Lecture 25: April 22, 2015
cs 573: Probabilistic Reasoning

Professor Nevatia

Spring 2015

Review

- HW#7 due today
- Please fill in class evaluation (on-line)
- Exam2: April 29, class period, here (except for remote DEN students)
  - Closed book/notes
  - Detailed list of topics in following slide
  - Style similar to Exam1 but likely to be more descriptive
- Previous Lecture
  - EM for clustering
    - K-means
    - Mixture of Gaussians
    - LDA
- Today's objective
  - Learning MN parameters
  - Intro to structure learning

Exam Topics

- Lec 13-25 (both inclusive); Exam1 topics will not be repeated in Exam 2
- Topics:
  - Gaussian Networks: Ch. 7, 14.2
  - Variational Approximation: 11.5 (only as covered in class)
  - MAP inference: 13.1, 13.2 and 13.3
  - Sampling: 12.1, 12.2 and 12.3 (including Metropolis-Hastings algorithm)
  - Temporal Models: 6.2, 15.1, 15.2, 15.3 (as covered in class)
  - Parameter Estimation: 17.1, 17.2 (exclude 17.2.4, 17.2.5), 17.3
  - Partially Observed Data: 19.1, 19.2 (except 19.2.2.5 and 19.2.2.6, 19.2.4)
  - Latent Dirichlet Allocation, class slides; Blei paper (optional)
  - Markov Networks: 20.1, 20.2, 20.3 (except 20.3.4)
  - Structure Learning18.1 to 18.4 (as covered in class)

Learning Undirected Models

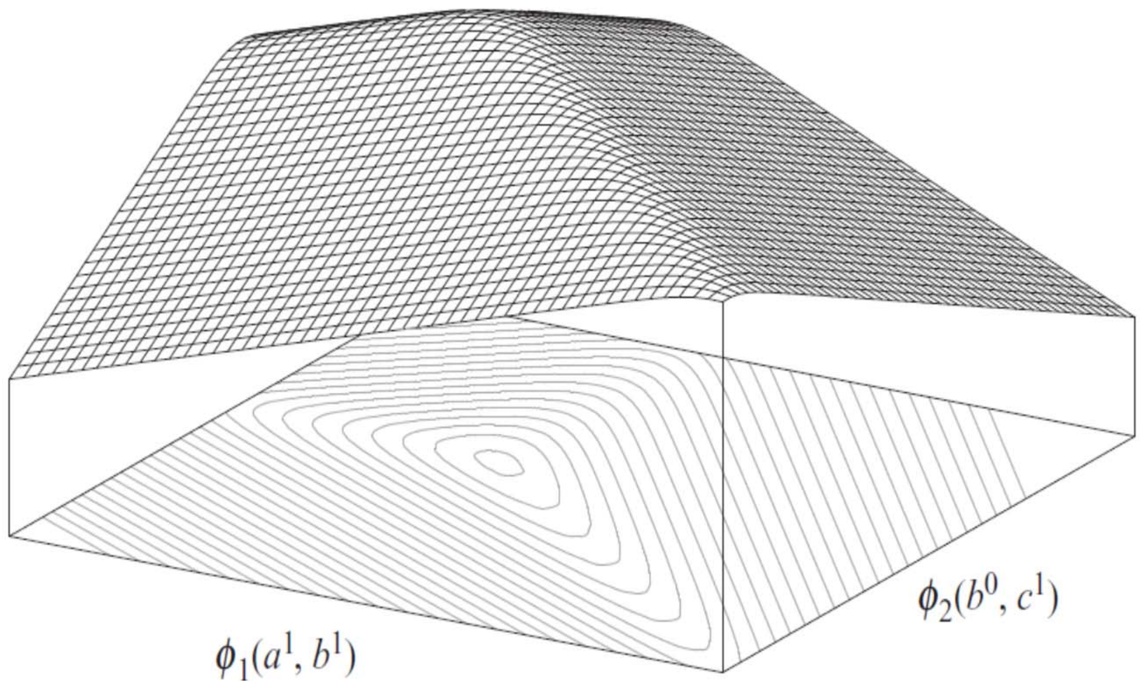
- Partition function is global, rather than local, as in CPDs for a BN, so MLE maximization is not local
- Consider simple A – B– C network
- $P(a, b, c) = (1/Z)\ \phi_1(a,b)\ .\ \phi_2(b,c)$
- $\ln P(a, b, c) = \ln \phi_1(a,b) + \ln \phi_2(b,c) - \ln Z$
- $l(\boldsymbol{\theta} : D) = P(D: \boldsymbol{\theta})$

  $= \Sigma_m (\ln \phi_1(a[m],b[m]) + \ln \phi_2 (b[m],c[m]) - \ln Z(\boldsymbol{\theta}))$

  $= \Sigma_{a,b} M[a,b] \ln \phi_1(a,b) + \Sigma_{b,c} M[b,c] \ln \phi_2(b,c) - M \ln Z(\boldsymbol{\theta})$
- We want to choose $\boldsymbol{\theta}$ to maximize *l* above
  - Seems that we should be able to choose $\phi_1$and ϕ_2 values independently, as in the case of a BN
    - However, Z term couples both as $Z(\boldsymbol{\theta}) = \Sigma_{a,b,c}\ \phi_1(a,b)\ \phi_2(b,c)$
- Local optimization is not possible.
  - If MN is chordal, can convert to BN, learn parameters and translate back.

Likelihood Function

Simple network A – B – C; binary valued variables



MNs with Log-linear Models

- Log-linear model review:

  $P(X_1, \ldots, X_n) = (1/z)\, \Pi_i\, \phi_i\, (D_i)$; in terms of potential functions

  $=(1/z) \exp \{-\Sigma_i\, \varepsilon_i\, (D_i)\}$; in terms of log potential functions

  $= (1/z) \exp \{-\sum_k w_k f_k\, (D_k)\}$; in terms of feature functions

  – Changing notation slightly:

    $P(X_1, \ldots, X_n : \boldsymbol{\theta}) = (1/z(\boldsymbol{\theta})) \exp \{\sum_i \theta_i\, f_i\, (D_i)\}$

  – Feature functions are typically defined by indicator functions e.g. $f_{a0,b0}\, (a,b) = 1\{A= a^0\}\ .\ 1\{B= b^0\}$; θ_i define the weights

- Learning task is to estimate the θ_i weights

  – Maximize log-likelihood for MLE

Log-likelihood

- $l\,(\boldsymbol{\theta} : D) = \sum_i \theta_i \,(\sum_m f_i(\xi[m]) - M \ln z(\boldsymbol{\theta})$

  $\ln z(\boldsymbol{\theta}) = \ln \sum_\xi \exp\,(\sum_i \theta_i \, f_i(\xi))$
- Sufficient statistics for likelihood function are sums of feature values in instances of D.
- Divide by M (number of samples) to get:

  $(1/M)\, l\,(\boldsymbol{\theta} : D) = \sum_i \theta_i \, E_D[f_i(\mathbf{d}_i)] - \ln z(\boldsymbol{\theta})$
- *Can be shown* that $\ln z(\boldsymbol{\theta})$ is a *convex* function (bowl-like)
  – $-\ln z(\boldsymbol{\theta})$ is concave
- Follows that $l\,(\boldsymbol{\theta} : D)$ is concave as it a sum of a linear function and a concave function
  – Unimodal, so has a single global optimum but may be achieved for several values of $\boldsymbol{\theta}$.

MLE Estimate

- How to find this maximum?
  - Use gradient ascent methods
  - Analytical form for gradient can be found (eq. 20.4)

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

  - The terms above are counts of samples where function f_i has values 1 (for binary functions)
    - 1st term is the number of observed counts, 2nd term is the # of expected counts which requires inference over the entire network (given **θ**)
  - Inference for all features can be made in one calibration round
  - Can be expensive if many iterations are needed and the network is large

CRFs

- In CRFs, we encode P(**Y**|**X**) rather than P(Y,X)
- We maximize conditional likelihood (or log of it):

$$l_{Y|X}(\boldsymbol{\theta} : D) = \Sigma_m \ln P\,(y[m]|\,(x[m]) : \boldsymbol{\theta}))$$

- Unimodal as before; we can also compute gradient formula

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

  - Similar to earlier case but now the expected counts are *given the evidence variables*, so inference must be run for **each sample** at each iteration step.
  - However, given evidence, we work with a simpler (reduced) network which compensates for some of the complexity
- If domain of X is large, training of conditional model is much more feasible then the complete generative model.
- CRFs remain highly popular; may require more training data

Markov Networks with Missing Data

- Similar complexities as for the case of BNs
- Likelihood function is no longer concave
- Let **o**[m] be observed values, **h**[m] assignment for missing variables
- Log-likelihood function:

$$\begin{aligned}\frac{1}{M}\ln P(\mathcal{D}\mid\boldsymbol{\theta}) &= \frac{1}{M}\sum_{m=1}^{M}\ln\left(\sum_{\boldsymbol{h}[m]}P(\boldsymbol{o}[m],\boldsymbol{h}[m]\mid\boldsymbol{\theta})\right)\\ &= \frac{1}{M}\sum_{m=1}^{M}\ln\left(\sum_{\boldsymbol{h}[m]}\tilde{P}(\boldsymbol{o}[m],\boldsymbol{h}[m]\mid\boldsymbol{\theta})\right)-\ln Z.\end{aligned}$$

- Computing the gradient

$$\frac{\partial}{\partial\theta_i}\frac{1}{M}\ell(\boldsymbol{\theta}:\mathcal{D})=\frac{1}{M}\left[\sum_{m=1}^{M}\boldsymbol{E}_{\boldsymbol{h}[m]\sim P(\mathcal{H}[m]\mid\boldsymbol{o}[m],\boldsymbol{\theta})}[f_i]\right]-\boldsymbol{E}_{\boldsymbol{\theta}}[f_i].$$

- Difficulty now is that we need to apply inference for every instance separately
- EM: similar to BN case but M-step requires an iterative optimization.
  – We skip details of the M-step

Structure Learning

- How can we learn the structure (links) of the graphical model from examples?
- Constraint-based approach
  - Find conditional independences from statistical analysis of the data and construct a graph from them
  - Large search space; difficulties caused by noise and lack of sufficient quantity of data
- Score Structures
  - Search through the space of possible graphs
  - Add edges, one at a time and evaluate; for HMMs, add states, one at a time and evaluate
  - Likelihood will always increase with complexity of the structure, resulting in a fully connected graph (one state per time instance in HMMs)
  - Balance likelihood and complexity of graph (*regularize*)

Constraint-Based Approach

- Find conditional independences from statistical analysis of the data and construct a graph from them
- How many conditional independences should we evaluate?
    - Every variable compared to another, given values of any of the subsets of all other variables
- Build a class PDAG (I-map equivalence class)
    - Bound the in-degree of the network
    - Algorithm is straight-forward, we skip details
- Evaluating independence is difficult in presence of noise or lacking sufficient quantity of data. To test if X and Y are independent given Z, evaluate the following:

$$d_{\chi^2}(\mathcal{D}) = \sum_{x,y,z} \frac{(M[x,y,z] - M \cdot \hat{P}(z)\hat{P}(x \mid z)\hat{P}(y \mid z))^2}{M \cdot \hat{P}(z)\hat{P}(x \mid z)\hat{P}(y \mid z)}$$

Likelihood Score of a Structure

- Maximum likelihood score: max likelihood given graph structure

  $\text{score}_L\,(G : D) = l\,(\boldsymbol{\theta}_G : D)$
- Consider case of two variables X and Y
- Difference in score between connecting and not connecting is given by

$$\text{score}_L(\mathcal{G}_1 \ : \ \mathcal{D}) - \text{score}_L(\mathcal{G}_0 \ : \ \mathcal{D}) = M \sum_{x,y} \hat{P}(x,y) \log \frac{\hat{P}(y \mid x)}{\hat{P}(y)} = M \cdot \boldsymbol{I}_{\hat{P}}(X;Y),$$

- Ip is "mutual information"
- Generalizes to BNs, score decomposes over "families"

$$\text{score}_L(\mathcal{G} \ : \ \mathcal{D}) = M \sum_{i=1}^{n} \boldsymbol{I}_{\hat{P}}(X_i;\text{Pa}^{\mathcal{G}}_{X_i}) - M \sum_{i=1}^{n} \boldsymbol{H}_{\hat{P}}(X_i).$$

- Note that the entropy term does not change. Adding an edge will always improve the score (unless true conditional independence is observed) so we can end up with fully connected graph
- Must balance by "regularizing" with complexity of graph

Bayesian Score

- As in Bayesian parameter learning, all parameters are distributions rather than single values
- We can define $\text{score}_B(G:D) = \log P(D|G) + \log P(G)$
  - D is the training data, G is graph structure
  - P(G) is a prior probability of structure; could be a function of number of edges or some other measure of graph complexity
- Better behaved then likelihood score, optimal is not necessarily the fully connected graph
- It can be shown that, with Dirichlet priors, as the number of samples tends to infinity,

$$\log P(D|G) = \ell(\theta_G : D) - (\log M)/2 \, \text{Dim}[G] + O(1)$$

- Define BIC (Bayesian Information Criterion) score as:

$$\text{score}_{BIC}(G:D) = \ell(\theta_G : D) - (\log M)/2 \, \text{Dim}[G]$$

Dim is number of independent parameters in G,

Bayesian Score for BN

- BIC score decomposes over a “family” (child node and parents)
  - Makes it easier to optimize over small parts of the network
- Structure search
  - In special cases (trees, known order) efficient solutions that optimize BIC can be found
  - In general case, we need to conduct a search over all structures
    - In general, problem of finding max BIC score is NP-hard

Tree Structure Graphs

- Tree structures can be learned efficiently
- Complexity of the tree structure is constant so we only need to maximize the likelihood.
- Start with a completely connected graph of all the variable nodes in the problem set
  - Let weight of the edge between nodes X and Y be the *mutual information* between the two variables.
    - $I\,(X{:}Y) = \Sigma_x \Sigma_y\, P(x,y) * \log\, (P(x,y)/(P(x)\; P(y))))\}$
- Can be shown that the maximal spanning tree of this graph provides the MLE tree structure (Chow-Liu Algorithm)
  - More efficient algorithm for undirected tree; can then convert to directed tree where parameter learning is more efficient
- Forest: multiple trees, must use a regularizing term or we will get a tree as the solution

General Graphs

- In general case, we need to conduct a search over all structures
  - Operations are edge addition, edge deletion and edge reversal
  - In general, problem of finding max BIC score is NP-hard
    - Use heuristic methods
    - Limit the search space: part of the structure can be specified by a user
- Bayesian approach would also suggest use of multiple (all) graphs and averaging the results
  - Beyond scope of our course

HMM Structure

- Structure here essentially consists of selecting number of states
  - May also not allow some transitions, so terms in transition matrix are zero
- We can start with a fixed number, apply EM algorithm and compute best likelihood score (we only get a local optimum)
- Add states and recompute likelihood
- As for a general BN, adding states will always improve likelihood (unless we get stuck in a local optimum) so a regularizing term is needed. We may also stop if rate of increase becomes “small”
- Such “tuning” is common in many applications where the states may not have explicit, semantic meaning