $p(\Theta | D, \xi)$

Optimization methods for ML: gradient-ascent, conjugate gradient, Newton-Rhapon, 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 Θ'
2. **Maximization step.** Compute the new estimates of Θ for the completed data

Stop when no improvement possible

EM

Let H – be a set 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 Θ'

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

$$\underbrace{\log P(D | \Theta, \xi)}_{\text{Log-likelihood of data}} = Q(\Theta | \Theta') + H(\Theta | \Theta')$$

Log-likelihood of data

EM algorithm

Algorithm (general formulation)

Initialize parameters Θ

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 Θ ($\Theta = \Theta'$)

Questions: Why this leads to the ML estimate ?

What is the advantage of the algorithm?

CS 2750 Machine Learning

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)

$$KL(P | R) = \sum_i P_i \log \frac{P_i}{R_i} \geq 0 \quad \text{Is always positive !!!}$$

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

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

CS 2750 Machine Learning

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)

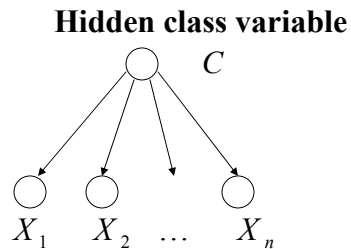
$$Q(\Theta | \Theta') = E_{H|D, \Theta'} \log P(H, D | \Theta, \xi)$$

- 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:

- $P(\mathbf{X})$ 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

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

- **Parameters:**

π_j prior on class j

θ_{ijk} probability of an attribute i having value k given class j

- **Indicator variables:**

δ_j^l for example l , the class is j ; if true (=1) else false (=0)

δ_{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

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

$$\begin{aligned} \log P(H, D | \Theta, \xi) &= \log \prod_{l=1}^N \prod_j \pi_j^{\delta_j^l} \prod_i \prod_k \theta_{ijk}^{\delta_{ijk}^l} \\ &= \sum_{l=1}^N \sum_j (\delta_j^l \log \pi_j + \sum_i \sum_k \delta_{ijk}^l \log \theta_{ijk}) \end{aligned}$$

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

$$E_{H|D, \Theta'}(\delta_j^l) = p(C_l = j | D_l, \Theta') \quad \text{Substitutes 0,1}$$

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

EM for Naïve Bayes model

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

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

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

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

- 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

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

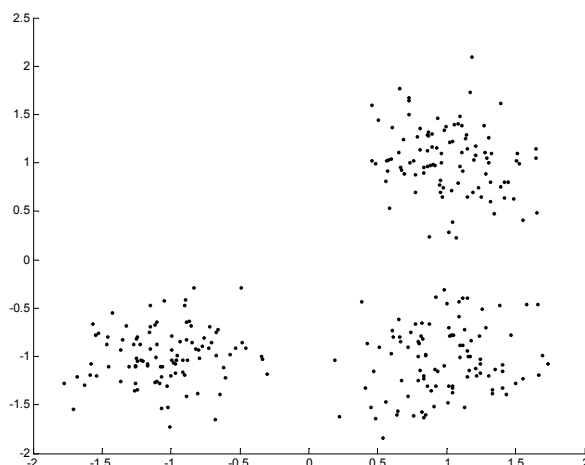
$$\theta_{ijk} = \frac{\tilde{N}_{ijk}}{\sum_{k=1}^{r_i} \tilde{N}_{ijk}} \leftarrow \text{Parameter value maximizing } Q$$

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

may require inference

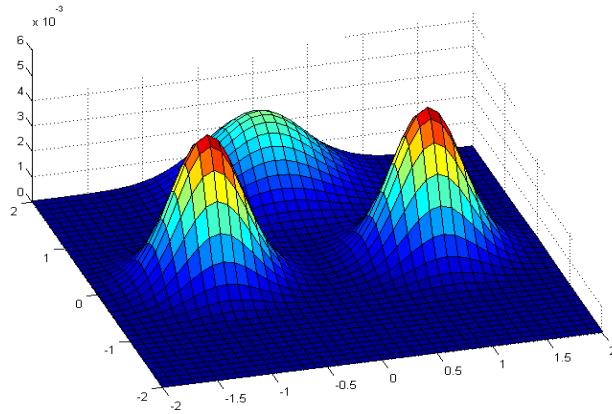
- Again:**
 - Use expected counts instead of counts

Gaussian mixture model



Mixture of Gaussians

- Density function for the Mixture of Gaussians model



CS 2750 Machine Learning

Gaussian mixture model

Probability of 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

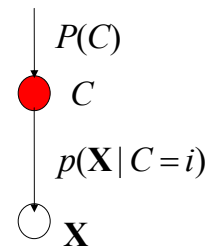
$$p(C = i)$$

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

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

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

Special feature: C is hidden !!!!



CS 2750 Machine Learning

Generative Naïve Bayes classifier model

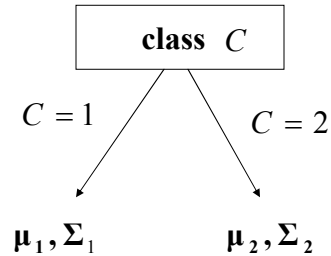
- Generative classifier model based on the Naïve Bayes
- Assume the class labels are known. The ML estimate is

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

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

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

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



CS 2750 Machine Learning

Gaussian mixture model

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

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

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

Count replaced with the expected count

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

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

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

CS 2750 Machine Learning

Gaussian mixture algorithm

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

- **Algorithm:**

Initialize means μ_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:
$$h_{il} = \frac{p(C_l = i | \Theta') p(x_l | C_l = i, \Theta')}{\sum_{u=1}^m p(C_l = u | \Theta') p(x_l | C_l = u, \Theta')}$$

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

New mean:
$$\mu_i = \frac{\sum_{l=1}^N h_{il} x_l}{\sum_{l=1}^N h_{il}}$$

Gaussian mixture model. Gradient ascent.

- A set of parameters

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

Assume unit variance terms and fixed priors

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

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

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

$$\frac{\partial l(\Theta)}{\partial \mu_i} = \sum_{l=1}^N h_{il} (x_l - \mu_i) \quad \text{- very easy on-line update}$$

