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: $\bar{\delta}_{i}([X_{11}, X_{21}, X_{12}])\bar{\delta}_{i-1}([X_{22}, X_{32}])\bar{\delta}_{i-2}([X_{33}, X_{13}])$
Computational Savings 1/2

- Answering queries in Cluster 2
  - Exact inference:
    \[ \pi_2 = \delta_{1\rightarrow 2} \cdot \delta_{3\rightarrow 2} \cdot \delta_{2\rightarrow 2} \]
    - Exponential in joint space of cluster 2 (6 variables)

Computational Savings 2/2

- Answering queries in Cluster 2
  - Exact inference:
    \[ \pi_2 = \pi_2^0 \cdot \delta_{1\rightarrow 2} \cdot \delta_{3\rightarrow 2} \]
    - 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
Factor Sets

- A factor set \( \phi = \{ \phi_1, \ldots, \phi_k \} \) provides a compact representation for high-dimensional factor \( \phi_1 \times \cdots \times \phi_k \).

- **Belief propagation**
  - Multiplication of factor sets: easy, simply the union of the factors in each factor set multiplied.

  **Example:** compute \( \delta_{2,3} \rightarrow \delta \)

```
\[
\begin{align*}
\delta_{2,3} &= \sum_{x_{11} x_{12} x_{21} x_{22} x_{31} x_{32}} \delta_{1,3} \pi_{23} \delta_{1,2} \\
\pi_{23} &= p(x_{21} x_{22} x_{31} x_{32}) \approx p(x_{11}) p(x_{21}) p(x_{22})
\end{align*}
\]
```

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)
  - Assignments of original factors $\pi^0$ to clusters/cliques
  - The factorized form of each sepset
    - Can be represented by a network for each edge $C_i \rightarrow C_j$ that specifies the factorization (in previous examples we assumed empty network)

- 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
  \[
  \delta_{i,j}[X] = \prod_k \delta_{i,j}[X_k]
  \]

- 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 $\bar{\pi}_i \leftarrow \prod_{v \in \partial(i) \rightarrow (i)} \phi_v$
  - For each edge $C_i - C_j$ set $\mu_{ij} \leftarrow 1$
- While unset cliques exist
  - Select $C_i - C_j$
  - Send message from $C_i$ to $C_j$
    - Marginalize the clique over the sepset $\hat{\bar{\sigma}}_{i,j} \leftarrow \rho \left( \sum_{C_{-i,j}} \bar{\pi}_i \right)$
    - Update the belief at $C_j$ $\bar{\pi}_j \leftarrow \bar{\pi}_j \frac{\hat{\bar{\sigma}}_{j,i}}{\mu_{ij}}$
    - Update the sepset at $C_i - C_j$ $\mu_{ij} \leftarrow \hat{\bar{\sigma}}_{i,j}$

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
Structured Variational Approx.
- Select a simple family of distributions $Q$
- Find $Q \in Q$ that maximizes $F[P_F, Q]$

Mean Field Approximation
- $Q(x) = \Pi Q(X_i)$
- $Q$ 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.

\[
F[P_F, Q] = \sum_{\theta \in \Phi} E_{\theta}[\ln \phi] - H_Q(U)
\]

\[
E_{\theta}[\ln \phi] = \sum_{\phi} Q(\theta) \ln \phi(\theta) = \sum_{\theta} \prod_{X_i = \theta_i} Q(x_i) \ln \phi(\theta)
\]

\[
H_Q(U) = \sum_{\phi} H_Q(\phi)
\]

- 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 \( Q(x) = \prod Q(x_i) \) that maximizes \( F[P_F, Q] \)
  - Subject to for all \( i \): \( \sum Q(x_i) = 1 \)
Mean Field Maximization

- Theorem: $Q(x_i)$ is a local maximum of the mean field given $Q(x_1), \ldots, Q(x_{i-1}), Q(x_{i+1}), \ldots, Q(x_n)$ if and only if

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

- Proof in K&F on pages 451-452

---

Mean Field Maximization: Intuition

- We can rewrite $Q(x_i)$ as:

$$Q(x_i) = \frac{1}{Z_i} \exp \{ E_Q[\ln P_F(x_i | x_{-i})] \} \exp \{ E_Q[\ln Z P_F(x_{-i})] \}$$

- $Q(x_i)$ 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

$$P_F(x_i) = \sum_{x_{-i}} P_F(x_i | x_{-i}) = E_P[P_F(x_i | x_{-i})]$$

- 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: 
  $$Q(x_i) = \frac{1}{Z} \exp \left\{ \sum_{\phi \in F} E_\phi [\ln \phi(x_i)] \right\}$$
- To: 
  $$Q(x_i) = \frac{1}{Z} \exp \left\{ \sum_{\phi \in \phi(X \neq i)} E_\phi [\ln \phi(U_{\phi}, x_i)] \right\}$$
- 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 $F[P, Q]$

Structured Approximations

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

Maximize $F[P, Q]$
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 $\phi_2(B, C)$.
- The log-likelihood of an instance $\langle a, b, c \rangle$:
  \[ \ln P(a, b, c) = \ln \phi_1(a, b) + \ln \phi_2(b, c) - \ln Z \]
  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:
  \[ l(\theta : D) = \sum_m \left[ \ln \phi_1(a[m], b[m]) + \ln \phi_2(b[m], c[m]) - \ln Z(\theta) \right] \]
  \[ = \sum_{a,b} M[a,b] \ln \phi_1(a,b) + \sum_{b,c} M[b,c] \ln \phi_2(b,c) - M \ln Z(\theta) \]

The Likelihood Function 2/2

- Sufficient statistics that summarize the data: the joint counts $M[a,b], M[b,c]$ in $D$.
- The first and second term involves $\phi_1$ and $\phi_2$ alone, respectively.
- The third term is the log-partition function $\ln Z$, where
  \[ Z(\theta) = \sum_{a,b} \phi_1(a,b) \phi_2(b,c) \]
  is a function of both $\phi_1$ and $\phi_2$; 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 $\phi_1$, $Z$ changes, possibly changing the value of $\phi_2$ that maximizes $\ln Z(\theta)$. → In MNs, we cannot estimate each parameter independently.
Log-Linear Model 1/2

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

$$P(X_1,...,X_n : \theta) = \frac{1}{Z(\theta)} \exp \left\{ \sum_{i=1}^k \theta_i f_i(D_i) \right\} \prod_i \phi_i$$

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

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

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

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

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

Log-Linear Model 2/2

- **Sufficient statistics**: sums of the feature values in the instances in $D$

- Dividing it by the number of instances $M$,

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

- where $\mathbb{E}_D[f_i(d)]$ 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 a sum of two functions.
  \[
  l(\theta : D) = \sum_i \theta \left( \sum f_i(\xi[m]) \right) - M \ln Z(\theta)
  \]
- The first function is linear in the parameters (increasing the parameters directly increases this term).
- Let’s examine the second term in more detail.
- One important property of the partition function is that it is **convex** in the parameters \( \Theta \).
- Proof? The Hessian – the matrix of the function’s second derivatives – is positive semidefinite.
- The likelihood function is convex in \( \Theta \).

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
  \[ \frac{1}{M} l(\theta : D) = \frac{1}{N} \sum_{i} \theta E_{D}[f_{i}][d_{i}] \ln Z(\theta) \]

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

\[ \frac{\partial}{\partial \theta} \left( \frac{1}{N} \sum_{i} \theta E_{D}[f_{i}][d_{i}] \ln Z(\theta) \right) = \sum_{i} f_{i}(\xi) \frac{1}{Z(\theta)} \exp \left\{ \sum_{i} \theta f_{i}(\xi) \right\} = E_{D}[f_{i}] \]

- The gradient is
  \[ \frac{\partial}{\partial \theta} \frac{1}{M} l(\theta : D) = E_{D}[f_{i}][d_{i}] - E_{\theta}[f_{i}] \]

Maximum Likelihood Estimation 2/2

- The gradient is
  \[ \frac{\partial}{\partial \theta} l(\theta : D) = M E_{D}(f_{i})[d_{i}] - M E_{\theta}[f_{i}] \]

  - Number of times feature \( f_{i} \) is true in data \( D \)
  - Expected number of times feature \( f_{i} \) is true according to model

- The MLE of parameters \( \hat{\theta} \) satisfies, for all \( i \),
  \[ E_{D}[f_{i}][d_{i}] = E_{\hat{\theta}}[f_{i}] \]

- Numerical optimization: gradient ascent method or 2nd order-based (Newton’s method)
  - Requires inference at each step (slow!)
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 $\mathbf{X}$ and a predetermined set of variables $\mathbf{Y}$ that we want to query.

- **Discriminative training**
  - We train the network as a conditional random field (CRF) that encodes a conditional distribution $P(\mathbf{Y}|\mathbf{X})$.
  - Training the model encoding $P(\mathbf{Y},\mathbf{X})$ — generative training.

- Given the training data consisting of pairs $D=${\{(\mathbf{y}[m],\mathbf{x}[m])\}}_{m=1}^{M}$, specifying assignments to $\mathbf{Y}$ and $\mathbf{X}$, an appropriate objective function to use in this situation is the **conditional likelihood**.

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

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

Conditionally Trained Models 2/2

- The gradient is

$$\frac{\partial l_{\mathbf{Y},\mathbf{X}}(\theta : D)}{\partial \theta} \approx \sum_{m=1}^{M} \mathbb{E}_D[f_i(y[m], \mathbf{x}[m])] - \mathbb{E}_\theta[f_i(y[m], \mathbf{x}[m])]$$

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

- **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 $\mathbf{x}[m]$.

  - The inference is executed on a simpler model, because conditioning on evidence in a Markov network can only reduce the computational cost.
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

Collective Classification

- Taking a set of interrelated instances and jointly labeling them

- Example: handwriting recognition

- 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)
Acknowledgement

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