

# Learning with Approximate Inference

Sargur Srihari  
srihari@cedar.buffalo.edu

# Topics

- Learning MN parameters using Gradient Ascent and Belief Propagation
  - Difficulty with exact methods
- Approximate methods
  - Approximate Inference
    - Belief Propagation
    - Sampling-based Learning
    - MAP-based Learning
- Ex: CRF for Protein Structure Prediction

# Likelihood function for a Markov Network

- Log-linear form of MN with parameters  $\theta$  is

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

where  $\theta = \{ \theta_1, \dots, \theta_k \}$  are  $k$  parameters, each associated with a feature  $f_i$  defined over instances of  $D_i$

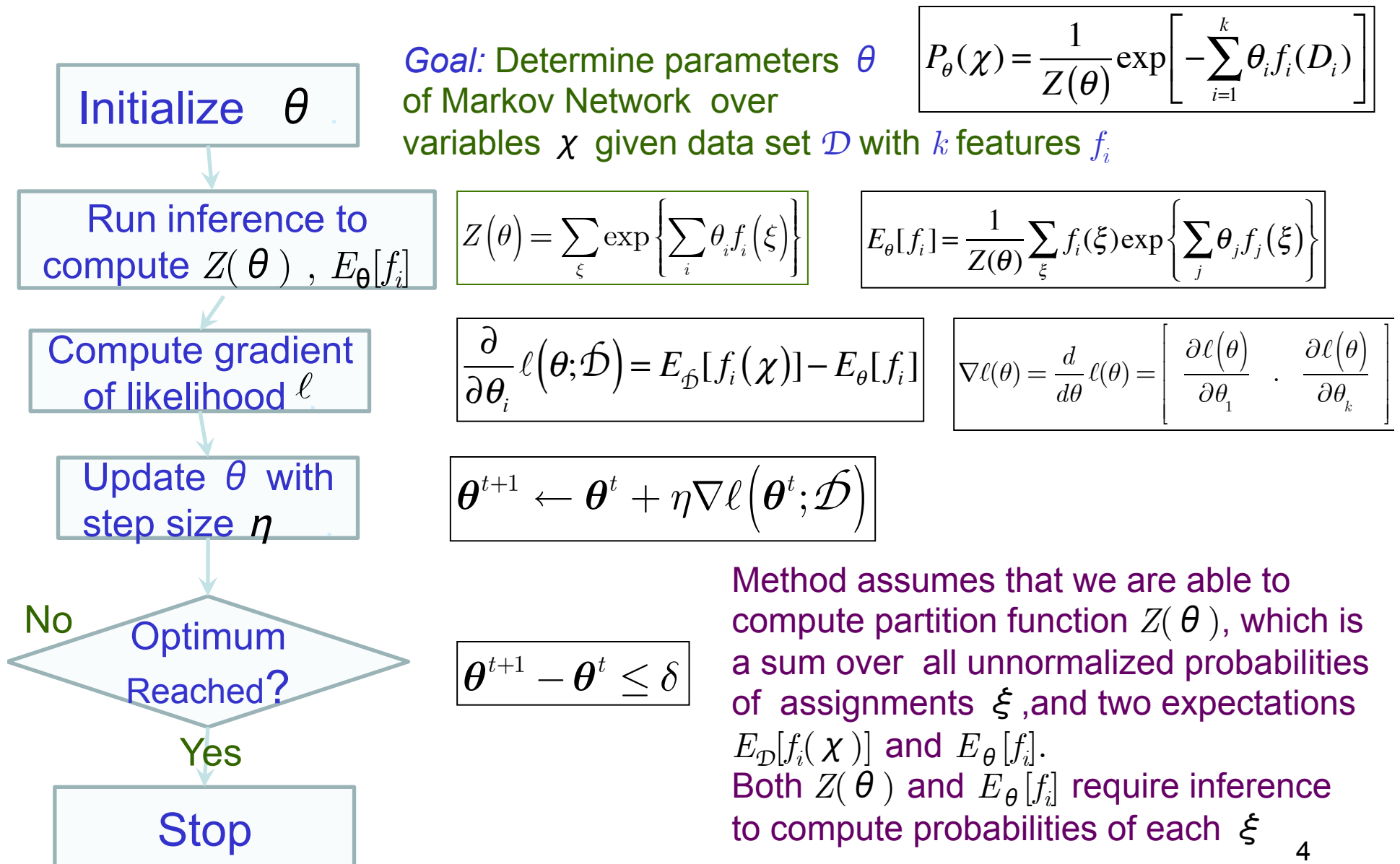
Where the partition function defined as:

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

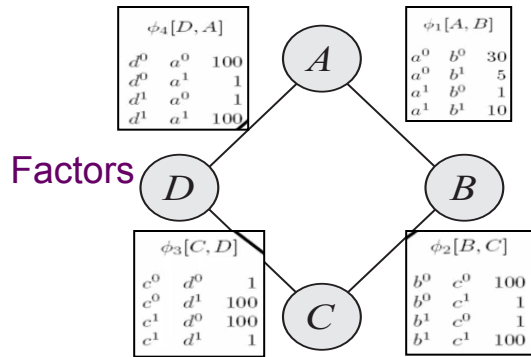
- We wish to find  $\theta$  that maximizes the Log-likelihood

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

# Gradient-based learning of Undirected PGM



# Exact Inference: Belief Propagation



## 1. Gibbs Distribution

$$P(A, B, C, D) = \frac{1}{Z} \phi_1(A, B) \cdot \phi_2(B, C) \cdot \phi_3(C, D) \cdot \phi_4(D, A)$$

where

$$Z = \sum_{A, B, C, D} \phi_1(A, B) \cdot \phi_2(B, C) \cdot \phi_3(C, D) \cdot \phi_4(D, A)$$

$$Z = 7,201,840$$

$$\tilde{P}_{\Phi}(A, B, C, D) = \phi_1(A, B) \phi_2(B, C) \phi_3(C, D) \phi_4(D, A)$$

Unnormalized Distribution

Assignment				Unnormalized
$a^0$	$b^0$	$c^0$	$d^0$	300000
$a^0$	$b^0$	$c^0$	$d^1$	300000
$a^0$	$b^0$	$c^1$	$d^0$	300000
$a^0$	$b^0$	$c^1$	$d^1$	30
$a^0$	$b^1$	$c^0$	$d^0$	500
$a^0$	$b^1$	$c^0$	$d^1$	500
$a^0$	$b^1$	$c^1$	$d^0$	5000000
$a^0$	$b^1$	$c^1$	$d^1$	500
$a^1$	$b^0$	$c^0$	$d^0$	100
$a^1$	$b^0$	$c^0$	$d^1$	1000000
$a^1$	$b^0$	$c^1$	$d^0$	100
$a^1$	$b^0$	$c^1$	$d^1$	100
$a^1$	$b^1$	$c^0$	$d^0$	10
$a^1$	$b^1$	$c^0$	$d^1$	100000
$a^1$	$b^1$	$c^1$	$d^0$	100000
$a^1$	$b^1$	$c^1$	$d^1$	100000

## 2. Clique Tree (triangulated):

Initial Potentials:

Each  $\psi$  has every factor involving its arguments

$$\psi_1(A, B, D) = \phi_1(A, B) \phi_2(B, C) \phi_3(C, D) \phi_4(D, A)$$

$$\psi_2(B, C, D) = \phi_1(A, B) \phi_2(B, C) \phi_3(C, D) \phi_4(D, A)$$

Clique/Cluster  $C_1$

1.  $A, B, D$

Sepset

$\{B, D\}$

Cluster  $C_2$

2.  $B, C, D$

Computing Clique Beliefs ( $\beta_i$ ), Sepset Beliefs ( $\mu_{i,j}$ )

$$\beta_1(A, B, D) = \tilde{P}_{\Phi}(A, B, D) = \sum_C \psi_1(A, B, D) = \sum_C \phi_1(A, B) \phi_2(B, C) \phi_3(C, D) \phi_4(D, A)$$

e.g.,  $\beta_1(a^0, b^0, d^0) = 300,000 + 300,000 = 600,000$

$$\mu_{1,2}(B, D) = \sum_{C_1 - S_{1,2}} \beta_1(C_1) = \sum_A \beta_1(A, B, D)$$

e.g.,  $\mu_{1,2}(b^0, d^0) = 600,000 + 200 = 600,200$

$$\beta_2(B, C, D) = \tilde{P}_{\Phi}(B, C, D) = \sum_A \mu_{1,2}(B, D) \cdot \psi_2(B, C, D) = \sum_A \psi_2(B, C, D)$$

e.g.,  $\beta_2(b^0, c^0, d^0) = 300,000 + 100 = 300,100$

All Clique and Sepset Beliefs

Assignment			max <sub>C</sub>	Assignment			max <sub>A,C</sub>	Assignment			max <sub>B,C,D</sub>
a <sup>0</sup>	b <sup>0</sup>	d <sup>0</sup>	600,000	b <sup>0</sup>	d <sup>0</sup>	600,200	b <sup>0</sup>	c <sup>0</sup>	d <sup>0</sup>	300,100	
a <sup>0</sup>	b <sup>0</sup>	d <sup>1</sup>	300,030		c <sup>0</sup>	d <sup>1</sup>		1,300,130	c <sup>0</sup>	d <sup>1</sup>	300,100
a <sup>0</sup>	b <sup>1</sup>	d <sup>0</sup>	5,000,500		c <sup>1</sup>	d <sup>0</sup>		5,100,510	c <sup>1</sup>	d <sup>0</sup>	100,100
a <sup>0</sup>	b <sup>1</sup>	d <sup>1</sup>	1,000		d <sup>1</sup>	201,000		c <sup>1</sup>	d <sup>1</sup>	5,100,100	
a <sup>1</sup>	b <sup>0</sup>	d <sup>0</sup>	200	b <sup>1</sup>	c <sup>0</sup>	d <sup>0</sup>	100,100	b <sup>1</sup>	c <sup>1</sup>	d <sup>0</sup>	100,100
a <sup>1</sup>	b <sup>0</sup>	d <sup>1</sup>	1,000,100					b <sup>1</sup>	c <sup>1</sup>	d <sup>1</sup>	100,100
a <sup>1</sup>	b <sup>1</sup>	d <sup>0</sup>	100,010								
a <sup>1</sup>	b <sup>1</sup>	d <sup>1</sup>	200,000								

$\beta_1(A,B,D)$ 
 $\mu_{1,2}(B,D)$ 
 $\beta_2(B,C,D)$

$\beta_1(A, B, D)$

$\mu_{1,2}(B, D)$

$\beta_2(B, C, D)$

$$\tilde{P}_{\Phi}(\chi) = \frac{\prod_{i \in V_T} \beta_i(C_i)}{\prod_{(i,j) \in E_T} \mu_{i,j}(S_{i,j})}$$

Verifying Inference

$$\frac{\tilde{P}_{\Phi}(a^1, b^0, c^1, d^0) = 100}{\beta_1(a^1, b^0, d^0) \beta_2(b^0, c^1, d^0)} = \frac{200 \cdot 300 \cdot 100}{600 \cdot 200} = 100$$

# Example of Computing $E_{\mathcal{D}}[f_i(\chi)]$

$\chi = \{A, B, C\}$ ; Pairwise Markov Network 

– Variables are binary

– Three clusters:  $C_1 = \{A, B\}$ ,  $C_2 = \{B, C\}$ ,  $C_3 = \{C, A\}$

• Log-linear model  $P_{\theta}(\chi) = \frac{1}{Z(\theta)} \exp \left[ - \sum_{i=1}^k \theta_i f_i(D_i) \right]$  with two features:

•  $f_{00}(x, y) = 1$  if  $x=0, y=0$  and 0 otherwise and

•  $f_{11}(x, y) = 1$  if  $x=1, y=1$  and 0 otherwise

Both shared over all clusters

– Assume we have three data instances  $(A, B, C)$ :

–  $(0, 0, 0) \rightarrow$  Cluster  $AB$ , Cluster  $BC$ , Cluster  $CA$  have  $f_{00}(x, y) = 1$

–  $(0, 1, 0) \rightarrow$  Only Cluster  $CA$  has  $f_{00}(x, y) = 1$

–  $(1, 0, 0) \rightarrow$  Only Cluster  $BC$  has  $f_{00}(x, y) = 1$

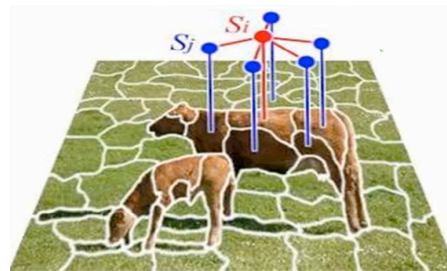
• Unnormalized empirical feature counts, pooled over all clusters:

$$\begin{aligned} E_{\tilde{P}}[f_{00}] &= (3 + 1 + 1) / 3 = 5 / 3 \\ E_{\tilde{P}}[f_{11}] &= (0 + 0 + 0) / 3 = 0 \end{aligned}$$

Can normalize using counts over all features  
i.e.,  $(5/3) + 0$

# Difficulty with Exact Methods

- Exact Parameter Estimation Methods assume ability to compute
  - Partition function  $Z(\theta)$  and
  - Expectations  $E_{p_\theta} [f_i]$
- In many applications structure of network does not allow exact computation of these terms
  - In image segmentation, grid networks lead to exponential size clusters for exact inference



Cluster graph is clique tree with overlapping factors

AB—BC--CA

# Discussion on Approximate Methods

## 1. In this section: approximate inference

- Decouple inference from Learning Parameters
  - Inference is a black-box
- But approximation may interfere with learning
  - Non-convergence of inference can lead to oscillating estimates of the gradient & no learning convergence

## 2. Next section: approximate objective function

- Whose optimization doesn't require much inference
- *Approximately optimizing the likelihood function can be reformulated as exactly optimizing an approximate objective*



# Approximate Inference

- Learning with Approximate Inference methods
  1. Belief Propagation
  2. MAP-based Learning

# Approximate Inference: Belief Propagation

- Popular Approach for Approximate Inference is Belief Propagation and its variants
  - An algorithm from this family would be used for inference in the model resulting from the learning procedure
- We should use the same inference algorithm that will be used for querying it
  - Model trained with same inference algorithm is better than model trained with exact inference!
- BP is run with each iteration of Gradient Ascent to compute expected feature count  $E_{P_\theta} [f_i]$

# Difficulty with Belief Propagation

- In principle, BP is run in each Gradient Ascent iteration
  - to compute  $E_{P_\theta} [f_i]$  used in gradient computation
    - Due to family preservation property each feature must be a subset of a cluster  $C_i$  in the cluster graph
      - Hence to compute  $E_{P_\theta} [f_i]$  we can compute BP marginals over  $C_i$
- But BP often does not converge
  - Marginals derived often oscillate
    - Final results depend on where we stop
  - As a result gradient computed is unstable
    - Hurt convergence properties of gradient descent
    - Even more severe with line search

# Convergent alternatives to Belief Propagation

- We describe three BP methods:
  1. Pseudo- moment matching
    - Reformulate task of learning with approximate inference as optimizing an alternative objective
  2. Maximum Entropy approximation (CAMEL)
    - General derivation that allows us to reformulate ML with BP as a unified optimization problem with an approximate objective
  3. MAP-based Learning
    - Approximate expected feature counts with their counts in the single MAP assignment in current MN

# Pseudo-moment Matching

- Begin with analysis of fixed points in learning
  - Converged BP beliefs must satisfy  $E_{\beta_i(C_i)}[f_{C_i}] = E_D[f_i(C_i)]$ 
    - Or  $\beta_i(c_i^j) = \hat{P}(c_i^j)$  Convergent point is a set of beliefs that match the data
- Define for each sepset  $S_{i,j}$  between  $C_i$  and  $C_j$ 

$\phi_i \leftarrow \frac{\beta_i}{\mu_{i,j}}$

$C_1: A, B, D$

$S_{12} = \{B, D\}$

$C_2: B, C, D$

  - We use the final set of potentials as the parameterization of the Markov Network- Provides a closed-form solution for both inference and learning
  - Cannot be used with parameter regularization, non-table factors or CRFs

# BP and Maximum Entropy

- A more general derivation that allows us to reformulate maximum likelihood learning with belief propagation as a unified optimization problem with an approximate objective
- Opens door to use of better approximation algorithms
- Start with dual of maximum-likelihood problem

Maximum-Entropy:

**Find**  $Q(\chi)$

**Maximizing**  $H_Q(\chi)$

**Subject to**  $E_Q[f_i] = E_D[f_i], i=1, \dots, k$

# Tractable Max Entropy

- Assume a cluster graph  $U$  consisting of
  - a set of clusters  $\{C_i\}$  connected by sepsets  $S_{ij}$ .
- Rather than optimize maximum-entropy
  - over the space of distributions  $Q$  we optimize

$$H_Q(\chi) \approx \sum_{C_i \in U} H_{\beta_i}(C_i) - \sum_{C_i - C_j \in S} H_{\mu_{i,j}}(S_{i,j})$$

- This is exact for tree cluster graph but approx otherwise

Approx-Maximum-Entropy:

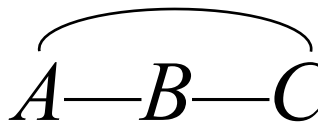
**Find**  $Q(\chi)$

**Maximizing**  $\sum_{C_i \in U} H_{\beta_i}(C_i) - \sum_{C_i - C_j \in U} H_{\mu_{i,j}}(S_{i,j})$

**Subject to**  $E_Q[f_i] = E_D[f_i], i=1, \dots, k \quad Q \in Local(U)$

- Approximation is exact when cluster graph is a tree  
Method known as CAMEL (*Constrained Approx Max Entropy Learning*)

# Example of Max Ent Learning

- Pairwise Markov Network  $A \text{---} B \text{---} C$ 

  - Variables are binary
  - Three clusters:  $C_1 = \{A, B\}$ ,  $C_2 = \{B, C\}$ ,  $C_3 = \{C, A\}$
  - Log-linear model with features
    - $f_{00}(x, y) = 1$  if  $x=0, y=0$  and 0 otherwise for  $x, y$ , instance of  $C_i$
    - $f_{11}(x, y) = 1$  if  $x=1, y=1$  and 0 otherwise
  - Three data instances  $(A, B, C)$ :  $(0, 0, 0), (0, 1, 0), (1, 0, 0)$ 
    - Unnormalized Feature counts, pooled over all clusters, are
 
$$E_{\hat{P}}[f_{00}] = (3 + 1 + 1) / 3 = 5 / 3$$

$$E_{\hat{P}}[f_{11}] = (0 + 0 + 0) / 3 = 0$$



# CAMEL Optimization Problem

- Optimization problem takes the form
  - with two types of constraints:

Find  $Q = \{\beta_1, \beta_2, \beta_3, \mu_{1,2}, \mu_{2,3}, \mu_{1,3}\}$   
 maximizing  $H_{\beta_1}(A, B) + H_{\beta_2}(B, C) + H_{\beta_3}(A, C)$   
 subject to  $-H_{\mu_{1,2}}(B) - H_{\mu_{2,3}}(C) - H_{\mu_{1,3}}(A)$

$$\sum_i E_{\beta_i}[f_{00}] = 5/3$$

$$\sum_i E_{\beta_i}[f_{11}] = 0$$

$$\sum_a \beta_1(a, b) - \sum_c \beta_2(b, c) = 0 \quad \forall b$$

$$\sum_b \beta_2(b, c) - \sum_a \beta_3(a, c) = 0 \quad \forall c$$

$$\sum_c \beta_3(a, c) - \sum_b \beta_1(a, b) = 0 \quad \forall a$$

$$\sum_{\mathbf{c}_i} \beta_i(\mathbf{c}_i) = 1 \quad i = 1, 2, 3$$

$$\beta_i \geq 0 \quad i = 1, 2, 3.$$

Type 1 Constraints:

$$E_Q[f_i] = E_D[f_i], i=1, \dots, k$$

Type 2 Constraints:  
 Marginals from  
 Cluster-graph  
 approximation

# CAMEL Solutions

- CAMEL optimization is a constrained maximization problem with
  - linear constraints and
  - a nonconcave objective
- Several solution algorithms, one of which is
  - Lagrange multipliers for all constraints and optimize over resulting new variables

# Sampling-based Learning

- We wish to maximize the log-likelihood

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

- In the sampling-based approach we use samples to estimate the partition function  $Z(\theta)$  and use that estimate to perform gradient ascent

# Expressing $Z(\theta)$ as an expectation

- Partition function  $Z(\theta)$  is a summation over an exponentially large space
  - One approach to approximating this summation is to reformulate as expectation wrt a distribution  $Q(\chi)$

$$\begin{aligned} Z(\theta) &= \sum_{\xi} \exp \left\{ \sum_i \theta_i f_i(\xi) \right\} \\ &= \sum_{\xi} \frac{Q(\xi)}{Q(\xi)} \exp \left\{ \sum_i \theta_i f_i(\xi) \right\} \\ &= E_Q \left[ \frac{1}{Q(\chi)} \exp \left\{ \sum_i \theta_i f_i(\chi) \right\} \right] \end{aligned}$$

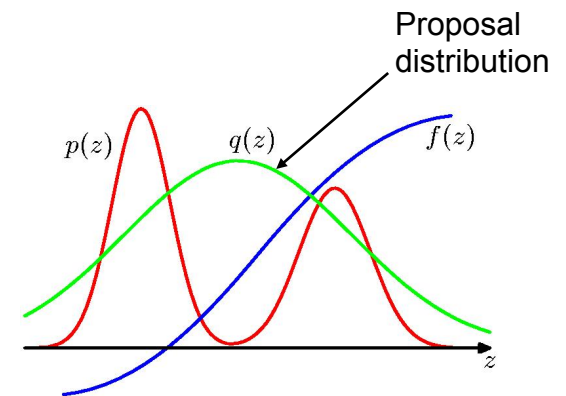
Since  $E_Q[g(\chi)] = \sum_{\xi} Q(\xi)g(\xi)$

- This is precisely the form of importance sampling estimator

# Importance sampling

- To determine  $E_p[f] = \frac{1}{M} \sum_{m=1}^M f(z^{[m]})$  from  $p(z)$ , we draw samples  $\{z^{(m)}\}$  from a simpler dist.  $q(z)$

$$\begin{aligned}
 E[f] &= \int f(z) p(z) dz \\
 &= \int f(z) \frac{p(z)}{q(z)} q(z) dz \\
 &= \frac{1}{M} \sum_{m=1}^M \frac{p(z^{(m)})}{q(z^{(m)})} f(z^{(m)})
 \end{aligned}$$



Unlike rejection sampling  
All of the samples are retained

- Samples are weighted by ratios  $r_l = p(z^{(l)}) / q(z^{(l)})$ 
  - Known as importance weights
    - Which corrects the bias introduced by wrong distribution

# Importance Sampling Estimator

- We can approximate the partition function by generating samples from  $Q$  and correcting appropriately via weights
- We can simplify this expression by choosing  $Q$  to be  $P_{\theta^0}$  for some set of parameters  $\theta^0$

$$\begin{aligned} Z(\theta) &= E_{P_{\theta^0}} \left[ \frac{Z(\theta^0) \exp \left\{ \sum_i \theta_i f_i(\chi) \right\}}{\exp \left\{ \sum_i \theta_i^0 f_i(\chi) \right\}} \right] \\ &= Z(\theta^0) E_{P_{\theta^0}} \left[ \exp \left\{ \sum_i (\theta_i - \theta_i^0) f_i(\chi) \right\} \right] \end{aligned}$$

# Samples to approximate $\ln Z(\theta)$

- If we can sample instances  $\xi^1, \dots, \xi^K$  from  $P_{\theta^0}$ 
  - We can approximate the log-partition function as

$$\ln Z(\theta) \approx \ln \left( \frac{1}{K} \sum_{k=1}^K \exp \left\{ \sum_i (\theta_i - \theta_i^0) f_i(\xi^k) \right\} \right) + \ln Z(\theta^0)$$

- We can plug this approximation of  $\ln Z(\theta)$  into the log-likelihood

$$\frac{1}{M} \ell(\theta : D) = \sum_i \theta_i (E_D[f_i(d_i)]) - \ln Z(\theta)$$

and optimize it

- Note that  $\ln Z(\theta^0)$  is a constant

- that we can ignore in the optimization and the resulting expression is a simple function of  $\theta$  which can be optimized using gradient ascent

where  $E_D[f_i(\mathbf{d}_i)]$  is the empirical expectation of  $f_i$ , i.e., its average in the data set

# Gradient ascent + Sampling

- Gradient ascent over  $\theta$  relative to

$$\ln Z(\theta) \approx \ln \left( \frac{1}{K} \sum_{k=1}^K \exp \left\{ \sum_i (\theta_i - \theta_i^0) f_i(\xi^k) \right\} \right) + \ln Z(\theta^0)$$

- is equivalent to utilizing an importance sampling estimator directly to approximate the expected counts in the gradient of equation

$$\frac{\partial}{\partial \theta_i} \frac{1}{M} \ell(\theta : D) = E_D[f_i(\chi)] - E_\theta[f_i]$$

- However, it is generally useful to view such methods as exactly optimizing an approximate objective rather than approximately optimizing the exact likelihood



## Quality of importance sampling estimator

- It depends on the difference between  $\theta$  and  $\theta^0$
- The greater the difference, the larger the variance of the importance weights
- Thus this type of approximation is reasonable only in a neighborhood of surrounding  $\theta^0$

# How to use this approximation?

- Iterate between two steps
  1. Use a sampling procedure to generate samples from current parameter set  $\theta^t$
  2. Then use gradient descent to find  $\theta^{t+1}$  that improves the approximate log-likelihood based on the samples
    - We can then regenerate samples and repeat the process.
- As the samples are regenerated from a new distribution,
  - we can hope that that they are generated from a distribution not too far from the one we are currently optimizing maintaining a reasonable approximation

# MAP-based Learning

- Another approach to inference in learning
  - Approximating expected feature counts with the counts in the single MAP assignment to current MN
- Approximate gradient at assignment  $\theta$  is
 

$$E_D[f_i(\chi)] - f_i(\xi^{MAP}(\theta))$$

  - where  $\xi^{MAP}(\theta) = \arg \max_{\xi} P(\xi | \theta)$  is the MAP assignment given the current set of parameters  $\theta$
- Approach also called as *Viterbi training*
- Equivalent to exact optimization of approximate objective

$$\frac{1}{M} \ell(\theta : D) - \ln P(\xi^{MAP}(\theta) | \theta)$$

# Ex: CRFs for Protein Structure

- Predict the 3\_D structure of proteins
  - Proteins are chains of residues, each containing one of 20 possible amino acids
    - Amino acids are linked together into a common backbone structure onto which amino-specific side-chains are attached
  - Predict the side-chain conformations given the backbone.
  - Full configuration consists of upto four angles each of which takes on a continuous value
    - Discretize angle into a small no (3) rotamers
- Address optimization as MAP for CRF