

Learning Parameters of Undirected Models

Sargur Srihari

srihari@cedar.buffalo.edu

Topics

- Difficulties due to Global Normalization
- Likelihood Function
- Maximum Likelihood Parameter Estimation
 - Simple and Conjugate Gradient Ascent
- Conditionally Trained Models
- Parameter Learning with Missing Data
 - Gradient Ascent vs. EM
- Alternative Formulation of Max Likelihood
 - Maximum Entropy subject to Constraints
- Parameter Priors and Regularization

Local vs Global Normalization

- BN: Local normalization within each CPD

$$P(X) = \prod_{i=1}^N P(X_i \mid pa(X_i))$$

- MN: Global normalization (partition function)

$$P_{\Phi}(X) = \frac{1}{Z} \prod_{i=1}^m \phi_i(D_i) \text{ and } Z = \sum_{X_1, \dots, X_n} \prod_{i=1}^m \phi_i(D_i)$$

- Global factor Z couples all parameters preventing decomposition
- Significant computational ramifications
 - M.L. parameter estimation has no closed-form soln.
 - Need iterative methods

Issues in Parameter Estimation

- Simple ML parameter estimation (even with complete data) cannot be solved in closed form
- Need iterative methods such as gradient ascent
- Good news:
 - Likelihood function is concave
 - Methods converge with global optimum
- Bad news:
 - Each step in iterative algorithm requires inference
 - Simple parameter estimation expensive/intractable
 - Bayesian estimation is practically infeasible

Discriminative Training

- Common use of MNs is in settings such as image segmentation where we have a particular inference task in mind
- Train the network discriminatively to get good performance for our particular inference task

Likelihood Function

- Basis for all discussion of learning
- How likelihood function can be optimized to find maximum likelihood parameter estimates
- Begin with form of likelihood function for Markov networks, its properties and their computational implications
- Existence of a global partition function couples the different parameters in a Markov network
 - greatly complicating the estimation problem

Example of very simple MN

- Markov network $A—B—C$

- Parameterized by potentials $\phi_1(A, B)$, $\phi_2(B, C)$:

- Recall Gibbs:

$$P(a, b, c) = (1/Z) \phi_1(a, b) \phi_2(b, c) \text{ where}$$

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

- Log-likelihood of instance (a, b, c) is

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

$\phi_1[A, B]$		
a^0	b^0	30
a^0	b^1	5
a^1	b^0	1
a^1	b^1	10

$\phi_2[B, C]$		
b^0	c^0	100
b^0	c^1	1
b^1	c^0	1
b^1	c^1	100

- Log-likelihood of data set \mathcal{D} with M instances:

$$\begin{aligned} \ell(\theta : D) &= \sum_{m=1}^M (\ln \phi_1(a[m], b[m]) + \ln \phi_2(b[m], c[m]) - \ln Z(\theta)) \\ &= \sum_{a, b} M[a, b] \ln \phi_1(a, b) + \sum_{b, c} M[b, c] \ln \phi_2(b, c) - M \ln Z(\theta) \end{aligned}$$

$$M[a, b] = \text{no. with value } (a, b)$$

Parameter θ consists of all values of factors ϕ_1 and ϕ_2

Summing over instances $\{a[m], b[m], c[m]\}$, $m=1, \dots, M$

Then summing over different values of $a \in \{a^0, a^1\}$ and $b \in \{b^0, b^1\}$

- This likelihood has three terms which we analyze

Coupled Likelihood of Simple MN

- Simple Markov network $A—B—C$
- Log-likelihood of data set \mathcal{D} with M instances:

$$\begin{aligned} \ell(\theta : D) &= \sum_{m=1}^M (\ln \phi_1(a[m], b[m]) + \ln \phi_2(b[m], c[m]) - \ln Z(\theta)) \\ &= \sum_{a,b} M[a,b] \ln \phi_1(a,b) + \sum_{b,c} M[b,c] \ln \phi_2(b,c) - M \ln Z(\theta) \end{aligned}$$

$M[a,b]$ = no. with value (a,b)

- This likelihood consists of three terms.

- First term involves only ϕ_1
- Second only ϕ_2 . But third involves

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

- Thus $\ln Z(\theta)$ is a function of both ϕ_1 and ϕ_2 . It couples the two potentials in the likelihood function
- When we change one potential ϕ_1 , $Z(\theta)$ changes, possibly changing the value of ϕ_2 that maximizes $-\ln Z(\theta)$

Log-Likelihood Surface for $A \text{---} B \text{---} C$

- Two factors $\phi_1(A, B)$, $\phi_2(B, C)$
 - With binary variables each factor has four values
 - Thus, total of 8 parameters

$\phi_1[A, B]$		
a^0	b^0	30
a^0	b^1	5
a^1	b^0	1
a^1	b^1	10

$\phi_2[B, C]$		
b^0	c^0	100
b^0	c^1	1
b^1	c^0	1
b^1	c^1	100

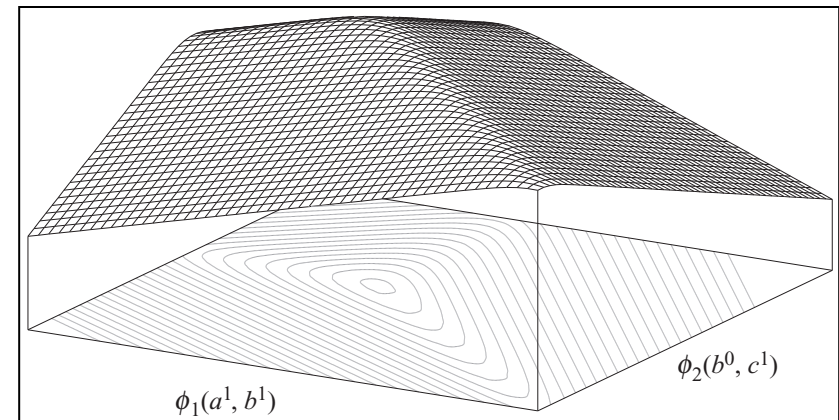
- We need parameters (values of ϕ_i) that maximize

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

- Ex: Log-likelihood surface

- X-axis: $\phi_1(a^1, b^1)$
- Y-axis: $\phi_2(b^0, c^1)$
- Other parameters set to 1
- Data set has $M=100$ with

$$M[a^1, b^1] = 40, \quad M[b^0, c^1] = 40$$



- Coupling problem:

- When ϕ_1 changes, ϕ_2 that maximizes $-\ln Z(\theta)$ also changes

Equivalent Bayesian Network

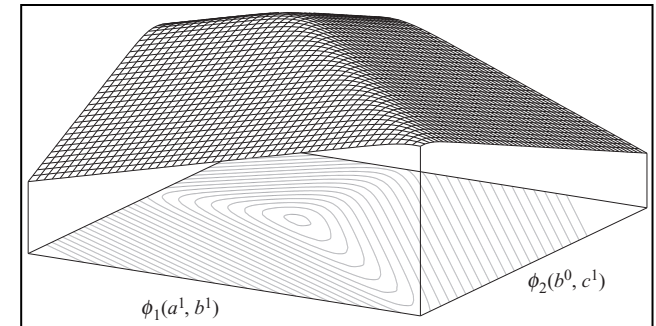
- Log-likelihood $A—B—C$ wrt two factors $\phi_1(A,B)$, $\phi_2(B,C)$
 - With binary variables we would have 8 parameters
- In this case, parameter coupling is avoidable
 - Since $A—B—C$ has an equivalent BN:

$$A \rightarrow B \rightarrow C$$

- We can estimate parameters using

$$\phi_1(A,B) = P(A)P(B|A)$$

$$\phi_2(B,C) = P(C|B)$$



- In general, cannot convert learned BN parameters into equivalent MN
 - Optimal likelihood achievable by the two representations is not the same

Log-linear form for MN

- Instead of Gibbs, use log-linear framework
 - Joint distribution of n variables X_1, \dots, X_n
 - k features $\mathcal{F} = \{ f_i(D_i) \}_{i=1, \dots, k}$ k depends on no. of values D_i takes
 - where D_i is a sub-graph and f_i maps D_i to \mathcal{R}

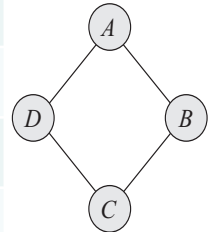
$$P(X_1, \dots, X_n; \theta) = \frac{1}{Z(\theta)} \exp \left\{ \sum_{i=1}^k \theta_i f_i(D_i) \right\}$$

- Parameters θ_i are weights we put on features
 - If we have a sample ξ then its features are $f_i(\xi(D_i))$ which has the shorthand $f_i(\xi)$.
- Representation is general
 - can capture Markov networks with global structure and local structure

Ex: Diamond network & binary features f

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

A	B	$\mathcal{F}(A, B)$
a^0	b^0	$f_{a^0, b^0} = I\{A=a^0\}I\{B=b^0\}$
a^0	b^1	f_{a^0, b^1}
a^1	b^0	f_{a^1, b^0}
a^1	b^1	f_{a^1, b^1}



- Variables: $A—B—C—D—$
- A feature for every entry in every table
 - $f_i(D_i)$ are sixteen indicator functions defined over clusters, AB, BC, CD, DA

$$f_{a^0 b^0}(A, B) = I\{A = a^0\}I\{B = b^0\} \longrightarrow \begin{array}{l} \text{Val}(A) = \{a^0, a^1\} \quad \text{Val}(B) = \{b^0, b^1\} \\ f_{a^0, b^0} = 1 \text{ if } a = a^0, b = b^0 \\ 0 \text{ otherwise, etc.} \end{array}$$

etc.

- With this representation

$$\theta_{a^0 b^0} = \ln \phi_1(a^0, b^0)$$

Parameters θ are potentials which are weights put on features

Log Likelihood using log-linear form of MN

- With log-linear form for probability distribution:

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

$\theta = \{ \theta_1, \dots, \theta_k \}$ are table entries
 f_i are features over instances of D_i

- Let \mathcal{D} be a data set of M samples $\xi[m]_{m=1, \dots, M}$
- Log-likelihood (product of indep probabilities)
 - log gets rid of exp, converts product to sum:

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

- Sufficient statistics (likelihood depends only on this)

Dividing
by no.
of samples

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

$E_{\mathcal{D}}[f_i(d_i)]$ is empirical
expectation: average
in the data set

Properties of Log-Likelihood

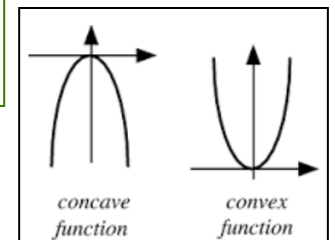
- Log-likelihood is a sum of two functions

$$\ell(\theta : D) = \underbrace{\sum_i \theta_i \left(\sum_m f_i(\xi[m]) \right)}_{\text{First Term}} - \underbrace{M \ln Z(\theta)}_{\text{Second Term}}$$

- First term is linear in the parameters θ
 - Increasing parameters increases this term
 - But likelihood has upper-bound of probability 1
- The Second term $\ln Z(\theta)$ balances first term

- Partition function is defined as
- It is convex as seen next
 - » Its negative is concave

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



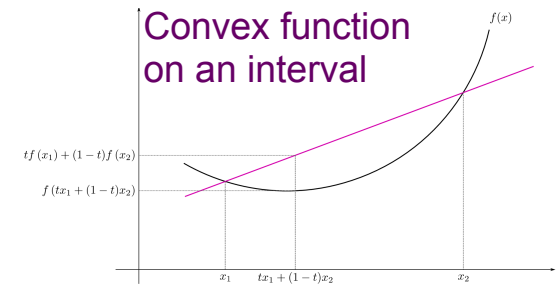
- Sum of linear fn. and concave is concave, So

- There are no local optima
 - Can use gradient ascent

Proof that $\ln Z(\theta)$ is Convex

A function $f(\vec{x})$ is convex if for every $0 \leq \alpha \leq 1$

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



- In other words, function is bowl-like
 - Every interpolation between the images of two points is larger than the image of their interpolation
- One way to show that a Function is convex
 - is to show that its Hessian (matrix of the function's second derivatives) is positive-semi-definite
 - Computing the derivatives of $\ln Z(\theta)$ is discussed next

Derivatives of $\ln Z(\theta)$

- Proposition: Let \mathcal{F} be a set of features. Then

$$\frac{\partial}{\partial \theta_i} \ln Z(\theta) = E_\theta[f_i]$$

$$\frac{\partial^2}{\partial \theta_i \partial \theta_j} \ln Z(\theta) = \text{Cov}_\theta[f_i; f_j]$$

where $E_\theta[f_i]$ is shorthand for $E_{P(\chi, \theta)}[f_i]$

Note: Expected value is defined as weighted average.

For a Bernoulli variable $x \in \{0, 1\}$ with $P(1)=p$, the distribution is $\text{Bern}(x|p) = p^x(1-p)^{1-x}$ and the expected value is p

- Proof: Since $\ln Z(\theta) = \ln \sum_{\xi} \exp \left\{ \sum_i \theta_i f_i(\xi) \right\}$

Partial derivatives
wrt θ_i and θ_j

$$\begin{aligned} \frac{\partial}{\partial \theta_i} \ln Z(\theta) &= \frac{1}{Z(\theta)} \sum_{\xi} \frac{\partial}{\partial \theta_i} \exp \left\{ \sum_j \theta_j f_j(\xi) \right\} = \frac{1}{Z(\theta)} \sum_{\xi} f_i(\xi) \exp \left\{ \sum_j \theta_j f_j(\xi) \right\} = E_\theta[f_i] \\ \frac{\partial^2}{\partial \theta_i \partial \theta_j} \ln Z(\theta) &= \frac{\partial}{\partial \theta_j} \left[\frac{1}{Z(\theta)} \sum_{\xi} \frac{\partial}{\partial \theta_i} \exp \left\{ \sum_k \theta_k f_k(\xi) \right\} \right] = \text{Cov}_\theta[f_i; f_j] \end{aligned}$$

- Since covariance matrix of features is positive semi-definite, we have $-\ln Z(\theta)$ is a concave function of θ

- Corollary: log-likelihood function is concave

Non-unique Solution

- Since $\ln Z(\theta)$ is convex, $-\ln Z(\theta)$ is concave
- Implies that log-likelihood is unimodal
 - Has no local optima
- However does not imply uniqueness of global optimum
- Multiple parameterizations can result in same distribution
 - A feature for every entry in the table is always redundant, e.g.,
$$f_{a0,b0} = 1 - f_{a0,b1} - f_{a1,b0} - f_{a1,b1}$$
 - A continuum of parameterizations

Maximum Likelihood Parameter Estimation

- Task: Estimate parameters of a MN with a fixed structure given a fully observable data set \mathcal{D}
- Simplest variant of the problem is maximum likelihood parameter estimation
- Log-likelihood with features $\mathcal{F} = \{ f_i, i = 1, \dots, k \}$ is

$$\ell(\theta : D) = \sum_{i=1} \theta_i \left(\sum_m f_i(\xi[m]) \right) - M \ln Z(\theta)$$

- Although log-likelihood function
 - is concave, no analytical form for its maximum
 - Can use iterative methods, e.g. gradient ascent

Gradient of Log-likelihood

- Log-likelihood: $\ell(\theta : D) = \sum_{i=1} \theta_i \left(\sum_m f_i(\xi[m]) \right) - M \ln Z(\theta)$
- Average log-likelihood: $\frac{1}{M} \ell(\theta : D) = \sum_i \theta_i (E_D[f_i(d_i)]) - \ln Z(\theta)$ Dividing by no. of samples M
- We have seen that gradient of second term $\ln Z(\theta)$ is

$$\frac{\partial}{\partial \theta_i} \ln Z(\theta) = \frac{1}{Z(\theta)} \sum_{\xi} \frac{\partial}{\partial \theta_i} \exp \left\{ \sum_j \theta_j f_j(\xi) \right\} = \frac{1}{Z(\theta)} \sum_{\xi} f_i(\xi) \exp \left\{ \sum_j \theta_j f_j(\xi) \right\} = E_{\theta}[f_i]$$
- We can compute gradient of average log-likelihood as

$$\frac{\partial}{\partial \theta_i} \frac{1}{M} \ell(\theta : D) = E_D[f_i(\chi)] - E_{\theta}[f_i]$$
First term is average value of f_i in data \mathcal{D} . Second term is expected value from distribution
- Provides a precise characterization of m.l. parameters θ
- *Theorem:* Let \mathcal{F} be a feature set. Then θ is a m.l. assignment if and only if $E_D[f_i(\chi)] = E_{\hat{\theta}}[f_i]$ for all i
 - i.e., expected value of each feature relative to P_{θ} matches its empirical expectation in \mathcal{D}

Need for Iterative Method

- Although log-likelihood function

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

- is concave, there is no analytical form for the maximum
- Since no closed-form solution
 - Can use iterative methods, e.g. gradient ascent as shown next
- Fortunately, exact form of gradient is known:

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

Computing the gradient

- Gradient wrt θ_i of log-likelihood $\ell(\theta; \mathcal{D})$ is

$$E_D[f_i(\chi)] - E_\theta[f_i]$$

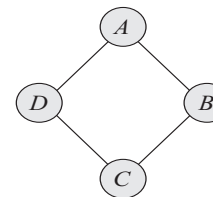
- It is the difference between the feature's

1. *empirical expectation* in data \mathcal{D} and
2. *Its expected value* relative to current parameterization

- Computing empirical expectation $E_D[f_i(\chi)]$ is easy

- Ex: for feature

$$f_{a^0 b^0}(a, b) = I\{a = a^0\} I\{b = b^0\}$$



All four binary-valued
 $a \in \{a^0, a^1\}$ etc.
 Features are indicator fns

it is the empirical frequency in \mathcal{D} of the event a^0, b^0

- At a particular parameterization θ , the expected f_i is

simply $P_\theta(a^0, b^0)$ since $E_\theta[f_i] = \frac{1}{Z(\theta)} \sum_{\xi} f_i(\xi) \exp\left\{\sum_j \theta_j f_j(\xi)\right\}$

- Note: for Bernoulli($x|p$), expectation is p

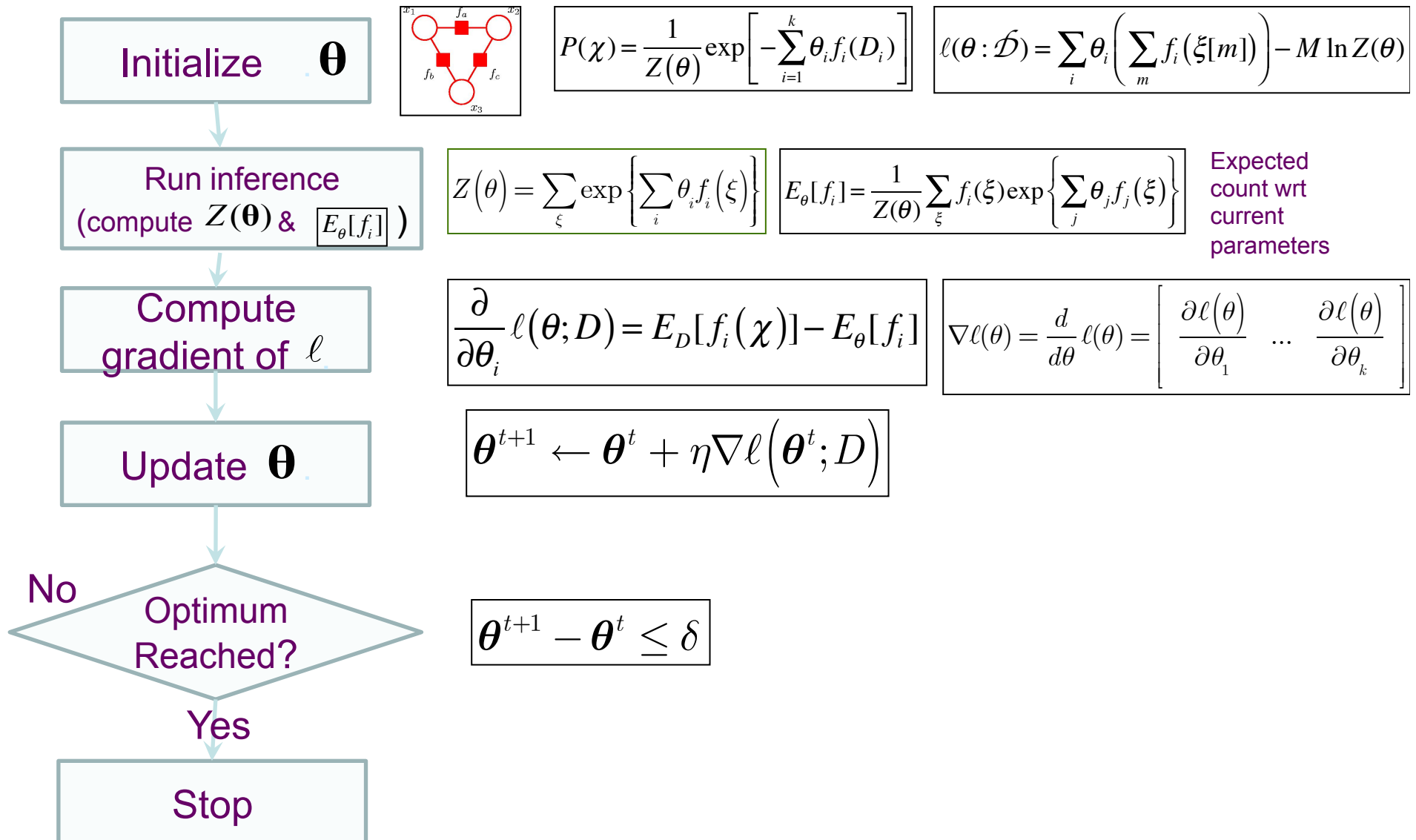
Computing the expected count

- Second term in the gradient is

$$E_{\theta}[f_i] = \frac{1}{Z(\theta)} \sum_{\xi} f_i(\xi) \exp \left\{ \sum_j \theta_j f_j(\xi) \right\}$$

- We need to compute the different probabilities of the form $P_{\theta^t}(a, b)$
 - Since expectation is a probability-weighted average
- Computing probability requires running inference over the network

Iterative solution for MRF parameters θ



Difficulty with Iterative method

- Gradient ascent over parameter space
- Good news:
 - likelihood function is concave
 - Guaranteed to converge to global optimum
- Bad news:
 - each step needs inference
 - Simple parameter estimation is intractable
 - Bayesian parameter estimation even harder
 - Integration done using MCMC

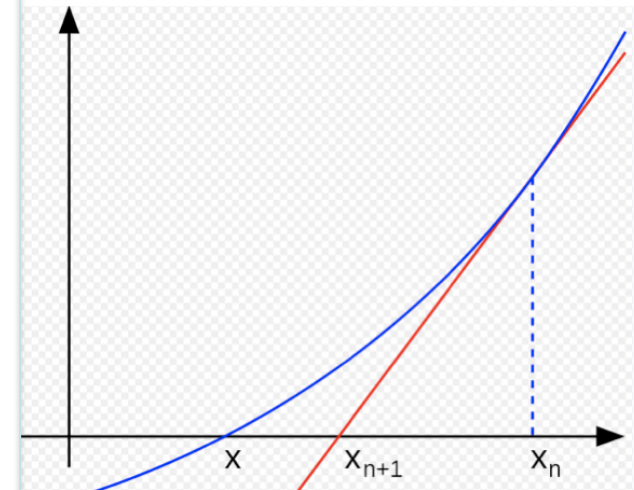
Use of Second Order Solution

- Computational cost of parameter estimation is very high
 - Gradient ascent is not efficient
- Much faster convergence using second order methods based on Hessian

Intuition for Second-Order Solution

- Newton's method
 - finds zeroes of a function using derivatives
- More efficient than simple gradient descent
- Quasi Newton's method
 - uses an approximation to the gradient
- Since we are solving for derivative of $l(\theta, \mathcal{D})$
 - need second derivative (Hessian)

Newton in one-dim.



$$f'(x_n) = \frac{\text{rise}}{\text{run}} = \frac{\Delta y}{\Delta x} = \frac{f(x_n) - 0}{x_n - x_{n+1}}.$$

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$

Derivation of Hessian of the Log-Likelihood

$$\begin{aligned}
 \frac{\partial}{\partial \theta_i} \ln Z(\theta) &= \frac{1}{Z(\theta)} \sum_{\xi} \frac{\partial}{\partial \theta_i} \exp \left\{ \sum_j \theta_j f_j(\xi) \right\} = \frac{1}{Z(\theta)} \sum_{\xi} f_i(\xi) \exp \left\{ \sum_j \theta_j f_j(\xi) \right\} = E_{\theta}[f_i] \\
 \frac{\partial^2}{\partial \theta_i \partial \theta_j} \ln Z(\theta) &= \frac{\partial}{\partial \theta_j} \left[\frac{1}{Z(\theta)} \sum_{\xi} \frac{\partial}{\partial \theta_i} \exp \left\{ \sum_k \theta_k f_k(\xi) \right\} \right] \\
 &= -\frac{1}{Z(\theta)^2} \left(\frac{\partial}{\partial \theta_j} Z(\theta) \right) \sum_{\xi} f_i(\xi) \exp \left\{ \sum_k \theta_k f_k(\xi) \right\} + \frac{1}{Z(\theta)} \sum_{\xi} f_i(\xi) f_j(\xi) \exp \left\{ \sum_k \theta_k f_k(\xi) \right\} \\
 &= -\frac{1}{Z(\theta)^2} Z(\theta) E_{\theta}[f_j] \sum_{\xi} f_i(\xi) \tilde{P}(\xi : \theta) + \frac{1}{Z(\theta)} \sum_{\xi} f_i(\xi) f_j(\xi) \tilde{P}(\xi : \theta) \\
 &= -E_{\theta}[f_j] \sum_{\xi} f_i(\xi) P(\xi : \theta) + \sum_{\xi} f_i(\xi) f_j(\xi) P(\xi : \theta) \\
 &= E_{\theta}[f_i f_j] - E_{\theta}[f_i] E_{\theta}[f_j] \\
 &= \text{Cov}_{\theta}[f_i; f_j]
 \end{aligned}$$

$$\frac{\partial}{\partial \theta_i \partial \theta_j} \ell(\theta, D) = -M \text{Cov}_{\theta}(f_i, f_j)$$

Computation of Hessian

- Log-likelihood has the form

$$\ell(\theta : D) = \sum_{i=1} \theta_i \left(\sum_m f_i(\xi[m]) \right) - M \ln Z(\theta)$$

- Solution 1: Hessian

$$\frac{\partial}{\partial \theta_i \partial \theta_j} \ell(\theta, D) = -M \text{Cov}_{\theta}(f_i, f_j)$$

- Requires joint expectation of two features, often computationally infeasible
- Solution 2: Commonly used
 - *L-BFGS* (a quasi-Newton algorithm)
 - uses gradient ascent line search to avoid computing the Hessian

L-BFGS Algorithm

- Limited-memory BFGS
- Approximates the Broyden-Fletcher-Goldfarb-Shanno (BFGS)
- Popular algorithm for ML parameter estimation
 - Algorithm of choice for log-linear models and CRFs
- Uses an estimation of the inverse Hessian to steer through variable space

Choosing η : Line Search

- We can choose η in several different ways
- Popular approach: set η to a small constant
- Another approach is called *line search*:
- Evaluate $f(\mathbf{x} - \eta \nabla_{\mathbf{x}} f(\mathbf{x}))$ for several values of η and choose the one that results in smallest objective function value

Line Search (with ascent)

- In Gradient Ascent, we increase f by moving in the direction of the gradient

$$\theta^{t+1} \leftarrow \theta^t + \eta \nabla f(\theta^t)$$

- In Line Search: step size η adaptively chosen
 - Choose direction to ascend and continue in direction until we start to descend
 - Define “line” in direction of gradient

$$g(\eta) = \vec{\theta}^t + \eta \nabla f(\theta^t)$$

- Note that this is a linear function of η and hence it is a “line”

Line Search: determining η

- Given $g(\eta) = \vec{\theta}^t + \eta \nabla f(\theta^t)$
- Find three points $\eta_1 < \eta_2 < \eta_3$ so that $f(g(\eta_2))$ is larger than at both $f(g(\eta_1))$ and $f(g(\eta_3))$
 - We say that $\eta_1 < \eta_2 < \eta_3$ **bracket** a maximum
- If we find an η' so that we can find a new tighter bracket $\eta_1 < \eta' < \eta_2$
 - To find η' use binary search

Choose $\eta' = (\eta_1 + \eta_3)/2$
- Method ensures that new bracket is half of the old one
 - Note: Other methods are:
 - *Brent's method* uses a quadratic function instead of linear
 - *Conjugate gradient descent* converges faster than line search

