Probabilistic Graphical Models

Lecture 6: Learning and Inference. EM Algorithm

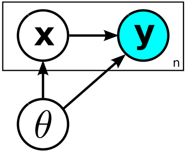
Volkan Cevher, Matthias Seeger
Ecole Polytechnique Fédérale de Lausanne

17/10/2011

Outline

Learning is Estimation

$$P(\boldsymbol{y}, \boldsymbol{x}, \theta) = P(\boldsymbol{y}|\boldsymbol{x}, \theta)P(\boldsymbol{x}|\theta)P(\theta)$$



$\boldsymbol{y}$ Observed variable

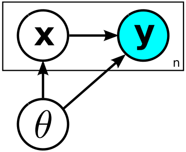
$\boldsymbol{x}$ Latent variable (nuisance)

θ Parameters (latent query, “higher up”)

- Learning: What is a (single) good value for θ? [argmax . . .]
  For which θ does model fit data $D = \{\boldsymbol{y}_1, \dots, \boldsymbol{y}_n\}$? $\Rightarrow$ Estimation

Learning is Estimation

$$P(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{\theta}) = P(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta})P(\boldsymbol{x}|\boldsymbol{\theta})P(\boldsymbol{\theta})$$

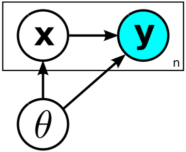


$\boldsymbol{y}$ Observed variable
$\boldsymbol{x}$ Latent variable (nuisance)
$\boldsymbol{\theta}$ Parameters (latent query, "higher up")

- Learning: What is a (single) good value for $\boldsymbol{\theta}$? [argmax ...]
  For which $\boldsymbol{\theta}$ does model fit data $D = \{\boldsymbol{y}_1, \dots, \boldsymbol{y}_n\}$? $\Rightarrow$ Estimation
  - Maximum likelihood (ML) estimation: $\hat{\boldsymbol{\theta}} = \operatorname{argmax} \log P(D|\boldsymbol{\theta})$
  - Maximum a posteriori (MAP) estimation:
    $\hat{\boldsymbol{\theta}} = \operatorname{argmax} \log P(\boldsymbol{\theta}|D) = \operatorname{argmax}(\log P(D|\boldsymbol{\theta}) + \log P(\boldsymbol{\theta}))$

Learning is Estimation

$$P(\boldsymbol{y},\boldsymbol{x},\theta) = P(\boldsymbol{y}|\boldsymbol{x},\theta)P(\boldsymbol{x}|\theta)P(\theta)$$



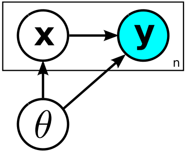
$\boldsymbol{y}$ Observed variable
$\boldsymbol{x}$ Latent variable (nuisance)
θ Parameters (latent query, "higher up")

- Learning: What is a (single) good value for θ? [argmax ...]
  For which θ does model fit data $D = \{\boldsymbol{y}_1, \dots, \boldsymbol{y}_n\}$? $\Rightarrow$ Estimation
  - Maximum likelihood (ML) estimation: $\hat{\theta} = \operatorname{argmax} \log P(D|\theta)$
  - Maximum a posteriori (MAP) estimation:
    $\hat{\theta} = \operatorname{argmax} \log P(\theta|D) = \operatorname{argmax}(\log P(D|\theta) + \log P(\theta))$
- Inference: What is posterior $P(\theta|D)$? $[\int \dots]$
  - Range / shape of "good values" mass
  - Uncertainty in estimates

  Terms like "MAP inference": Just wrong

Learning is Estimation

$$P(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{\theta}) = P(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta})P(\boldsymbol{x}|\boldsymbol{\theta})P(\boldsymbol{\theta})$$



$\boldsymbol{y}$ Observed variable
$\boldsymbol{x}$ Latent variable (nuisance)
$\boldsymbol{\theta}$ Parameters (latent query, "higher up")

- Learning: What is a (single) good value for $\boldsymbol{\theta}$? [argmax . . .]
  For which $\boldsymbol{\theta}$ does model fit data $D = \{\boldsymbol{y}_1, \dots, \boldsymbol{y}_n\}$? $\Rightarrow$ Estimation
    - Maximum likelihood (ML) estimation: $\hat{\boldsymbol{\theta}} = \operatorname{argmax} \log P(D|\boldsymbol{\theta})$
    - Maximum a posteriori (MAP) estimation:
      $\hat{\boldsymbol{\theta}} = \operatorname{argmax} \log P(\boldsymbol{\theta}|D) = \operatorname{argmax}(\log P(D|\boldsymbol{\theta}) + \log P(\boldsymbol{\theta}))$
- Inference: What is posterior $P(\boldsymbol{\theta}|D)$? $[\int \dots]$
    - Range / shape of "good values" mass
    - Uncertainty in estimates

  Terms like "MAP inference": Just wrong
- Still: We'll need inference $[P(\boldsymbol{x}|\boldsymbol{y}, \boldsymbol{\theta})]$ for learning

Why Learning can be Hard

- Data from $N(\boldsymbol{y}|\boldsymbol{\mu}, \sigma^2\boldsymbol{I})$. Learn mean $\boldsymbol{\mu}$ $\Rightarrow$ That's not hard. Why?

Why Learning can be Hard

- Data from $N(\boldsymbol{y}|\boldsymbol{\mu},\sigma^2\boldsymbol{I})$. Learn mean $\boldsymbol{\mu}$ $\Rightarrow$ That's not hard. Why?
    - Model directed graph. CPTs: Nice form (Gaussian)
    - No latent variables except parameters

  $\Rightarrow$ No inference required
- Learning gets hard if you need inference: Nice-form distributions become nasty through marginalization
    - Latent nuisance variables
    - Undirected models (MRFs)

Why Learning can be Hard

- Data from $N(\boldsymbol{y}|\boldsymbol{\mu},\sigma^2\boldsymbol{I})$. Learn mean $\boldsymbol{\mu}$ $\Rightarrow$ That's not hard. Why?
  - Model directed graph. CPTs: Nice form (Gaussian)
  - No latent variables except parameters

  $\Rightarrow$ No inference required
- Learning gets hard if you need inference: Nice-form distributions become nasty through marginalization
  - Latent nuisance variables
  - Undirected models (MRFs)
- Marginalization creates log partition functions

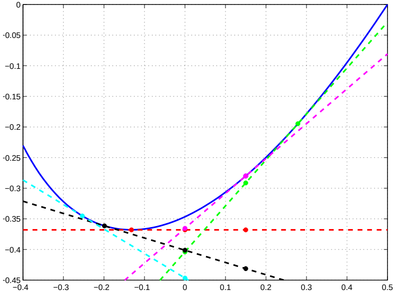
$$\text{Bayes net, latent variables} \qquad\qquad \text{Markov random field}$$
$$\underbrace{\log P(\boldsymbol{y}|\boldsymbol{\theta})}_{\text{coupled}} = \log \int \underbrace{P(\boldsymbol{y},\boldsymbol{x}|\boldsymbol{\theta})}_{\text{decoupled}}\, d\boldsymbol{x} \qquad \log Z = \log \sum_{\boldsymbol{x}} \prod_j \Phi_j(\boldsymbol{x}_{C_j})$$

$\Rightarrow$ Optimization of log partition functions needs inference

Convex Functions. Jensen's Inequality

- Convex set $\mathcal{C}$:
  $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{C},\ \lambda \in [0,1]$
  $\Rightarrow \lambda \boldsymbol{x}_1 + (1-\lambda)\boldsymbol{x}_2 \in \mathcal{C}$
- Convex function $f : \mathcal{C} \to \mathbb{R}$:
  $f(\lambda \boldsymbol{x}_1 + (1-\lambda)\boldsymbol{x}_2)$
  $\leq \lambda f(\boldsymbol{x}_1) + (1-\lambda) f(\boldsymbol{x}_2)$
- Same concept: f convex $\Leftrightarrow$
  $\mathrm{epi}(f) := \{(\boldsymbol{x}, y) \,|\, f(\boldsymbol{x}) \leq y\}$ convex

F3



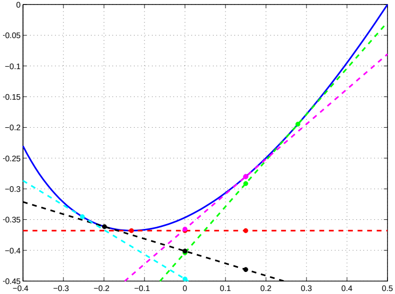
Dig for yourself about convexity:

- Boyd, Vandenberghe: Convex Optimization (2004)

[http://www.stanford.edu/∼boyd/cvxbook/]

Convex Functions. Jensen's Inequality

- Convex set $\mathcal{C}$:
  $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{C}, \lambda \in [0, 1]$
  $\Rightarrow \lambda \boldsymbol{x}_1 + (1 - \lambda)\boldsymbol{x}_2 \in \mathcal{C}$
- Convex function $f : \mathcal{C} \to \mathbb{R}$:
  $f(\lambda \boldsymbol{x}_1 + (1 - \lambda)\boldsymbol{x}_2)$
  $\leq \lambda f(\boldsymbol{x}_1) + (1 - \lambda) f(\boldsymbol{x}_2)$
- Same concept: f convex $\Leftrightarrow$
  $\mathrm{epi}(f) := \{(\boldsymbol{x}, y) \,|\, f(\boldsymbol{x}) \leq y\}$ convex



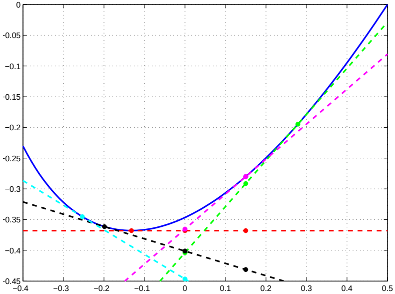
- Equivalent: For each $\boldsymbol{x}_0 \in \mathcal{C}$, there exists $\boldsymbol{u}$ s.t.

F3b

$$f(\boldsymbol{x}) \geq \boldsymbol{u}^T(\boldsymbol{x} - \boldsymbol{x}_0) + f(\boldsymbol{x}_0) \quad \text{for all } \boldsymbol{x} \in \mathcal{C}$$

Convex Functions. Jensen's Inequality

- Convex set $\mathcal{C}$:
  $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{C},\ \lambda \in [0,1]$
  $\Rightarrow \lambda \boldsymbol{x}_1 + (1-\lambda)\boldsymbol{x}_2 \in \mathcal{C}$
- Convex function $f : \mathcal{C} \to \mathbb{R}$:
  $f(\lambda \boldsymbol{x}_1 + (1-\lambda)\boldsymbol{x}_2)$
  $\leq \lambda f(\boldsymbol{x}_1) + (1-\lambda) f(\boldsymbol{x}_2)$
- Same concept: f convex $\Leftrightarrow$
  $\mathrm{epi}(f) := \{(\boldsymbol{x}, y) \mid f(\boldsymbol{x}) \leq y\}$ convex



- Equivalent: For each $\boldsymbol{x}_0 \in \mathcal{C}$, there exists $\boldsymbol{u}$ s.t.

$$f(\boldsymbol{x}) \geq \boldsymbol{u}^T(\boldsymbol{x} - \boldsymbol{x}_0) + f(\boldsymbol{x}_0) \quad \text{for all } \boldsymbol{x} \in \mathcal{C}$$

- Jensen's inequality: $f : \mathcal{C} \to \mathbb{R}$ convex, P distribution over $\mathcal{C}$

$$\mathrm{E}_P[f(\boldsymbol{x})] \geq f(\mathrm{E}_P[\boldsymbol{x}])$$

Bounding Log Partition Functions

Recall the problem:

$$\max_{\boldsymbol{\theta}} \log \int \Phi(\boldsymbol{x}|\boldsymbol{\theta})\, d\boldsymbol{x}, \quad \Phi(\boldsymbol{x}|\boldsymbol{\theta}) = \prod_j \Phi_j(\boldsymbol{x}_{C_j}|\boldsymbol{\theta})$$

- $t \mapsto -\log(t)$ convex function: For positive f: $\log \mathrm{E}_Q[f(\boldsymbol{x})] \geq \mathrm{E}_Q[\log f(\boldsymbol{x})]$ (by Jensen)

Bounding Log Partition Functions

Recall the problem:

$$\max_{\boldsymbol{\theta}} \log \int \Phi(\boldsymbol{x}|\boldsymbol{\theta})\, d\boldsymbol{x}, \quad \Phi(\boldsymbol{x}|\boldsymbol{\theta}) = \prod_j \Phi_j(\boldsymbol{x}_{C_j}|\boldsymbol{\theta})$$

- $t \mapsto -\log(t)$ convex function: For positive f: $\log \mathrm{E}_Q[f(\boldsymbol{x})] \geq \mathrm{E}_Q[\log f(\boldsymbol{x})]$ (by Jensen)
- Variational mean field inequality F4

$$\log \int \Phi(\boldsymbol{x})\, d\boldsymbol{x} \geq \sup_Q \mathrm{E}_Q \left[\log \frac{\Phi(\boldsymbol{x})}{Q(\boldsymbol{x})} \right]$$

Bounding Log Partition Functions

Recall the problem:

$$\max_{\boldsymbol{\theta}} \log \int \Phi(\boldsymbol{x}|\boldsymbol{\theta})\, d\boldsymbol{x}, \quad \Phi(\boldsymbol{x}|\boldsymbol{\theta}) = \prod_j \Phi_j(\boldsymbol{x}_{C_j}|\boldsymbol{\theta})$$

- $t \mapsto -\log(t)$ convex function: For positive f: $\log \mathrm{E}_Q[f(\boldsymbol{x})] \geq \mathrm{E}_Q[\log f(\boldsymbol{x})]$ (by Jensen)
- Variational mean field inequality

$$\log \int \Phi(\boldsymbol{x})\, d\boldsymbol{x} \geq \sup_Q \mathrm{E}_Q\left[\log \frac{\Phi(\boldsymbol{x})}{Q(\boldsymbol{x})}\right]$$

- Pushing the log inside

$$\begin{array}{ll} \log \int \prod_j \dots & \text{Hard, no decoupling} \\ \int \log \prod_j \dots = \int \sum_j \log \dots & \text{Decoupling : Can be much easier} \end{array}$$

Towards Expectation Maximization

$$\log \int \Phi(\boldsymbol{x})\, d\boldsymbol{x} \;\geq\; \sup_Q \mathrm{E}_Q \left[\log \frac{\Phi(\boldsymbol{x})}{Q(\boldsymbol{x})} \right]$$

Here is a very simple question: What is the best $Q(\boldsymbol{x})$ I could choose?

F5

Towards Expectation Maximization

$$\log \int \Phi(\boldsymbol{x})\,d\boldsymbol{x} = \sup_Q \mathrm{E}_Q\left[\log \frac{\Phi(\boldsymbol{x})}{Q(\boldsymbol{x})}\right]$$

Here is a very simple question: What is the best $Q(\boldsymbol{x})$ I could choose?
$Q(\boldsymbol{x}) = \Phi(\boldsymbol{x})/Z$: The posterior in this situation

Expectation Maximization (Full Generality)

F5a

Goal: Maximize $\log \int \Phi(\boldsymbol{x}|\theta)\,d\boldsymbol{x}$. Iterate:

- Expectation (E step): Tightest lower bound
  $Q(\boldsymbol{x}) \leftarrow \Phi(\boldsymbol{x}|\theta)/Z(\theta)$
- Maximization (M step): Maximize lower bound
  $\theta \leftarrow \operatorname{argmax} \mathrm{E}_Q[\log \Phi(\boldsymbol{x}|\theta)]$ for fixed Q

EM Algorithm for Gaussian Mixtures

Gaussian mixture model: $P(\boldsymbol{y}|x) = N(\boldsymbol{\mu}_x, \boldsymbol{I})$, $P(x = k) = 1/K$

Observed data: $\boldsymbol{y}_1, \dots, \boldsymbol{y}_n \in \mathbb{R}^d$
Latent indicators: $x_1, \dots, x_n \in \{1, \dots, K\}$

How to find cluster centers $\boldsymbol{\mu}_k$?

EM Algorithm for Gaussian Mixtures

Gaussian mixture model: $P(\boldsymbol{y}|x) = N(\boldsymbol{\mu}_x, \boldsymbol{I})$, $P(x = k) = 1/K$

Observed data: $\boldsymbol{y}_1, \dots, \boldsymbol{y}_n \in \mathbb{R}^d$
Latent indicators: $x_1, \dots, x_n \in \{1, \dots, K\}$

How to find cluster centers $\boldsymbol{\mu}_k$?

Translation

general $\to$ particular
$\Phi(\boldsymbol{x})$ $\to \prod_i P(\boldsymbol{y}_i|x_i)P(x_i)$ [joint likelihood]
$\boldsymbol{\theta}$ $\to \boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_K$ [cluster centers]
$Q(\boldsymbol{x})$ $\to Q(\boldsymbol{x}) = \prod_i Q(x_i)$
$Z(\boldsymbol{\theta})$ $\to Z = \prod_i Z_i$, $Z_i = \sum_{x_i} P(\boldsymbol{y}_i|x_i)P(x_i)$

Note: Decoupling

$$\log \Phi(\boldsymbol{x}) = \sum_i \log[P(\boldsymbol{y}_i|x_i)P(x_i)], \quad \log Z = \sum_i \log Z_i$$

EM Algorithm for Gaussian Mixtures

Gaussian mixture model: $P(\boldsymbol{y}|x) = N(\boldsymbol{\mu}_x, \boldsymbol{I})$, $P(x = k) = 1/K$

Observed data: $\boldsymbol{y}_1, \dots, \boldsymbol{y}_n \in \mathbb{R}^d$
Latent indicators: $x_1, \dots, x_n \in \{1, \dots, K\}$

How to find cluster centers $\boldsymbol{\mu}_k$?

Iterate:

1. Expectation: Posterior distribution for each datapoint

$$Q(x_i = k) \leftarrow P(x_i = k|\boldsymbol{y}_i)$$

2. Maximization: Posterior average of all datapoints

$$\boldsymbol{\mu}_k \leftarrow n_k^{-1} \sum\nolimits_i Q(x_i = k)\boldsymbol{y}_i = \text{argmax} \sum\nolimits_i Q(x_i = k) \log P(\boldsymbol{y}_i|x_i = k),$$

$n_k = \sum_i Q(x_i = k)$. Posterior weighted maximum likelihood

EM Algorithm for Bayesian Networks

$$P(\boldsymbol{x}) = \prod_j P(x_j | \boldsymbol{x}_{\pi_j}, \boldsymbol{\theta}_j), \quad \pi_j : \text{parents of node } j, \quad j = 1, \dots, J$$

Parameters $\boldsymbol{\theta}_j$: CPT for $x_j | \boldsymbol{x}_{\pi_j}$. Data: $D = \{\boldsymbol{x}^{(i)} = (x_j^{(i)})\}$, $i = 1, \dots, n$.
In each $\boldsymbol{x}^{(i)}$: Coefficients can be missing

- All $\boldsymbol{x}^{(i)}$ complete: Match CPTs to empirical averages (counts) $\Rightarrow$ No EM needed

EM Algorithm for Bayesian Networks

$$P(\boldsymbol{x}) = \prod_j P(x_j | \boldsymbol{x}_{\pi_j}, \boldsymbol{\theta}_j), \quad \pi_j : \text{parents of node } j, \quad j = 1, \dots, J$$

Parameters $\boldsymbol{\theta}_j$: CPT for $x_j | \boldsymbol{x}_{\pi_j}$. Data: $D = \{\boldsymbol{x}^{(i)} = (x_j^{(i)})\}$, $i = 1, \dots, n$.
In each $\boldsymbol{x}^{(i)}$: Coefficients can be missing

- All $\boldsymbol{x}^{(i)}$ complete: Match CPTs to empirical averages (counts) $\Rightarrow$ No EM needed
- Partially observed $\boldsymbol{x}^{(i)}$: Your exercise sheet!

Log Partition Function: A Closer Look

Log partition function of $P(\boldsymbol{x}) = \Phi(\boldsymbol{x})/Z$: F7

$$\log Z = \log \sum\nolimits_{\boldsymbol{x}} e^{\sum_j \Psi_j(\boldsymbol{x}_{C_j})}, \quad \Psi(\boldsymbol{x}) = \sum\nolimits_j \Psi_j(\boldsymbol{x}_{C_j}) = \log \Phi(\boldsymbol{x})$$

Note: Can have $\sum_{\boldsymbol{x}} \to \int \dots d\boldsymbol{x}$

Log Partition Function: A Closer Look

Log partition function of $P(\boldsymbol{x}) = \Phi(\boldsymbol{x})/Z$:

$$\log Z = \log \sum_{\boldsymbol{x}} e^{\sum_j \Psi_j(\boldsymbol{x}_{C_j})}, \quad \Psi(\boldsymbol{x}) = \sum_j \Psi_j(\boldsymbol{x}_{C_j}) = \log \Phi(\boldsymbol{x})$$

Note: Can have $\sum_{\boldsymbol{x}} \to \int \dots d\boldsymbol{x}$

1. Moment-generating: F7a

$$\nabla_{\theta} \log Z = \mathrm{E}_P[\nabla_{\theta} \Psi(\boldsymbol{x})] = \sum_j \mathrm{E}_P[\nabla_{\theta} \Psi_j(\boldsymbol{x}_{C_j})]$$

Log Partition Function: A Closer Look

Log partition function of $P(\boldsymbol{x}) = \Phi(\boldsymbol{x})/Z$:

$$\log Z = \log \sum_{\boldsymbol{x}} e^{\sum_j \Psi_j(\boldsymbol{x}_{C_j})}, \quad \Psi(\boldsymbol{x}) = \sum_j \Psi_j(\boldsymbol{x}_{C_j}) = \log \Phi(\boldsymbol{x})$$

Note: Can have $\sum_{\boldsymbol{x}} \to \int \dots d\boldsymbol{x}$

1. Moment-generating:

$$\nabla_{\boldsymbol{\theta}} \log Z = \mathrm{E}_P[\nabla_{\boldsymbol{\theta}} \Psi(\boldsymbol{x})] = \sum_j \mathrm{E}_P[\nabla_{\boldsymbol{\theta}} \Psi_j(\boldsymbol{x}_{C_j})]$$

2. Convex: $(v_{\boldsymbol{x}}) \mapsto \log \sum_{\boldsymbol{x}} e^{v_{\boldsymbol{x}}}$ F7b

Log Partition Function: A Closer Look

Log partition function of $P(\boldsymbol{x}) = \Phi(\boldsymbol{x})/Z$:

$$\log Z = \log \sum_{\boldsymbol{x}} e^{\sum_j \Psi_j(\boldsymbol{x}_{C_j})}, \quad \Psi(\boldsymbol{x}) = \sum_j \Psi_j(\boldsymbol{x}_{C_j}) = \log \Phi(\boldsymbol{x})$$

Note: Can have $\sum_{\boldsymbol{x}} \to \int \dots d\boldsymbol{x}$

1. Moment-generating:

$$\nabla_{\boldsymbol{\theta}} \log Z = \mathrm{E}_P[\nabla_{\boldsymbol{\theta}} \Psi(\boldsymbol{x})] = \sum_j \mathrm{E}_P[\nabla_{\boldsymbol{\theta}} \Psi_j(\boldsymbol{x}_{C_j})]$$

2. Convex: $(v_{\boldsymbol{x}}) \mapsto \log \sum_{\boldsymbol{x}} e^{v_{\boldsymbol{x}}}$

Consequence of (1):

- Computing $\nabla_{\boldsymbol{\theta}} \log Z$: Exactly same as E step (posterior moments over clique marginals)
- Can use any gradient-based optimizer instead of EM

F7c

Learning Markov Random Fields

$$P(\boldsymbol{x}) = Z^{-1} e^{\Psi(\boldsymbol{x})}, \quad \Psi(\boldsymbol{x}) = \sum_j \Psi_j(\boldsymbol{x}_{C_j})$$

Note: All $\boldsymbol{x}$ observed here $\to \tilde{\boldsymbol{x}}$

- Maximum likelihood:

$$\max_\theta \log P(\tilde{\boldsymbol{x}}) = \max_\theta \left(\Psi(\tilde{\boldsymbol{x}}) - \log Z\right)$$

Minus $\log Z$: EM won't do

Learning Markov Random Fields

$$P(\boldsymbol{x}) = Z^{-1} e^{\Psi(\boldsymbol{x})}, \quad \Psi(\boldsymbol{x}) = \sum_j \Psi_j(\boldsymbol{x}_{C_j})$$

Note: All $\boldsymbol{x}$ observed here $\rightarrow \tilde{\boldsymbol{x}}$

- Maximum likelihood:

$$\max_\theta \log P(\tilde{\boldsymbol{x}}) = \max_\theta \left(\Psi(\tilde{\boldsymbol{x}}) - \log Z\right)$$

  Minus $\log Z$: EM won't do

- Gradient-based optimization: F8

$$\nabla_\theta \log P(\tilde{\boldsymbol{x}}) = \sum_j \left(\nabla_\theta \Psi_j(\tilde{\boldsymbol{x}}_{C_j}) - \mathrm{E}_P[\nabla_\theta \Psi_j(\boldsymbol{x}_{C_j})] \right)$$

Learning Markov Random Fields

$$P(\boldsymbol{x}) = Z^{-1} e^{\Psi(\boldsymbol{x})}, \quad \Psi(\boldsymbol{x}) = \sum_j \Psi_j(\boldsymbol{x}_{C_j})$$

Note: All $\boldsymbol{x}$ observed here $\rightarrow \tilde{\boldsymbol{x}}$

- Maximum likelihood:

$$\max_{\boldsymbol{\theta}} \log P(\tilde{\boldsymbol{x}}) = \max_{\boldsymbol{\theta}} \left(\Psi(\tilde{\boldsymbol{x}}) - \log Z\right)$$

  Minus $\log Z$: EM won't do

- Gradient-based optimization:

$$\nabla_{\boldsymbol{\theta}} \log P(\tilde{\boldsymbol{x}}) = \sum_j \left(\nabla_{\boldsymbol{\theta}} \Psi_j(\tilde{\boldsymbol{x}}_{C_j}) - \mathrm{E}_P[\nabla_{\boldsymbol{\theta}} \Psi_j(\boldsymbol{x}_{C_j})]\right)$$

- Log-linear models: Surprisingly often, $\Psi_j(\boldsymbol{x}_{C_j}) = \boldsymbol{\theta}^T \boldsymbol{f}_j(\boldsymbol{x}_{C_j})$
  - $\nabla_{\boldsymbol{\theta}} \log P(\tilde{\boldsymbol{x}}) = \sum_j (\boldsymbol{f}_j(\tilde{\boldsymbol{x}}_{C_j}) - \mathrm{E}_P[\boldsymbol{f}_j(\boldsymbol{x}_{C_j})])$
  - Convex optimization problem

F8a

Iterative Proportional Fitting

Log-linear Markov random field, separable parameters:

$$P(\boldsymbol{x}) = Z^{-1} e^{\sum_j \boldsymbol{\theta}_j^T \boldsymbol{f}_j(\boldsymbol{x}_{C_j})}, \quad \boldsymbol{\theta} = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \dots)$$

- Assume: For each j, $Q(\boldsymbol{x}_{C_j})$: Can easily find $\Delta\boldsymbol{\theta}_j$ s.t. $\mathrm{E}_{Q_\Delta}[\boldsymbol{f}_j(\boldsymbol{x}_{C_j})] = \boldsymbol{f}_j(\tilde{\boldsymbol{x}}_{C_j})$, where $Q_\Delta(\boldsymbol{x}_{C_j}) \propto Q(\boldsymbol{x}_{C_j}) e^{(\Delta\boldsymbol{\theta}_j)^T \boldsymbol{f}_j(\boldsymbol{x}_{C_j})}$

Iterative Proportional Fitting

Log-linear Markov random field, separable parameters:

$$P(\boldsymbol{x}) = Z^{-1} e^{\sum_j \boldsymbol{\theta}_j^T \boldsymbol{f}_j(\boldsymbol{x}_{C_j})}, \quad \boldsymbol{\theta} = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \dots)$$

- Assume: For each j, $Q(\boldsymbol{x}_{C_j})$: Can easily find $\Delta\boldsymbol{\theta}_j$ s.t. $\mathrm{E}_{Q_\Delta}[\boldsymbol{f}_j(\boldsymbol{x}_{C_j})] = \boldsymbol{f}_j(\tilde{\boldsymbol{x}}_{C_j})$, where $Q_\Delta(\boldsymbol{x}_{C_j}) \propto Q(\boldsymbol{x}_{C_j}) e^{(\Delta\boldsymbol{\theta}_j)^T \boldsymbol{f}_j(\boldsymbol{x}_{C_j})}$
- Iterative proportional fitting (IPF): Iterate
  - Pick some potential j. Determine marginal $P(\boldsymbol{x}_{C_j})$ (inference)
  - Find $\Delta\boldsymbol{\theta}_j$: $\mathrm{E}_{P_\Delta}[\boldsymbol{f}_j(\boldsymbol{x}_{C_j})] = \boldsymbol{f}_j(\tilde{\boldsymbol{x}}_{C_j})$
  - Update $\boldsymbol{\theta}_j \leftarrow \boldsymbol{\theta}_j + (\Delta\boldsymbol{\theta}_j)$ [Afterwards: $\mathrm{E}_P[\boldsymbol{f}_j(\boldsymbol{x}_{C_j})] = \boldsymbol{f}_j(\tilde{\boldsymbol{x}}_{C_j})$]

Iterative Proportional Fitting

Log-linear Markov random field, separable parameters:

$$P(\boldsymbol{x}) = Z^{-1} e^{\sum_j \boldsymbol{\theta}_j^T \boldsymbol{f}_j(\boldsymbol{x}_{C_j})}, \quad \boldsymbol{\theta} = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \dots)$$

- Assume: For each j, $Q(\boldsymbol{x}_{C_j})$: Can easily find $\Delta\boldsymbol{\theta}_j$ s.t.
  $\mathrm{E}_{Q_\Delta}[\boldsymbol{f}_j(\boldsymbol{x}_{C_j})] = \boldsymbol{f}_j(\tilde{\boldsymbol{x}}_{C_j})$, where $Q_\Delta(\boldsymbol{x}_{C_j}) \propto Q(\boldsymbol{x}_{C_j}) e^{(\Delta\boldsymbol{\theta}_j)^T \boldsymbol{f}_j(\boldsymbol{x}_{C_j})}$
- Iterative proportional fitting (IPF): Iterate
  - Pick some potential j. Determine marginal $P(\boldsymbol{x}_{C_j})$ (inference)
  - Find $\Delta\boldsymbol{\theta}_j$: $\mathrm{E}_{P_\Delta}[\boldsymbol{f}_j(\boldsymbol{x}_{C_j})] = \boldsymbol{f}_j(\tilde{\boldsymbol{x}}_{C_j})$
  - Update $\boldsymbol{\theta}_j \leftarrow \boldsymbol{\theta}_j + (\Delta\boldsymbol{\theta}_j)$ [Afterwards: $\mathrm{E}_P[\boldsymbol{f}_j(\boldsymbol{x}_{C_j})] = \boldsymbol{f}_j(\tilde{\boldsymbol{x}}_{C_j})$]
- Coordinate ascent: Simple, other algorithms can be faster
- Requires inference with changing potentials (e.g., belief propagation)
- Problem with general MRFs: Inference hard [not always: CRFs, next lecture]

F9

Examples for Log-Linear Models

$\Psi_j(\boldsymbol{x}_{C_j}) = \theta^T \boldsymbol{f}_j(\boldsymbol{x}_{C_j})$: Seems special ... No, it's not: Very common!

- Discrete model, multinomial CPTs. F10

Examples for Log-Linear Models

$\Psi_j(\boldsymbol{x}_{C_j}) = \boldsymbol{\theta}^T \boldsymbol{f}_j(\boldsymbol{x}_{C_j})$: Seems special ... No, it's not: Very common!

- Discrete model, multinomial CPTs. $\pi_k = P(x = k)$

$$\theta_k = \log(\pi_k/\pi_K), \quad \boldsymbol{f}(x) = (\mathrm{I}_{\{x=k\}}), \quad k = 1, \dots, K-1,$$
$$\Rightarrow P(x) = Z^{-1} e^{\boldsymbol{\theta}^T \boldsymbol{f}(x)}, \quad Z = 1/\pi_K$$

  Directed Bayesian networks are not log-linear models, but

- Feature based models:
  $\boldsymbol{f}_j$ indicators for presence / strength of certain features

Examples for Log-Linear Models

$\Psi_j(\boldsymbol{x}_{C_j}) = \boldsymbol{\theta}^T \boldsymbol{f}_j(\boldsymbol{x}_{C_j})$: Seems special ... No, it's not: Very common!

- Discrete model, multinomial CPTs. $\pi_k = P(x = k)$

$$\theta_k = \log(\pi_k/\pi_K), \quad \boldsymbol{f}(x) = (\mathrm{I}_{\{x=k\}}), \quad k = 1, \dots, K-1,$$

$$\Rightarrow P(x) = Z^{-1} e^{\boldsymbol{\theta}^T \boldsymbol{f}(x)}, \quad Z = 1/\pi_K$$

  Directed Bayesian networks are not log-linear models, but

- Feature based models:
  $\boldsymbol{f}_j$ indicators for presence / strength of certain features
- Gaussian Markov random field
  Note: Positive definiteness comes for free ($\log Z < \infty$), does not destroy convexity

F10a

Further Points

- Learning MRF with latent variables:
  - Use EM for latent variables (marginal $\rightarrow$ joint likelihood)
  - Use ∇-based optimization / IPF for M step (convex optimization) [No need to maximize, just descent]

Further Points

- Learning MRF with latent variables:
  - Use EM for latent variables (marginal $\to$ joint likelihood)
  - Use ∇-based optimization / IPF for M step (convex optimization) [No need to maximize, just descent]
- Learning with inner maximization ("Viterbi learning")
  - Sometimes: MAP (argmax) easier than inference ($\int$)
  - Learning with maximization can work well (K-Means, ...)
  - In most cases: No equivalent guarantees to learning with inference

[Exceptions: Some work by Taskar, Altun, ...]

Wrap-Up

- Learning requires (marginal) inference in most cases
  $\Rightarrow$ Even frequentists need Bayesian inference
- Inequalities from convexity: Underlying very many ideas / methods
- Expectation Maximization: General-purpose algorithm for marginal likelihood maximization
- Log-linear Markov random fields: Learning is convex optimization