Readings: K&F 11.4, 11.5, 20.1, 20.2, 20.3, 20.4

Approximate Inference & Learning undirected Models

Lecture 17 – May 23, 2011
CSE 515, Statistical Methods, Spring 2011

Instructor: Su-In Lee
University of Washington, Seattle

Outline

- Approximate Inference
  - Inference as optimization
  - Generalized Belief Propagation
  - Propagation with approximate messages ←
    - Factorized messages
    - Approximate message propagation
  - Structured variational approximations
- Learning Undirected Models

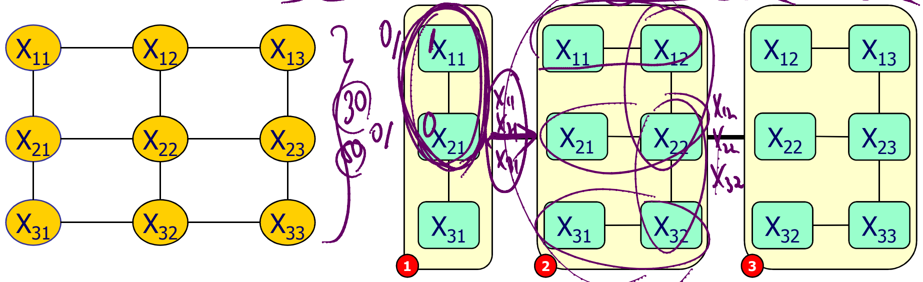
Propagation w. Approximate Msgs

- General idea
  - Perform BP (or GBP) as before, but propagate messages that are only approximate
  - Modular approach
    - General inference scheme remains the same
    - Can plug in many different approximate message computations

Factorized Messages

- Keep internal structure of the cliques in the tree
- Calibration involves sending messages that are joint over three variables
- Idea: simplify messages using factored representation
  - Example: $\tilde{\delta}_{1\to2}[X_{11}, X_{21}, X_{31}] = \tilde{\delta}_{1\to2}[X_{11}]\tilde{\delta}_{1\to2}[X_{21}]\tilde{\delta}_{1\to2}[X_{31}]$

$\tilde{\delta}_{1\to2}[X_{11}, X_{21}] \cdot \tilde{\delta}_{1\to2}[X_{31}]$

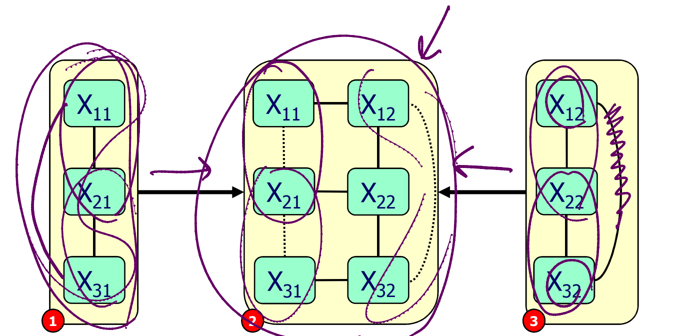


Markov network

Clique tree

Computational Savings 1/2

- Answering queries in Cluster 2
  - Exact inference: $\pi_2 = \pi_2^0 \cdot \delta_{1\to2} \cdot \delta_{3\to2}$
    - Exponential in joint space of cluster 2 (6 variables)



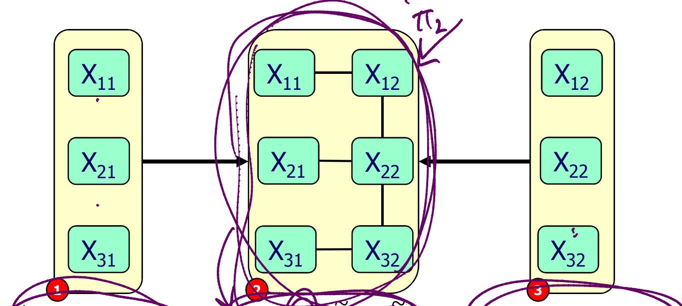
$\delta_{1\to2}[X_{11}, X_{21}, X_{31}]$ $\pi_2 = \pi_2^0 \cdot \delta_{1\to2} \cdot \delta_{3\to2}$ $\delta_{3\to2}[X_{12}, X_{22}, X_{32}]$

5

Computational Savings 2/2

- Answering queries in Cluster 2
  - Exact inference: $\pi_2 = \pi_2^0 \cdot \delta_{1\to2} \cdot \delta_{3\to2}$
    - Exponential in joint space of cluster 2 (6 variables)
  - Approximate inference with factored messages
    - Notice that subnetwork with factored messages is a tree
    - Perform efficient exact inference on subtree to answer queries

$\tilde{\pi}_2$

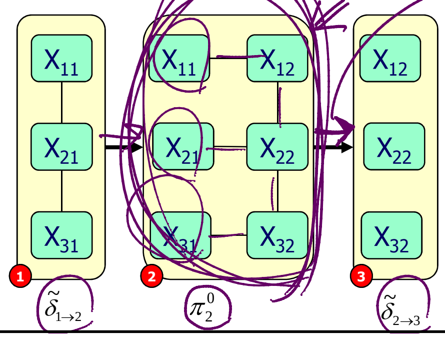


$\tilde{\delta}_{1\to2}[X_{11}, X_{21}, X_{31}]$ $\tilde{\pi}_2 = \pi_2^0 \cdot \tilde{\delta}_{1\to2} \cdot \tilde{\delta}_{3\to2}$ $\tilde{\delta}_{3\to2}[X_{12}, X_{22}, X_{32}]$

6

Factor Sets

- A factor set $\phi=\{\phi_1,\ldots,\phi_k\}$ provides a compact representation for high-dimensional factor $\phi_1\times,\ldots,\times\phi_k$
- Belief propagation $\delta_{2\to3}$ ϕ_1 ϕ_2
  - Multiplication of factor sets $\pi_2 = \pi_2^0 \delta_{1\to2}$
    - Easy: simply the union of the factors in each factor set multiplied
  - Marginalization of factor set: inference in simplified network
    - Example: compute $\delta_{2\to3}$



$$\tilde{\delta}_{2\to3} = \rho\left(\sum_{X_{11},X_{21},X_{31}} \pi_2^0 \cdot \tilde{\delta}_{1\to2}\right)$$

$$= \tilde{\delta}_{2\to3}[X_{12}]\tilde{\delta}_{2\to3}[X_{22}]\tilde{\delta}_{2\to3}[X_{32}]$$

M-projection

$$P(X_1,X_2,X_3) \approx P(X_1)P(X_2)P(X_3)$$

Global Approximate Inference

- Inference as optimization
- Generalized Belief Propagation
  - Define algorithm
  - Constructing cluster graphs
  - Analyze approximation guarantees
- Propagation with approximate messages
  - Factorized messages
  - Approximate message propagation
- Structured variational approximations

Approximate Message Propagation

- Input
  - Clique tree (or cluster graph) GBP
  - Assignments of original factors π^0 to clusters/cliques
  - The factorized form of each sepset
    - Can be represented by a network for each edge C_i—C_j that specifies the factorization (in previous examples we assumed empty network) μ_i π_i
- Two strategies for approximate message propagation
  - Sum-product message passing scheme
  - Belief update messages

Sum-Product Propagation

- Same propagation scheme as in exact inference
  - Select a root
  - Propagate messages towards the root
    - Each cluster collects messages from its neighbors and sends outgoing messages when possible
  - Propagate messages from the root
- Each message passing performs inference on cluster

$$\delta_{i\to j}[X] = \prod_k \delta_{i\to j}[X_k]$$

- Terminates in a fixed number of iterations
- Note: final marginals at each variable are not exact

Message Passing: Belief Propagation

- Same as BP but with approximate messages
- Initialize the clique tree
  - For each clique C_i set $\tilde{\pi}_i \leftarrow \prod_{\phi:\alpha(\phi)=i} \phi$
  - For each edge C_i—C_j set $\tilde{\mu}_{i,j} \leftarrow 1$
- While unset cliques exist
  - Select C_i—C_j
  - Send message from C_i to C_j
    - Marginalize the clique over the sepset $\tilde{\sigma}_{i\rightarrow j} \leftarrow \rho\left(\sum_{C_i - S_{i,j}} \tilde{\pi}_i\right)$
    - Update the belief at C_j $\tilde{\pi}_j \leftarrow \tilde{\pi}_j \dfrac{\tilde{\sigma}_{i\rightarrow j}}{\tilde{\mu}_{i,j}}$
    - Update the sepset at C_i–C_j $\tilde{\mu}_{i,j} \leftarrow \tilde{\sigma}_{i\rightarrow j}$

Approximation

$\pi_i \delta_i$

$\boxed{\text{Two message passing schemes differ in approximate inference}}$

Global Approximate Inference

- Inference as optimization
- Generalized Belief Propagation
  - Define algorithm
  - Constructing cluster graphs
  - Analyze approximation guarantees
- Propagation with approximate messages
  - Factorized messages
  - Approximate message propagation
- Structured variational approximations $\leftarrow Q$

$Q = \{\mu_{ij}, \pi_i\}$

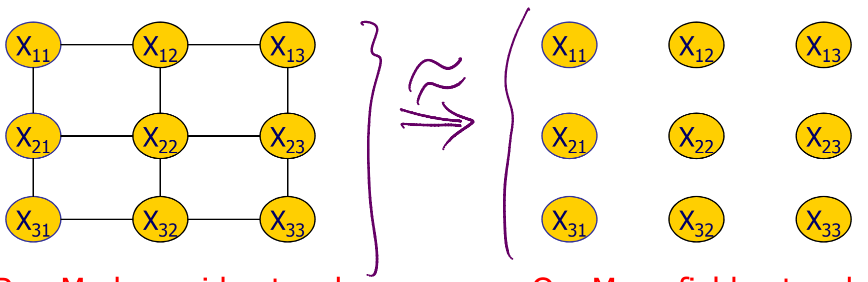
Structured Variational Approx.

- Select a simple family of distributions **Q**
- Find Q∈**Q** that maximizes $F[P_F, Q]$ $-D(Q \| P_F)$

Q

Mean Field Approximation

- $Q(x) = \prod Q(X_i)$
- Q loses much of the information of P_F
- Approximation is computationally attractive
  - Every query in Q is simple to compute
  - Q is easy to represent



P_F – Markov grid network

Q – Mean field network

Mean Field Approximation

- The energy functional is easy to compute, even for networks where inference is complex
  - The energy functional for a fully factored distribution Q can be rewritten simply as a sum of expectations, each one over a small set of variables.

$Q(x_i)$

ϕ

$$F[P_F, Q] = \sum_{\phi \in F} E_Q[\ln \phi] + H_Q(\mathbf{U})$$

$$E_Q[\ln \phi] = \sum_{\mathbf{u}_\phi} Q(\mathbf{u}_\phi) \ln \phi(\mathbf{u}_\phi) = \sum_{\mathbf{u}_\phi} \left(\prod_{X_i \in \mathbf{u}_\phi} Q(x_i) \right) \ln \phi(\mathbf{u}_\phi)$$

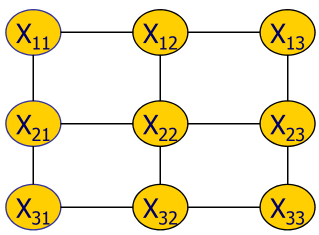
$$H_Q(\mathbf{U}) = \sum_i H_Q(X_i)$$

$Q = \prod Q(x_i)$

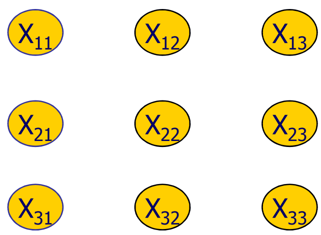
  - The complexity of this expression depends on the size of the factors in P_F and not on the topology of the network.

Mean Field Maximization

- Maximizing the Energy Functional of Mean-Field
  - Find $Q(x) = \Pi Q(X_i)$ that maximizes $F[P_F, Q]$ Q
  - Subject to for all i: $\Sigma_{x_i} Q(x_i) = 1$



P_F – Markov grid network



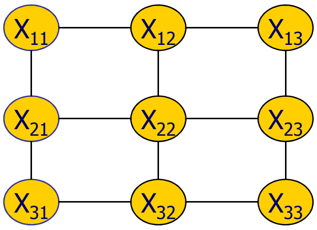
Q – Mean field network

Mean Field Maximization

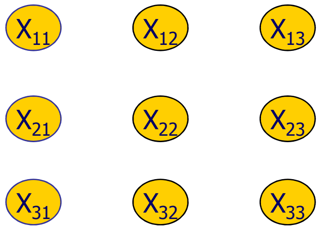
- Theorem: Q(X$_i$) is a local maximum of the mean field given Q(X$_1$),…Q(X$_{i-1}$),Q(X$_{i+1}$),…Q(X$_n$) if and only if

$$Q(x_i) = \frac{1}{Z_i}\exp\left\{\sum_{\phi\in F} E_Q[\ln\phi \mid x_i]\right\}$$

  - Proof in K&F on pages 451-452



P$_F$ – Markov grid network



Q – Mean field network

Mean Field Maximization: Intuition

- We can rewrite $Q(x_i) = \frac{1}{Z_i}\exp\left\{\sum_{\phi\in F} E_Q[\ln\phi \mid x_i]\right\}$ as:

$P_F = \frac{1}{Z}\prod \phi$

$$Q(x_i) = \frac{1}{Z_i}\exp\{E_Q[\ln P_F(x_i \mid \mathbf{X}_{-i})]\}\boxed{\exp\{E_Q[\ln Z P_F(\mathbf{X}_{-i})]\}}$$

Doesn't depend on x$_i$

This constant can be "absorbed" into the normalizing function.

$$Q(x_i) = \frac{1}{\tilde{Z}_i}\exp\{E_Q[\ln P_F(x_i \mid \mathbf{X}_{-i})]\}$$

- Q(x$_i$) is the geometric average of P$_F$(x$_i$|**X$_{-i}$**)
  - Relative to the probability distribution Q
  - In this sense, marginal is "consistent" with other marginals
- In P$_F$ we can also represent marginals

$X_{-i} = X - \{X_i\}$

$$P_F(x_i) = \sum_{\mathbf{x}_{-i}} P_F(\mathbf{x}_{-i})P_F(x_i \mid \mathbf{x}_{-i}) = E_{P_F}[P_F(x_i \mid \mathbf{x}_{-i})]$$

  - Arithmetic average with respect to P$_F$

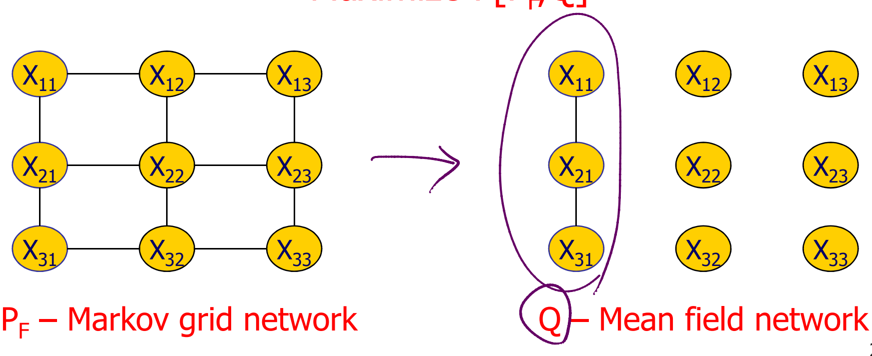
Mean Field: Algorithm

- Since terms that do not involve x_i can be "absorbed" into the normalization constant,
- Simplify: $$Q(x_i) = \frac{1}{Z_i}\exp\left\{\sum_{\phi\in F} E_Q[\ln\phi \mid x_i]\right\}$$
- To: $$Q(x_i) = \frac{1}{Z_i}\exp\left\{\sum_{\phi: X_i\in Scope(\phi)} E_Q[\ln\phi(U_\phi, x_i)]\right\}$$
  - Note: $Q(x_i)$ does not appear on right hand side
    - Can solve and reach optimal $Q(x_i)$ in one step
    - Note: step is only optimal given all other $Q(X_j)$ $(j\neq i)$
    - Suggests an iterative algorithm: in each step, find the optimal $Q(x_i)$, given all the other $Q(X_j)$ $(j\neq i)$
    - Convergence guaranteed to local maxima since each step improves $F[P_F,Q]$ $Q(x_i)$ $P_F \approx \prod Q(x_i)$

Structured Approximations

- Can use Q that are increasingly complex $Q = \prod Q(x_i)$
- As long as Q is easy (=inference feasible) efficient update equations can be derived

Maximize $F[P_F,Q]$



P_F – Markov grid network

Q – Mean field network

LEARNING UNDIRECTED GRAPHICAL MODELS

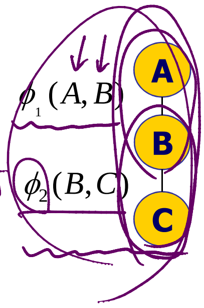
Learning Undirected Graphs

- The likelihood function
  - Log-linear representation
  - Properties of the likelihood function
- Learning parameters (weights)
  - Maximum likelihood estimation
  - Generatively vs Discriminatively
- Learning with alternative goals
- Learning with incomplete data
- Learning structure (features)

The Likelihood Function 1/2

- Consider the very simple network, parameterized by two potentials $\phi_1(A,B)$ and $\phi_2(B,C)$

$\ln P(A,B,C) = \ln \frac{1}{Z}\phi_1(A,B)\cdot\phi_2(B,C)$



- The log-likelihood of an instance <a,b,c> :

$$\ln P(a,b,c) = \ln \phi_1(a,b) + \ln \phi_2(b,c) - \ln Z$$

  - where Z is the partition function that ensures the distribution sums up to 1.

- Now, consider the log-likelihood function for a data set D containing M instances: $\langle a[m], b[m], c[m] \rangle$ $m=1..M$

$\pi \to \Sigma$

$$l(\boldsymbol{\theta}:D) = \sum_m \left[\ln \phi_1(a[m],b[m]) + \ln \phi_2(b[m],c[m]) - \ln Z(\boldsymbol{\theta})\right]$$

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

The Likelihood Function 2/2

$$l(\boldsymbol{\theta}:D) = \sum_{a,b} M[a,b] \ln \phi_1(a,b) + \sum_{b,c} M[b,c] \ln \phi_2(b,c) - M \ln Z(\boldsymbol{\theta})$$

- Sufficient statistics that summarize the data: the joint counts M[a,b], M[b,c] in D
- The first and second term involves ϕ_1 and ϕ_2 alone, respectively.
- The third term is the log-partition function ln Z, where

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

$\sum_{a,b,c} P(A,B,C) = \sum \frac{1}{Z}\phi_1\phi_2 = 1$

  - ln Z is a function of both ϕ_1 and ϕ_2; it couples the two potentials in the likelihood function.
  - Consider MLE: In BNs, we could estimate each parameter independently of the other ones. Here, when changing ϕ_1, Z changes, possibly changing the value of ϕ_2 that maximizes ln Z(**Θ**). → In MNs, we cannot estimate each parameter independently.

Log-Linear Model 1/2

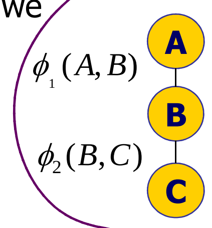
- Given a set of features $F=\{f_i(\mathbf{D}_i)\}_{i=1,\dots,k}$, where $f_i(\mathbf{D}_i)$ is a feature function defined over the variables in $\mathbf{D}_i$, we have:

$\phi.$

$$P(X_1,\dots,X_n:\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})}\exp\left\{\sum_{i=1}^{k}\theta_i f_i(\mathbf{D}_i)\right\}$$

$\prod \phi_i$

  - For example, in the previous example, we can define a set of features as:



$$f_1(A,B)=\begin{cases}1 & \text{, when } A=a^1 \text{ and } B=b^1\\ 0 & \text{, otherwise}\end{cases}$$

$$f_2(A,B)=\begin{cases}1 & \text{, when } A=a^1 \text{ and } B=b^0\\ 0 & \text{, otherwise}\end{cases}$$

:

- Let D be a data set of M instances $D=\{\xi[1],\dots,\xi[M]\}$, and let $F=\{f_1,\dots,f_K\}$ be a set of features that define a model:

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

Log-Linear Model 2/2

f_i

$$l(\boldsymbol{\theta}:D)=\sum_i \theta_i\left(\sum_m f_i(d[m])\right) - M\ln Z(\boldsymbol{\theta})$$

$A=a^1 \; B=b^0$

- Sufficient statistics: sums of the feature values in the instances in D
- Dividing it by the number of instances M,

$$\frac{1}{M}l(\boldsymbol{\theta}:D)=\sum_i \theta_i \mathbf{E}_D[f_i[\mathbf{d}_i]] - \ln Z(\boldsymbol{\theta})$$

  - where $\mathbf{E}_D[f_i(\mathbf{d}_i)]$ is the empirical expectation of f_i, that is, its average frequency in the data set.

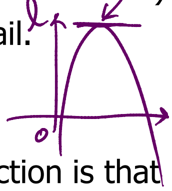
Properties of the Likelihood Function

- The likelihood function is θ a sum of two functions.

$$l(\boldsymbol{\theta} : D) = \sum_i \theta_i \underbrace{\left(\sum_m f_i(\xi[m]) \right)}_{\theta} - \underbrace{M \ln Z(\boldsymbol{\theta})}_{\theta}$$

  - The first function is linear in the parameters (increasing the parameters directly increases this term)
  - Let's examine the second term in more detail.

$$\ln Z(\boldsymbol{\theta}) = \ln \sum_{\xi} \exp\left\{ \sum_i \theta_i f_i(\xi) \right\}$$



  - One important property of the partition function is that it is convex in the parameters **Θ**.
  - Proof? The Hessian – the matrix of the function's second derivatives – is positive semidefinite.
- The likelihood function is ~~convex~~ concave. in **Θ**

Learning Undirected Graphs

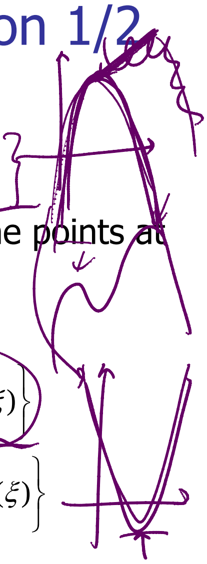
- The likelihood function
  - Log-linear representation
  - Properties of the likelihood function
- Learning parameters
  - Maximum likelihood estimation
  - Generatively vs Discriminatively
- Collective classification with HMM, MEMM, CRF
- Learning with incomplete data
- Learning structure (features)
- Learning with alternative objectives

Maximum Likelihood Estimation 1/2

- The average likelihood is

log

$$\frac{1}{M} l(\boldsymbol{\theta} : D) = \sum_i \theta_i \mathbf{E}_D[f_i[\mathbf{d}_i]] - \ln Z(\boldsymbol{\theta})$$



- For a concave function, the maxima are the points at which the gradient is 0

$$\frac{\partial}{\partial \theta_i} \sum_j \theta_j \mathbf{E}_D[f_j[\mathbf{d}_j]] = \mathbf{E}_D[f_i[\mathbf{d}_i]]$$

$$\frac{\partial}{\partial \theta_i} \ln Z(\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \sum_\xi \frac{\partial}{\partial \theta_i} \exp\left\{\sum_i \theta_i f_i(\xi)\right\}$$

$$= \sum_\xi f_i(\xi) \frac{1}{Z(\boldsymbol{\theta})} \exp\left\{\sum_i \theta_i f_i(\xi)\right\}$$

$$= \mathbf{E}_{\boldsymbol{\theta}}[f_i]$$

- The gradient is $\frac{\partial}{\partial \theta_i} \frac{1}{M} l(\boldsymbol{\theta} : D) = \mathbf{E}_D[f_i[\mathbf{d}_i]] - \mathbf{E}_{\boldsymbol{\theta}}[f_i]$

29

Maximum Likelihood Estimation 2/2

- The gradient is

p(Y=y)

$$\frac{\partial}{\partial \theta_i} l(\boldsymbol{\theta} : D) = M\mathbf{E}_D[f_i[\mathbf{d}_i]] - M\mathbf{E}_{\boldsymbol{\theta}}[f_i]$$

Number of times feature f_i is true in data D

Expected number of times feature f_i is true according to model $\boldsymbol{\theta}$

- The MLE of parameters $\hat{\boldsymbol{\theta}}$ satisfies, for all i,

$$\mathbf{E}_D[f_i[\mathbf{d}_i]] = \mathbf{E}_{\hat{\boldsymbol{\theta}}}[f_i]$$

$f_i \sim$ Area! $\theta_i = 6^0$

$$\frac{\partial}{\partial \theta_i} l(\theta : D)\Big|_{\theta = \hat{\theta}} = 0$$

- Numerical optimization: gradient ascent method or 2nd order-based (Newton's method)
  - Requires inference at each step (slow!)

MLE

30

Conditionally Trained Models 1/2

- We often want to use a Markov network to perform a particular inference task, where we have a known set of observed variables **X** and a predetermined set of variables **Y** that we want to query.

X Y P(X,Y)

- Discriminative training
  - We train the network as a conditional random field (CRF) that encodes a conditional distribution P(**Y**|**X**) P(Y,X)
  - Training the model encoding P(**Y**,**X**) : generative training
- Given the training data consisting of pairs D={(**y**[m],**x**[m])}$_{m=1,\dots,M}$, specifying assignments to **Y** and **X**, an appropriate objective function to use in this situation is the conditional likelihood.

$$l_{\mathbf{Y}|\mathbf{X}}(\boldsymbol{\theta}:D) = \ln P(\mathbf{y}[1,\dots,M] \mid \mathbf{x}[1,\dots,M],\boldsymbol{\theta})$$

P(y[1..M], x[1..M])

$$= \sum_{m=1}^{M} \ln P(\mathbf{y}[m] \mid \mathbf{x}[m],\boldsymbol{\theta})$$

31

Conditionally Trained Models 2/2

- The gradient is

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

Number of times feature f_i is true in data D

Expected number of times feature f_i is true according to model

  - Deceptively similar to the generative training case!
  - Key difference: Expected counts (2[nd] term) are computed as the summation of counts in M models defined by the different values of the conditioning variables **x**[m].

$E_\theta[f_i]$ $E_\theta[f_i \mid X[m]]$

- Inference: In generative training, each gradient step required only a single execution of inference. When training CRFs, we must execute inference for every single training instance m, conditioning on **x**[m]
  - The inference is executed on a simpler model, because conditioning on evidence in a Markov network can only reduce the computational cost.

32

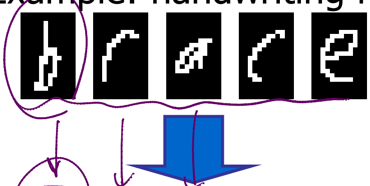
Learning Undirected Graphs

- The likelihood function
  - Log-linear representation
  - Properties of the likelihood function
- Learning parameters
  - Maximum likelihood estimation
  - Generatively vs Discriminatively
- Collective classification with HMM, MEMM, CRF
- Learning with incomplete data
- Learning structure (features)
- Learning with alternative objectives

33

Collective Classification

- Taking a set of interrelated instances and jointly labeling them
- Example: handwriting recognition



x *A sequence of observations*

- Use local information
- Exploit correlations

b r a c e

y *Label them with some joint label*

- Let's discuss some of the trade-offs between different models that one can apply to this task.
  - We focus on the context of labeling instances organized in a sequence (HMM, MEMM, CRF)

34

Acknowledgement

- These lecture notes were generated based on the slides from Prof Eran Segal.