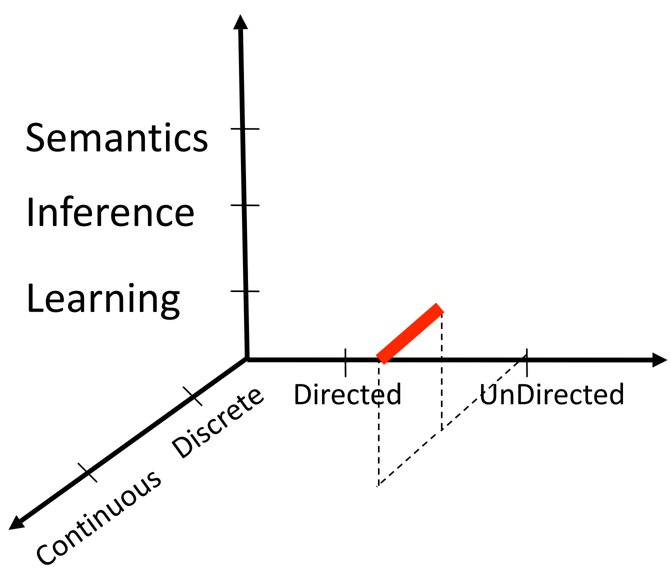
Probabilistic Graphical Models (Cmput 651): Learning Undirected Models

Matthew Brown

{14,17}/11/2008

Reading: Koller-Friedman Ch. 19

Space of topics



Learning Markov nets

	complete data	partial data
known structure	not as easy as for Bayes nets	hard
unknown structure	hard	very hard

↑
This lecture

Learning Markov nets is expensive

Markov factorization: $$P_{\mathcal{F}}(X) = \frac{1}{Z} \prod_{\phi_i \in \mathcal{F}} \phi_i(scope(\phi_i))$$

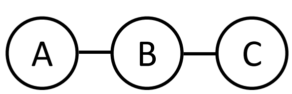
- partition function couples all the factors
  cannot separate parameter estimation into local groups
- no closed form solutions
  - even for max. likelihood w/ complete data
  (recall this is easy for Bayes nets)
- learning based on iteration
  - inference required in each step -> expensive
  - but convex (for complete data)
- structure learning also expensive

Outline

<u>Parameter learning</u>

Structure learning

Likelihood and the partition function (also see KF 19.2.1)



$$\phi_1[A, B]$$
$$\phi_2[B, C]$$

Log likelihood: $\ln P(a, b, c) = \ln \left(\frac{1}{Z} \phi_1[a, b] \phi_2[b, c] \right)$

$$= \ln \phi_1[a, b] + \ln \phi_2[b, c] - \ln Z$$

Log likelihood for data *D* with M instances:

$$\begin{aligned} \ell(\theta : \mathcal{D}) &= \sum_m (\ln \phi_1[a[m], b[m]] + \ln \phi_2[b[m], c[m]] - \ln Z) \\ &= \sum_{a,b} M[a, b] \ln \phi_1[a, b] + \sum_{b,c} M[b, c] \ln \phi_2[b, c] - M \ln Z(\theta) \end{aligned}$$

Counts of various assignments in *D*

Likelihood and the partition function (also see KF 19.2.1)

(Continued from last slide)

Log likelihood for data *D* with M instances:

$$\begin{aligned}\ell(\theta : \mathcal{D}) &= \sum_m \left(\ln \phi_1[a[m], b[m]] + \ln \phi_2[b[m], c[m]] - \ln Z\right) \\ &= \sum_{a,b} M[a,b] \ln \phi_1[a,b] + \sum_{b,c} M[b,c] \ln \phi_2[b,c] - M \ln Z(\theta)\end{aligned}$$

Counting terms involve only a single factor

Partition function term involves ALL factors:

$$Z(\theta) = \sum_{a,b,c} \phi_1[a,b]\phi_2[b,c]$$

Maximum likelihood estimation (also see KF 19.2.1)

Bayes nets: estimate conditional distributions separately for each node

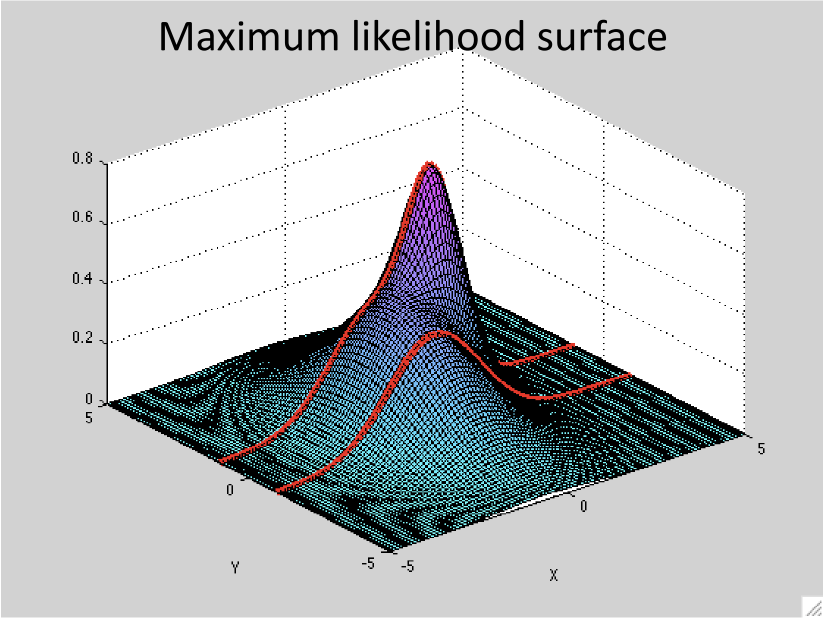
-> likelihood decomposable

vs.

Markov nets: partition function involves all factors

changing ϕ_1 could change optimal value for ϕ_2

Maximum likelihood estimation (also see KF 19.2.1)



Max w.r.t. X depends on value of Y

Log-likelihood (also see KF 19.2.2)

Log linear model: $P(X_1, \ldots, X_n : \theta) = \frac{1}{Z} \exp \sum_{i=1}^{k} \theta_i \phi_i[\boldsymbol{D}_i]$

weight indicator function

Log-likelihood: $\ell(\theta : \mathcal{D}) = \sum_i \theta_i \left(\sum_m \phi_i[\xi[m]] \right) - M \ln Z(\theta)$

ξ[m]=mth data point

Log-likelihood (also see KF 19.2.2)

Log-likelihood: $$\ell(\boldsymbol{\theta} : \mathcal{D}) = \sum_i \theta_i \left(\sum_m \phi_i[\xi[m]] \right) - M \ln Z(\boldsymbol{\theta})$$

Divide by M (no. data points)

$$\frac{1}{M}\ell(\boldsymbol{\theta} : \mathcal{D}) = \sum_i \theta_i \boldsymbol{E}_{\mathcal{D}}[\phi_i[\boldsymbol{d}_i]] - \ln Z(\boldsymbol{\theta})$$

Empirical expectation of ϕ_i

Partition function: $$\ln Z(\boldsymbol{\theta}) = \ln \sum_\xi \exp\left\{ \sum_i \theta_i \phi_i[\xi] \right\}$$

Sum over data points

Log-likelihood (also see KF 19.2.3)

$$\frac{1}{M}\ell(\boldsymbol{\theta} : \mathcal{D}) = \sum_i \theta_i \boldsymbol{E}_{\mathcal{D}}[\phi_i[\boldsymbol{d}_i]] - \ln Z(\boldsymbol{\theta})$$

Linear in $\boldsymbol{E}_{\mathcal{D}}[\phi_i[\boldsymbol{d}_i]]$

Partition function: $$\ln Z(\boldsymbol{\theta}) = \ln \sum_\xi \exp\left\{ \sum_i \theta_i \phi_i[\xi] \right\}$$

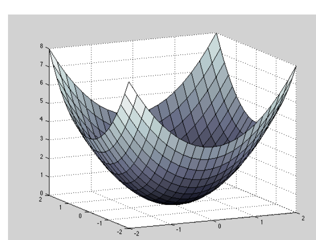
- is <u>convex</u>

$\Rightarrow$ log-likelihood is <u>concave</u>

Convex function:

$$f(\alpha\vec{x} + (1-\alpha)\vec{y}) \le \alpha f(\vec{x}) + (1-\alpha) f(\vec{y})$$

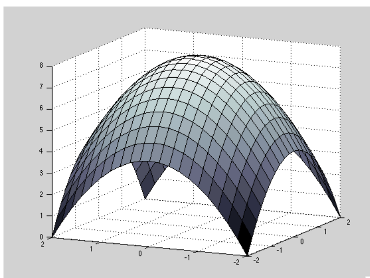
$$0 \ge \alpha \ge 1$$



Log-likelihood (also see KF 19.2.3)

Log-likelihood is concave

- ∃ global maximum
- ∄ local maxima
- global maximum may not be <u>unique</u>
    - Markov net parameterization may be redundant
        - i.e. multiple representations of same distribution



include simple examples based on diamond net if there's time

Maximum likelihood parameter estimation (KF 19.3.1)

Want to find parameters that maximize log-likelihood

Gradient = 0 at maximum

$$\frac{\partial}{\partial \theta_i}\frac{1}{M}\ell(\boldsymbol{\theta}:\mathcal{D}) = \sum_i \theta_i \boldsymbol{E}_{\mathcal{D}}[\phi_i[\boldsymbol{d}_i]] - \ln Z(\boldsymbol{\theta})$$

$$\frac{\partial}{\partial \theta_i}\frac{1}{M}\ell(\boldsymbol{\theta}:\mathcal{D}) = \boldsymbol{E}_{\mathcal{D}}[\phi_i[\mathcal{X}]] - \boldsymbol{E}_{\boldsymbol{\theta}}[\phi_i]$$

KF Proposition 19.2.3

$$\frac{\partial}{\partial \theta_i}\ln Z(\boldsymbol{\theta}) = \boldsymbol{E}_{\boldsymbol{\theta}}[\phi_i]$$

-> @ maximum: $\boldsymbol{E}_{\mathcal{D}}[\phi_i[\mathcal{X}]] = \boldsymbol{E}_{\hat{\boldsymbol{\theta}}}[\phi_i]$

BUT ∄ closed form solution!

- use gradient ascent instead

Maximum likelihood parameter estimation

$$\frac{\partial}{\partial \theta_i} \frac{1}{M} \ell(\boldsymbol{\theta} : \mathcal{D}) = \boldsymbol{E}_{\mathcal{D}}[\phi_i[\mathcal{X}]] - \boldsymbol{E}_{\boldsymbol{\theta}}[\phi_i]$$

Show numerical example using A-B-C network

Maximum likelihood parameter estimation (KF 19.3.1)

Want to find parameters that maximize log-likelihood

<u>Gradient ascent</u> $$\frac{\partial}{\partial \theta_i} \frac{1}{M} \ell(\boldsymbol{\theta} : \mathcal{D}) = \boldsymbol{E}_{\mathcal{D}}[\phi_i[\mathcal{X}]] - \boldsymbol{E}_{\boldsymbol{\theta}}[\phi_i]$$

Easy to compute

Expected counts over parameter space
- requires inference
- once at <u>each</u> step of gradient ascent
- typically expensive

Conditionally-trained models (also see KF 19.3.2)

Discriminative (instead of generative)

Conditional random field (CRF)

encodes $P(\boldsymbol{Y} \mid \boldsymbol{X})$

Maximize log conditional likelihood

$$\ell_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{\theta} : \mathcal{D}) = \ln P(\boldsymbol{y}[1,\ldots,M] \mid \boldsymbol{x}[1,\ldots,M], \boldsymbol{\theta}) = \sum_{m=1}^{M} \ln P(\boldsymbol{y}[m] \mid \boldsymbol{x}[m], \boldsymbol{\theta})$$

- is concave
- use gradient ascent

Conditionally-trained models (also see KF 19.3.2)

Gradient ascent on log conditional likelihood

$$\ell_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{\theta} : \mathcal{D}) = \ln P(\boldsymbol{y}[1,\ldots,M] \mid \boldsymbol{x}[1,\ldots,M], \boldsymbol{\theta}) = \sum_{m=1}^{M} \ln P(\boldsymbol{y}[m] \mid \boldsymbol{x}[m], \boldsymbol{\theta})$$

Gradient:

$$\frac{\partial}{\partial \theta_i} \ell_{\boldsymbol{Y}|\boldsymbol{X}}(\boldsymbol{\theta} : \mathcal{D}) = \sum_{m=1}^{M} \Big[\phi_i[\boldsymbol{y}[m], \boldsymbol{x}[m]] - \boldsymbol{E}_{\boldsymbol{\theta}}[\phi_i \mid \boldsymbol{x}[m]] \Big]$$

Counts on dataset

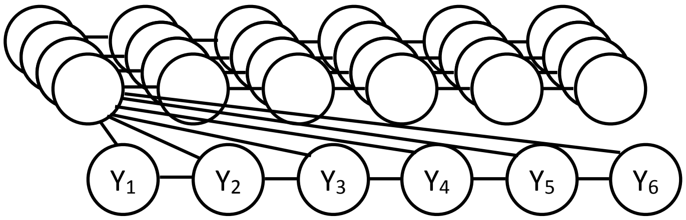
Expectation w.r.t. model conditioned on m^th data point

-> must run inference M times for each gradient step!

V. expensive

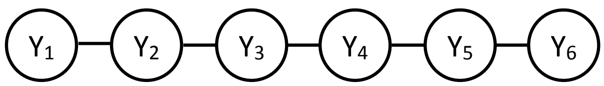
But, can be offset by simpler model (comp. to generative)

Conditionally-trained models (also see KF Example 19.3.3)



Evidence nodes w/ dense interconnectivity (only partly shown)

Discriminative model conditioned on evidence nodes



Inference on chain is much easier

Maximum entropy & maximum likelihood (KF 19.3.4)

Given some data D, find distribution Q that matches it without a lot of extra structure / assumptions

Maximum-Entropy

Find $Q(\mathcal{X})$

that maximize $\boldsymbol{H}_Q(\mathcal{X})$ ← Entropy
high value -> little structure

subject to

$$\boldsymbol{E}_Q[\phi_i] = \boldsymbol{E}_{\mathcal{D}}[\phi_i] \quad i = 1, \ldots, k$$

expectation constraints

Maximum entropy & maximum likelihood (KF 19.3.4)

Theorem: Max entropy solution Q* (to problem from previous slide) satisfies

$$Q^* = P_{\hat{\theta}}(\mathcal{X}) = \frac{1}{Z(\hat{\theta})} \exp\left\{\sum_i \hat{\theta}_i \phi_i[\mathcal{X}]\right\}$$

$\hat{\theta}$ = maximum likelihood solution relative to data set D

i.e. max likelihood and max entropy are related

Priors and regularization (also see KF 19.4)

Max. likelihood estimation prone to overfitting

Use priors to constrain θ parameters

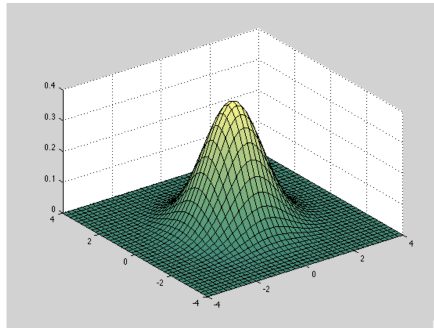
from log linear model $P(X_1, \ldots, X_n : \theta) = \frac{1}{Z} \exp \sum_{i=1}^{k} \theta_i \phi_i[D_i]$

closed form like Bayes formula not available

use max. a posteriori (MAP) instead - maximize $P(\theta)P(\mathcal{D} \mid \theta)$

Gaussian prior (also see KF 19.4.1)

$$P(\theta \mid \sigma^2) = \prod_{i=1}^{k} \frac{1}{\sqrt{2\pi}} \exp\left\{-\frac{\theta_i^2}{2\sigma^2}\right\}$$



σ^2 hyperparameter for variance

- Amount of regularization
- Can use different σ_i

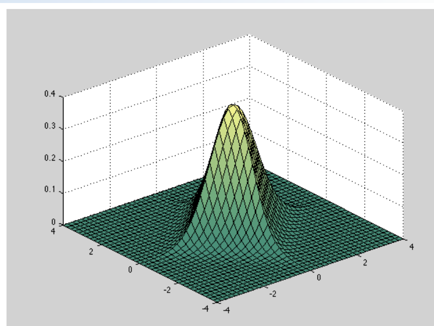
Typically use $Cov(\theta_i, \theta_j)=0$ for $i \neq j$

- i.e. assume θ_i independent

Predisposes θ's to be small

Laplacian prior (also see KF 19.4.1)

$$P_{Laplacian}(\theta \mid \beta) = \frac{1}{2\beta} \exp\left\{-\frac{|\theta|}{\beta}\right\}$$



β hyperparameter

- Can use different β_i

Assume θ_i independent

Also predisposes θ's to be small

Priors and regularization (also see KF 19.4.1)

$$P(\theta, \mathcal{D}) = P(\theta) P(\mathcal{D} \mid \theta)$$

Prior Likelihood

Penalty term regularizes MLE

Taking logarithms:

$$\ell(\theta : \mathcal{D}) = \sum_i \theta_i \left(\sum_m \phi_i[\xi[m]] \right) - M \ln Z(\theta) + \text{penalty term}$$

$$-\frac{1}{2\sigma^2} \sum_{i=1}^{k} \theta_i^2$$ Gaussian penalty

L_2 regularization

$$-\frac{1}{\beta} \sum_{i=1}^{k} |\theta_i|$$ Laplacian penalty

L_1 regularization

L_1 vs L_2 regularization (also see KF 19.4.1)

$$-\frac{1}{2\sigma^2} \sum_{i=1}^{k} \theta_i^2$$

L_2 regularization

larger θ values penalized more heavily

$$-\frac{1}{\beta} \sum_{i=1}^{k} |\theta_i|$$

L_1 regularization

uniform penalty

-> better at driving θ <u>all the way</u> to zero

-> sparser models learned (more θ_i=0)

Learning with approximation (also see KF 19.5)

How to learn when inference is hard?

- eg: grid networks

Approach 1: approximate inference inside learning loop

- generalized belief propagation
- particle-based methods

Approach 2: approximate cost function

- inference easier

In many cases, approaches 1 and 2 are formally equivalent.

Learning with belief propagation (also see KF 19.5.1)

Theorem: When using an approximate inference algorithm on the trained model, it is best to do the training with the same inference algorithm.

BUT: generalized belief propagation inside a gradient-based learning loop can cause problems:

- marginals are only approximate -> noise in gradient
- possible non-convergence -> unstable gradient
- can use heuristics & manual testing to address these
- one solution on next slide

Learning with belief propagation (also see KF 19.5.1.2)

Recall: Generalized belief propagation equivalent to optimization on approximate factored energy functional:

$$\tilde{F}[P_{\mathcal{F}}, Q] = \sum_i \boldsymbol{E}_{\boldsymbol{C}_i \sim \beta_i}[\ln \psi_i] + \sum_{\boldsymbol{C}_i \in \mathcal{T}} \boldsymbol{H}_{\beta_i}(\boldsymbol{C}_i) - \sum_{(\boldsymbol{C}_i - \boldsymbol{C}_j) \in \mathcal{T}} \boldsymbol{H}_{\mu_{i,j}}(\boldsymbol{S}_{i,j})$$

which comes from KL-Divergence.

Can use similar approach for learning

(go to next slide)

Learning with belief propagation (also see KF 19.5.1.2)

Want tractable approximation to:

Maximum-Entropy

Find $Q(\mathcal{X})$
that maximize $\boldsymbol{H}_Q(\mathcal{X})$
subject to

$$\boldsymbol{E}_Q[\phi_i] = \boldsymbol{E}_{\mathcal{D}}[\phi_i] \quad i = 1, \ldots, k$$

(equivalent to max. likelihood)

Solution = <u>constrained</u> optimization of factored form of entropy:

$$\boldsymbol{H}_Q(\mathcal{X}) \approx \sum_{\boldsymbol{C}_i \in \mathcal{K}} \boldsymbol{H}_{\beta_i}(\boldsymbol{C}_i) - \sum_{(\boldsymbol{C}_i - \boldsymbol{C}_j) \in \mathcal{K}} \boldsymbol{H}_{\mu_{i,j}}(\boldsymbol{S}_{i,j})$$

Learning with belief propagation (also see KF 19.5.1.2)

Constrained optimization of factored form of entropy:

$$\boldsymbol{H}_Q(\mathcal{X}) \approx \sum_{\boldsymbol{C}_i \in \mathcal{K}} \boldsymbol{H}_{\beta_i}(\boldsymbol{C}_i) - \sum_{(\boldsymbol{C}_i - \boldsymbol{C}_j) \in \mathcal{K}} \boldsymbol{H}_{\mu_{i,j}}(\boldsymbol{S}_{i,j})$$

valid objective function

- avoid (non)convergence issues
- use whatever optimization method you want

reformulation exact when cluster graph = tree

- approximate otherwise
  - e.g. generalized belief propagation

Alternate objective functions (also see KF 19.6)

Goals:

easier objective than likelihood

VALID objective

no convergence issues

Alternate objective functions (also see KF 19.6)

Likelihood of one data point:

$$\ell(\boldsymbol{\theta} : \xi) = \ln \tilde{P}(\xi \mid \boldsymbol{\theta}) - \ln Z(\boldsymbol{\theta}) = \ln \tilde{P}(\xi \mid \boldsymbol{\theta}) - \ln \left(\sum_{\xi'} \tilde{P}(\xi' \mid \boldsymbol{\theta}) \right)$$

Want to make this large

Want to make this small

In 2nd term, summation over all assignments to Val(X) requires inference -> expensive

Approach:

in 2nd term, use more tractable set than all of Val(X)

Pseudo-likelihood (also see KF 19.6.1)

Likelihood $P(\xi) = \prod_{j=1}^{n} P(x_j \mid x_1, \ldots, x_{j-1})$ (from chain rule)

$$P(\xi) \approx \prod_{j} P(x_j \mid x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n)$$

Pseudo-likelihood

$$\ell_{\text{pseudo}}(\boldsymbol{\theta} : \mathcal{D}) = \frac{1}{M} \sum_{m} \sum_{j} \ln P(x_j[m] \mid \boldsymbol{x}_{-j}[m], \boldsymbol{\theta})$$

$\boldsymbol{x}_{-j}$ means $x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n$

Note: $P(X_j | X_-j) = P(X_j | Neighbours_{X_j})$

Pseudo-likelihood (also see KF 19.6.1)

$$\ell_{\text{pseudo}}(\boldsymbol{\theta} : \mathcal{D}) = \frac{1}{M} \sum_m \sum_j \ln P(x_j[m] \mid \boldsymbol{x}_{-j}[m], \boldsymbol{\theta})$$

$\boldsymbol{x}_{-j}$ means $x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n$

Why do we care?

~ means unnormalized

$$P(x_j \mid \boldsymbol{x}_{-j}) = \frac{P(x_j, \boldsymbol{x}_{-j})}{P(\boldsymbol{x}_{-j})} = \frac{\tilde{P}(x_j, \boldsymbol{x}_{-j})}{\tilde{P}(\boldsymbol{x}_{-j})} = \frac{\tilde{P}(x_j, \boldsymbol{x}_{-j})}{\sum_{x'_j} \tilde{P}(x'_j, \boldsymbol{x}_{-j})}$$

-> no global partition function!

only local partition function

much cheaper to evaluate

Gradient of pseudo-likelihood (KF Definition 19.6.1)

$$\frac{\partial}{\partial \theta_i} \ell_{\text{pseudo}}(\boldsymbol{\theta} : \mathcal{D}) = \sum_{j : X_j \in Scope[\phi_i]} \left(\frac{1}{M} \sum_m \phi_i[\xi[m]] - \boldsymbol{E}_{x'_j \sim P_{\boldsymbol{\theta}}(X_j \mid \boldsymbol{x}_{-j}[m])} \left[\phi_i[x'_j, \boldsymbol{x}_{-j}[m]] \right] \right)$$

Much cheaper than likelihood's gradient:

$$\boldsymbol{E}_{x'_j \sim P_{\boldsymbol{\theta}}(X_j \mid \boldsymbol{x}_{-j}[m])} \left[\phi_i[x'_j, \boldsymbol{x}_{-j}[m]] \right]$$

-> summation only over X_j conditioned on its neighbours

(not over all of X as in partition function Z)

i.e. inference over only small part of graph (X_j and neighbours)

Pseudo-likelihood (also see KF 19.6.1)

Pseudo-likelihood is <u>concave</u>

-> unique, global maximum

Likelihood & pseudo-likelihood (KF Thoerem 19.6.2)

<u>Theorem</u>:

Assuming data generated by log-linear model P_{θ^*},

as no. data points $M \rightarrow \infty$, $P(\theta_{pl} = \theta^*) \rightarrow 1$,

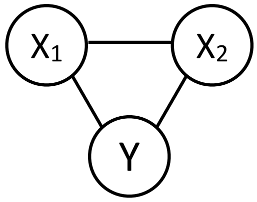
where θ_{pl} = global optimum of pseudo-likelihood objective

i.e. pseudo-likelihood converges to likelihood with large M

BUT, this assumes sufficiently expressive model and large M

- these assumptions typically do not hold

Likelihood & pseudo-likelihood (KF Example 19.6.3)



$X_1 \simeq X_2$

X_1, X_2 somewhat correlated with Y
Pseudo-likelihood will overestimate X_1-X_2 parameters and underestimate X_i-Y parameters.
Okay for $P(X_2|X_1)$ but not $P(X_2|Y)$

In general, pseudo-likelihood assumes X's neighbourhood is observed
"ignores" weaker or longer-ranger range dependencies

Contrastive optimization (also see KF 19.6.2)

log-likelihood:

$$\ell(\boldsymbol{\theta} : \xi) = \ln \tilde{P}(\xi \mid \boldsymbol{\theta}) - \ln Z(\boldsymbol{\theta}) = \ln \tilde{P}(\xi \mid \boldsymbol{\theta}) - \ln \left(\sum_{\xi'} \tilde{P}(\xi' \mid \boldsymbol{\theta}) \right)$$

Want to maximize contrast between these

Similarly motivated methods:
contrastive divergence
max. margin training

Outline

Parameter learning

Structure learning

Structure learning (also see KF 19.7)

constraint-based

- constrain structure to reflect independencies in P(X)

score-based

- score each structure, optimize score

Constraint-based structure learning (KF 19.7.1)

Similar to case for Bayes net.

Want H^* to factorize P^*

- assume: H^* perfect map for P^*, degree $H^* \leq d^*$

Test for independencies:

- Markov independence $(X \perp \mathcal{X} - \{X\} - \mathcal{N}_{\mathcal{H}^*}(X) \mid \mathcal{N}_{\mathcal{H}^*}(X)) \quad \forall X$
- Pairwise independence $(X \perp Y \mid \mathcal{X} - \{X, Y\}) \;\; \forall (X — Y) \notin \mathcal{H}$
- BUT, testing Markov or pairwise involves all variables
  - -> exponential in num. nodes
  - see next slide

Testing for independencies

Independence testing using pairwise independence:

$$(X \perp Y \mid \mathcal{X} - \{X, Y\}) \;\; \forall (X — Y) \notin \mathcal{H}$$

Suppose N binary nodes:

$\exists \binom{N}{2}$ sets {X,Y} to test

for each of 4 assignments {x,y}, must check equality under $2^{(N-2)}$ assignments to other nodes in $\mathcal{X} - \{X, Y\}$

-> exponential!

(similar argument for Markov independence)

Independence testing (KF 19.7.1)

(Assuming degree (max # edges / node) $H^* \leq d^* << N$)

Consider X,Y

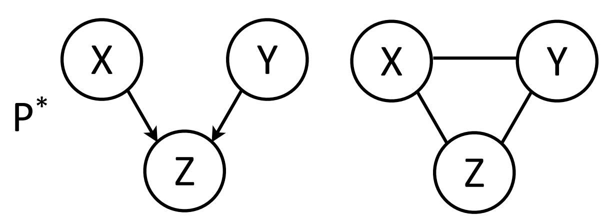
- no edge -> $\mathcal{N}_{\mathcal{H}^*}(X)$ and $\mathcal{N}_{\mathcal{H}^*}(Y)$ separate X and Y
  - i.e. $\exists$ set Z with $|Z| \leq \min(|\mathcal{N}_{\mathcal{H}^*}(X)|, |\mathcal{N}_{\mathcal{H}^*}(Y)|)$
  - such that $sep_{\mathcal{H}^*}(X; Y \mid Z)$
- so: $X — Y \notin \mathcal{H}^*$ if and only if $\exists Z, |z| \leq d^* \& P^* \models (X \perp Y \mid Z)$
- For each pair X,Y test for edge using $\sum_{k=0}^{d^*} \binom{n-2}{k}$ tests
  - Polynomial number of tests
- Each test involves $\leq d^*+2$ variables
  - $\leq 2^{d^*+2}$ assignments to check for binary nodes
- Tractable for small d^*

Constraint-based structure learning (KF 19.7.1)

Limitations:

- assume H^* perfect map for P^*
- assume bounded order for H^*
- assume enough data for reliable independence tests

Example



Correct H^* unreachable because $X \perp Y \mid \{\}$

-> discard X-Y edge

i.e. no Markov net is perfect map for P^*

Constraint-based structure learning (KF 19.7.1)

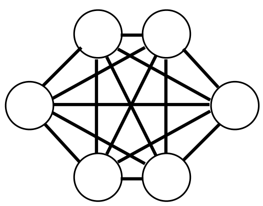
Limitations (cont'd):

For Markov nets, global independence structure not necessarily useful

eg: fully connected network with pairwise potentials

-> complex connectivity but compact factorization

constraint-based learning does not help find factorization



Score-based learning (KF 19.7.2)

<u>Hypothesis space</u>

- log-linear model

$$P(\mathcal{X} \mid \mathcal{M}, \theta) = \frac{1}{Z} \exp\left\{ \sum_{i \in \Phi[\mathcal{M}]} \theta_i \phi_i[\xi] \right\} = \frac{1}{Z} \exp\left\{ \phi^T \theta \right\}$$

Also:

- basic graph structure
- factor graph

-> different levels of model "granularity"

stru

Score-based learning (KF 19.7.2)

Hypothesis space

- log-linear model

$$P(\mathcal{X} \mid \mathcal{M}, \boldsymbol{\theta}) = \frac{1}{Z} \exp\left\{ \sum_{i \in \Phi[\mathcal{M}]} \theta_i \phi_i[\xi] \right\} = \frac{1}{Z} \exp\left\{ \phi^T \boldsymbol{\theta} \right\}$$

- Given set of features Ω,
  derive model $\mathcal{M}$ from features $\Phi[\mathcal{M}] \subseteq \Omega$
  by setting $\theta_i = 0$ if $\phi_i \notin \Phi[\mathcal{M}]$
  (Also, optimize other θ_i)
- Structure implicit in $\mathcal{M}$
  connect all $X \in scope(\phi_i), \forall \phi_i \in \Phi[\mathcal{M}]$

stru

Score function (also see KF 19.7.3.1)

log-likelihood

$$\text{score}_L(\mathcal{M} \ : \ \mathcal{D}) = \max_{\boldsymbol{\theta} \in \Theta[\mathcal{M}]} \ln P(\mathcal{D} \mid \mathcal{M}, \boldsymbol{\theta}) = \ell(\langle \mathcal{M}, \hat{\boldsymbol{\theta}}_{\mathcal{M}} \rangle : \mathcal{D})$$

model data

Overfitting problem

$\Phi[\mathcal{M}_1] \subset \Phi[\mathcal{M}_2]$ -> $\text{score}_L(\mathcal{M}_1 \ : \ \mathcal{D}) \leq \text{score}_L(\mathcal{M}_2 \ : \ \mathcal{D})$

more expressive model fits noise in D

must restrict factors' expressiveness or regularize

Greedy MN structure learning (also see KF Fig. 19.3)

Total feature set Ω

Initial feature set Φ_0

at all times: $\theta_i = 0, \forall \phi_i \notin \Phi$

Iterate {

Optimize θ_Φ (parameter optimization)

Iterate over modification operators $\mathcal{O}$ to structure {

$\mathcal{O}$ creates Φ_{mod} (see next slide)

$\hat{\Delta}_\mathcal{O}$ = improvement in score

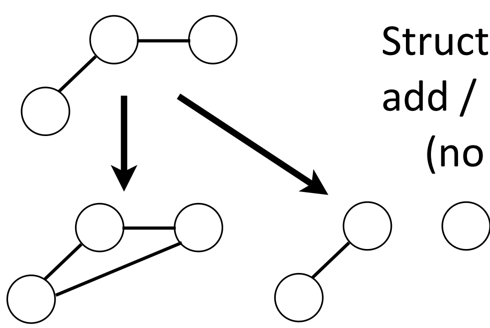
}

choose set of modifications $\mathcal{O}$ based on $\hat{\Delta}_\mathcal{O}$

-> new structure Φ

}

Structure modification



Structure modification steps $\mathcal{O}$:

add / remove single edge

(no directionality!)

Then need score for each modification $\hat{\Delta}_\mathcal{O}$

(see below)

Structure scoring (also see KF 19.7.4.2)

Bayes nets:

- structure score evaluation easy
  - closed form available
- score decomposes based on structure
  - change to structure changes only one term in the score
  - changes to different parts of structure do not interact in the score
- -> efficiency: dynamic programming, caching, etc.

Structure learning performance (also see KF 19.7.4.2)

Markov nets:

- structure score evaluation **harder**

Score for each modification $\hat{\Delta}_{\mathcal{O}}$

- must optimize $\theta_{\Phi_{mod}}$
  - requires inferences inside gradient ascent loop
  - can start from $\theta_{\Phi_{current}}$ to speed things up
  - still expensive (& inside overall structure iteration loop)

Cheaper: rank order $\hat{\Delta}_{\mathcal{O}}$ (instead of full evaluation)

Structure learning performance (also see KF 19.7.4.2)

Markov nets: structure score evaluation **harder**

- partition function couples everything
- computing likelihood score requires inference
- structure score requires parameter estimation (no closed form)
- structure score does not decompose
- -> expensive!

Good news:

- structure learning is convex (with fully observed data)

structure learning more expensive for Markov vs. Bayes

Regularization & structure learning (also see KF 19.7.3.2)

Bayesian score $\text{score}_B(\mathcal{G} : \mathcal{D}) = \log P(\mathcal{D} \mid \mathcal{G}) + \log P(\mathcal{G})$

$$P(\mathcal{D} \mid \mathcal{G}) = \int \underbrace{P(\mathcal{D} \mid \mathcal{M}, \theta)}_{\text{likelihood}} \underbrace{P(\theta \mid \mathcal{M})}_{\text{prior}} d\theta$$

- marginal likelihood
  - average based on parameter prior
- regularizes parameters
  - avoids overfitting
- efficient for Bayes nets
- but too hard to evaluate for Markov nets
  - (because of partition function)
  - use BIC score instead (next slide)

Regularization & structure learning (also see KF 19.7.3.2)

$$\text{score}_{BIC}(\mathcal{M} : \mathcal{D}) = \ell(\langle \mathcal{M}, \hat{\theta}_{\mathcal{M}} \rangle : \mathcal{D}) - \frac{\dim(\mathcal{M})}{2} \ln M$$

Asymptotic approximation to marginal likelihood

dim(M) = dimension of model

degrees of freedom

-> penalizes more complex models (i.e. more D.O.F.)

Other regularizations for structure learning (KF 19.7.4)

MAP score

L_1 regularization

$$\text{score}_{L_1}(\theta : \mathcal{D}) == \ell(\langle \mathcal{M}, \theta \rangle : \mathcal{D}) - \|\theta\|_1$$