

CS 6347

Lecture 12

Maximum Likelihood Learning

- Given samples x^1, \dots, x^M from some unknown distribution with parameters θ ...
- The **log-likelihood** of the evidence is defined to be

$$\log l(\theta) = \sum_m \log p(x|\theta)$$

- Goal: maximize the log-likelihood

MLE for Bayesian Networks



- Given samples x^1, \dots, x^M from some unknown Bayesian network that factors over the directed acyclic graph G
 - The parameters of a Bayesian model are simply the conditional probabilities that define the factorization
 - For each $i \in G$ we need to learn $p(x_i | x_{parents(i)})$, create a variable $\theta_{x_i | x_{parents(i)}}$

$$\log l(\theta) = \sum_m \sum_{i \in V} \log \theta_{x_i^m | x_{parents(i)}^m}$$

MLE for Bayesian Networks



$$\begin{aligned}\log l(\theta) &= \sum_m \sum_{i \in V} \log \theta_{x_i^m | x_{\text{parents}(i)}^m} \\ &= \sum_{i \in V} \sum_m \log \theta_{x_i^m | x_{\text{parents}(i)}^m} \\ &= \sum_{i \in V} \sum_{x_{\text{parents}(i)}} \sum_{x_i} N_{x_i, x_{\text{parents}(i)}} \log \theta_{x_i | x_{\text{parents}(i)}}\end{aligned}$$

MLE for Bayesian Networks



$$\begin{aligned}\log l(\theta) &= \sum_m \sum_{i \in V} \log \theta_{x_i^m | x_{\text{parents}(i)}^m} \\ &= \sum_{i \in V} \sum_m \log \theta_{x_i^m | x_{\text{parents}(i)}^m} \\ &= \sum_{i \in V} \sum_{x_{\text{parents}(i)}} \sum_{x_i} N_{x_i, x_{\text{parents}(i)}} \log \theta_{x_i | x_{\text{parents}(i)}}\end{aligned}$$

$N_{x_i, x_{\text{parents}(i)}}$ is the number of times
 $(x_i, x_{\text{parents}(i)})$ was observed in the samples

MLE for Bayesian Networks



$$\begin{aligned}\log l(\theta) &= \sum_m \sum_{i \in V} \log \theta_{x_i^m | x_{\text{parents}(i)}^m} \\ &= \sum_{i \in V} \sum_m \log \theta_{x_i^m | x_{\text{parents}(i)}^m} \\ &= \sum_{i \in V} \sum_{x_{\text{parents}(i)}} \sum_{x_i} N_{x_i, x_{\text{parents}(i)}} \log \theta_{x_i | x_{\text{parents}(i)}}\end{aligned}$$

Fix $x_{\text{parents}(i)}$ solve for $\theta_{x_i | x_{\text{parents}(i)}}$ for all x_i
(on the board)

MLE for Bayesian Networks



$$\theta_{x_i|x_{parents(i)}} = \frac{N_{x_i, x_{parents(i)}}}{\sum_{x'_i} N_{x'_i, x_{parents(i)}}} = \frac{N_{x_i, x_{parents(i)}}}{N_{x_{parents(i)}}}$$

- This is just the empirical conditional probability distribution
 - Worked out nicely because of the factorization of the joint distribution
- Similar to the coin flips result from last time

- Let's compute the MLE for MRFs that factor over the graph G as $p(x|\theta) = \frac{1}{Z(\theta)} \prod_C \psi_C(x_C|\theta)$
- The parameters θ control the allowable potential functions
- Again, suppose we have samples x^1, \dots, x^M from some unknown MRF of this form

$$\log l(\theta) = \left[\sum_m \sum_C \log \psi_C(x_C^m|\theta) \right] - M \log Z(\theta)$$

- Let's compute the MLE for MRFs that factor over the graph G as $p(x|\theta) = \frac{1}{Z(\theta)} \prod_C \psi_C(x_C|\theta)$
- The parameters θ control the allowable potential functions
- Again, suppose we have samples x^1, \dots, x^M from some unknown MRF of this form

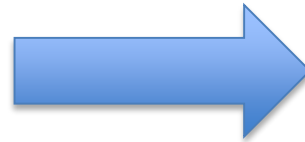
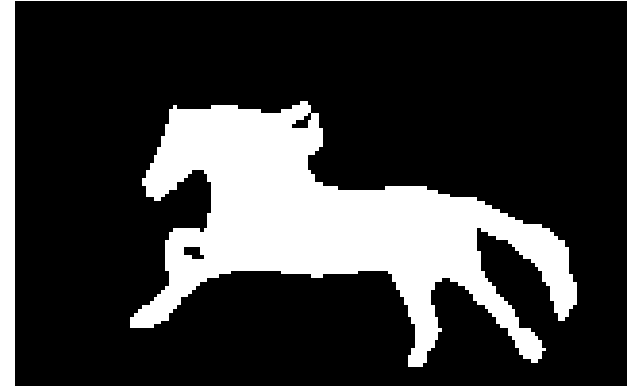
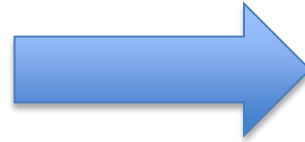
$$\log l(\theta) = \left[\sum_m \sum_C \log \psi_C(x_C^m|\theta) \right] - M \log Z(\theta)$$

$Z(\theta)$ couples all of the potential functions together!

Even computing $Z(\theta)$ by itself was a challenging task...

- Learning MRFs is quite restrictive
 - Most “real” problems are really conditional models
- Example: image segmentation
 - Represent a segmentation problem as a MRF over a two dimensional grid
 - Each x_i is an binary variable indicating whether or not the pixel is in the foreground or the background
 - How do we incorporate pixel information?
 - The potentials over the edge (i, j) of the MRF should depend on x_i, x_j as well as the pixel information at nodes i and j

Image Segmentation



- The pixel information is called a **feature** of the model
 - Features will consist of more than just a scalar value (i.e., pixels, at the very least, are vectors of RGBA values)
- Vector of features y (e.g., one vector of features y_i for each $i \in V$)
 - We think of the joint probability distribution as a conditional distribution $p(x|y, \theta)$
- This makes MLE even harder
 - Samples are pairs $(x^1, y^1), \dots, (x^M, y^M)$
 - The feature vectors can be different for each sample: need to compute $Z(\theta, y^m)$ in the log-likelihood!

- MLE seems daunting for MRFs and CRFs
 - Need a nice way to parameterize the model and to deal with features
- We often assume that the models are **log-linear** in the parameters
 - Many of the models that we have seen so far can easily be expressed as log-linear models of the parameters
 - Feature vectors should also be incorporated in a log-linear way

- The potential on the clique C should be a log-linear function of the parameters

$$\psi_C(x_C|y, \theta) = \exp(\langle \theta, f_C(x_C, y) \rangle)$$

where

$$\langle \theta, f_C(x_C, y) \rangle = \sum_k \theta_k \cdot f_C(x_C, y)_k$$

- Here, f is a **feature map** that takes the input variables and returns a vector the same size as θ

- Over complete representation: one parameter for each clique C and choice of x_C

$$p(x|\theta) = \frac{1}{Z} \prod_C \exp(\theta_C(x_C))$$

- $f_C(x_C)$ is a 0-1 vector that is indexed by C and x_C whose only non-zero component corresponds to $\theta_C(x_C)$
- One parameter per clique

$$p(x|\theta) = \frac{1}{Z} \prod_C \exp(\theta_C f_C(x_C))$$

- $f_C(x_C)$ is a vector that is indexed ONLY by C whose only non-zero component corresponds to θ_C

MLE for Log-Linear Models



$$p(x|y, \theta) = \frac{1}{Z(\theta, y)} \prod_c \exp(\langle \theta, f_c(x_c, y) \rangle)$$

$$\begin{aligned} \log l(\theta) &= \sum_m \left[\left[\sum_c \langle \theta, f_c(x_c^m, y^m) \rangle \right] - \log Z(\theta, y^m) \right] \\ &= \left\langle \theta, \sum_m \sum_c f_c(x_c^m, y^m) \right\rangle - \sum_m \log Z(\theta, y^m) \end{aligned}$$

MLE for Log-Linear Models



$$p(x|y, \theta) = \frac{1}{Z(\theta, y)} \prod_c \exp(\langle \theta, f_c(x_c, y) \rangle)$$

$$\log l(\theta) = \sum_m \left[\left[\sum_c \langle \theta, f_c(x_c^m, y^m) \rangle \right] - \log Z(\theta, y^m) \right]$$

$$= \underbrace{\left\langle \theta, \sum_m \sum_c f_c(x_c^m, y^m) \right\rangle}_{\text{Linear in } \theta} - \underbrace{\sum_m \log Z(\theta, y^m)}_{\text{Depends non-linearly on } \theta}$$

Linear in θ

Depends non-linearly
on θ

Concavity of MLE



We will show that $\log Z(\theta, y)$ is a convex function of θ ...

Fix a distribution $q(x|y)$

$$\begin{aligned} D(q||p) &= \sum_x q(x|y) \log \frac{q(x|y)}{p(x|y, \theta)} \\ &= \sum_x q(x|y) \log q(x|y) - \sum_x q(x|y) \log p(x|y, \theta) \\ &= -H(q) - \sum_x q(x|y) \log p(x|y, \theta) \\ &= -H(q) + \log Z(\theta, y) - \sum_x \sum_C q(x|y) \langle \theta, f_C(x_C, y) \rangle \\ &= -H(q) + \log Z(\theta, y) - \sum_C \sum_{x_C} q_C(x_C|y) \langle \theta, f_C(x_C, y) \rangle \end{aligned}$$

$$\log Z(\theta, y) = \max_q \left[H(q) + \sum_C \sum_{x_C} q_C(x_C | y) \underbrace{\langle \theta, f_C(x_C, y) \rangle}_{\text{Linear in } \theta} \right]$$

- If a function $g(x, y)$ is convex in x for each y , then $\max_y g(x, y)$ is convex in x
 - As a result, $\log Z(\theta, y)$ is a convex function of θ for a fixed value of y

MLE for Log-Linear Models



$$p(x|y, \theta) = \frac{1}{Z(\theta, y)} \prod_c \exp(\langle \theta, f_c(x_c, y) \rangle)$$

$$\log l(\theta) = \sum_m \left[\left[\sum_c \langle \theta, f_c(x_c^m, y^m) \rangle \right] - \log Z(\theta, y^m) \right]$$

$$= \underbrace{\left\langle \theta, \sum_m \sum_c f_c(x_c^m, y^m) \right\rangle}_{\text{Linear in } \theta} - \underbrace{\sum_m \log Z(\theta, y^m)}_{\text{Convex in } \theta}$$

Linear in θ

Convex in θ

$$p(x|y, \theta) = \frac{1}{Z(\theta, y)} \prod_c \exp(\langle \theta, f_c(x_c, y) \rangle)$$

$$\log l(\theta) = \sum_m \left[\left[\sum_c \langle \theta, f_c(x_c^m, y^m) \rangle \right] - \log Z(\theta, y^m) \right]$$

$$= \underbrace{\left\langle \theta, \sum_m \sum_c f_c(x_c^m, y^m) \right\rangle - \sum_m \log Z(\theta, y^m)}_{\text{Concave in } \theta}$$

Concave in θ

Could optimize it using gradient ascent!
(need to compute $\nabla_{\theta} \log Z(\theta, y)$)

- What is the gradient of the log-likelihood with respect to θ ?

$$\nabla_{\theta} \log Z(\theta, y^m) = ?$$

(worked out on board)

- What is the gradient of the log-likelihood with respect to θ ?

$$\nabla_{\theta} \log Z(\theta, y^m) = \sum_C \sum_m \sum_{x_C} p_C(x_C | y^m, \theta) f_C(x_C, y^m)$$

This is the expected value of the feature maps under the joint distribution

- What is the gradient of the log-likelihood with respect to θ ?

$$\nabla_{\theta} \log l(\theta) = \sum_C \sum_m \left(f_C(x_C^m, y^m) - \sum_{x_C} p_C(x_C | y^m, \theta) f_C(x_C, y^m) \right)$$

- To compute/approximate this quantity, we only need to compute/approximate the marginal distributions $p_C(x_C | y, \theta)$
- This requires performing marginal inference on a different model at each step of gradient ascent!

- Let $f(x^m, y^m) = \sum_C f_C(x_C^m, y^m)$
- Setting the gradient with respect to θ equal to zero and solving gives

$$\sum_m f(x^m, y^m) = \sum_m \sum_x p(x|y^m, \theta) f(x, y^m)$$

- This condition is called **moment matching** and when the model is an MRF instead of a CRF this reduces to

$$\frac{1}{M} \sum_m f(x^m) = \sum_x p(x|\theta) f(x)$$

- As an example, consider a log-linear MRF

$$p(x) = \frac{1}{Z} \prod_C \exp(\theta_C(x_C))$$

- That is, $f_C(x_C)$ is a vector that is indexed by C and x_C whose only non-zero component corresponds to $\theta_C(x_C)$
- The moment matching condition becomes

$$\frac{1}{M} \sum_m 1_{x_C=x_C^m} = p_C(x_C|\theta), \quad \text{for all } C, x_C$$

- Recall that we can also incorporate prior information about the parameters into the MLE problem
 - This involved solving an augmented MLE

$$\prod_m p(x^m | \theta) p(\theta)$$

- What types of priors should we choose for the parameters?

- Recall that we can also incorporate prior information about the parameters into the MLE problem
 - This involved solving an augmented MLE

$$\left[\prod_m p(x^m | \theta) \right] p(\theta)$$

- What types of priors should we choose for the parameters?
 - Gaussian prior: $p(\theta) \propto \exp\left(-\frac{1}{2}(\theta - \mu)^T \Sigma^{-1}(\theta - \mu)\right)$
 - Uniform over $[0,1]$

- Recall that we can also incorporate prior information about the parameters into the MLE problem
 - This involved solving an augmented MLE

$$\left[\prod_m p(x^m | \theta) \right] \exp\left(-\frac{1}{2} \theta^T D \theta\right)$$

Gaussian prior with a diagonal covariance matrix all of whose entries are equal to λ

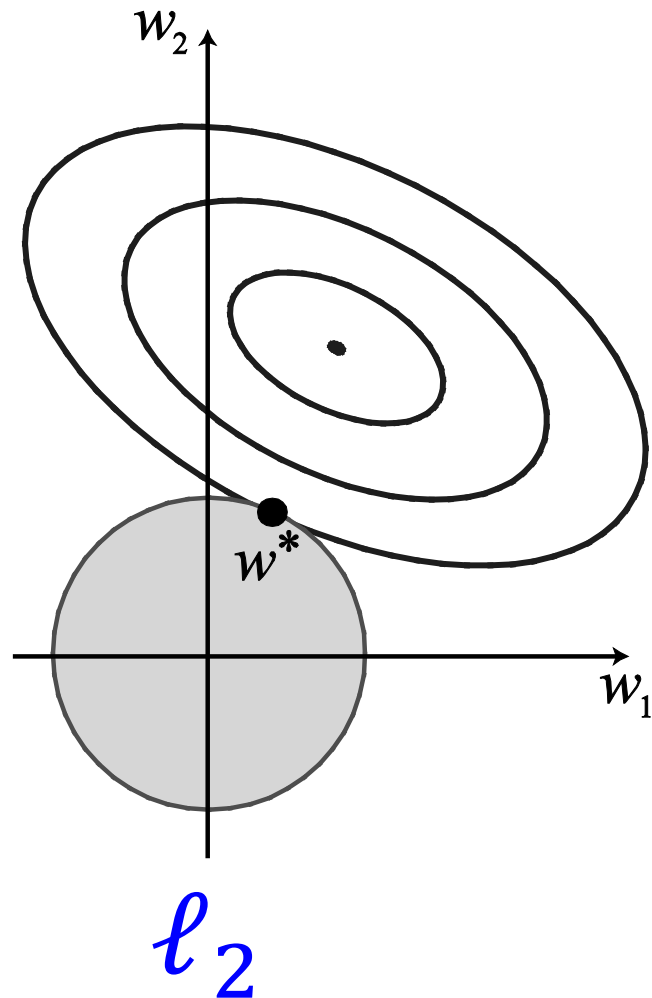
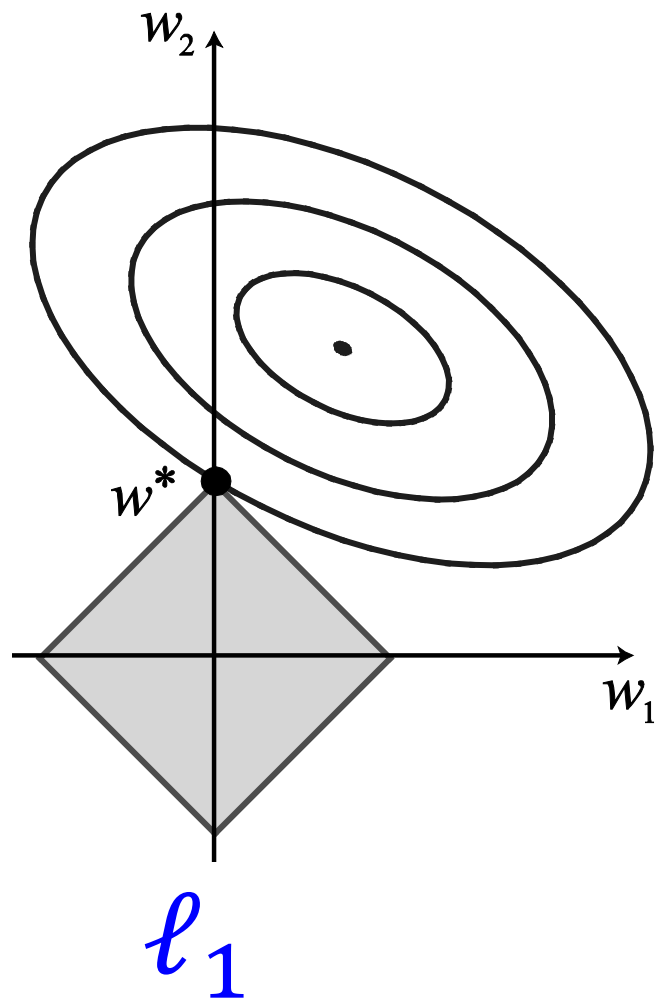
- What types of priors should we choose for the parameters?
 - Gaussian prior: $p(\theta) \propto \exp\left(-\frac{1}{2} (\theta - \mu)^T \Sigma^{-1} (\theta - \mu)^T\right)$
 - Uniform over $[0,1]$

- Using the previous Gaussian prior yields the following log-optimization problem

$$\begin{aligned}\log\left(\left[\prod_m p(x^m|\theta)\right] \exp\left(-\frac{1}{2}\theta^T D\theta\right)\right) &= \left[\sum_m \log p(x^m|\theta)\right] - \frac{\lambda}{2} \sum_k \theta_k^2 \\ &= \left[\sum_m \log p(x^m|\theta)\right] - \frac{\lambda}{2} \|\theta\|_2^2\end{aligned}$$

Known as ℓ_2 regularization

Regularization



$$\log Z(\theta, y) = \max_q \left[H(q) + \sum_C \sum_{x_C} q_C(x_C|y) \langle \theta, f_C(x_C, y) \rangle \right]$$

$$\log l(\theta) = \left\langle \theta, \sum_m \sum_C f_C(x_C^m, y^m) \right\rangle - \sum_m \log Z(\theta, y^m)$$

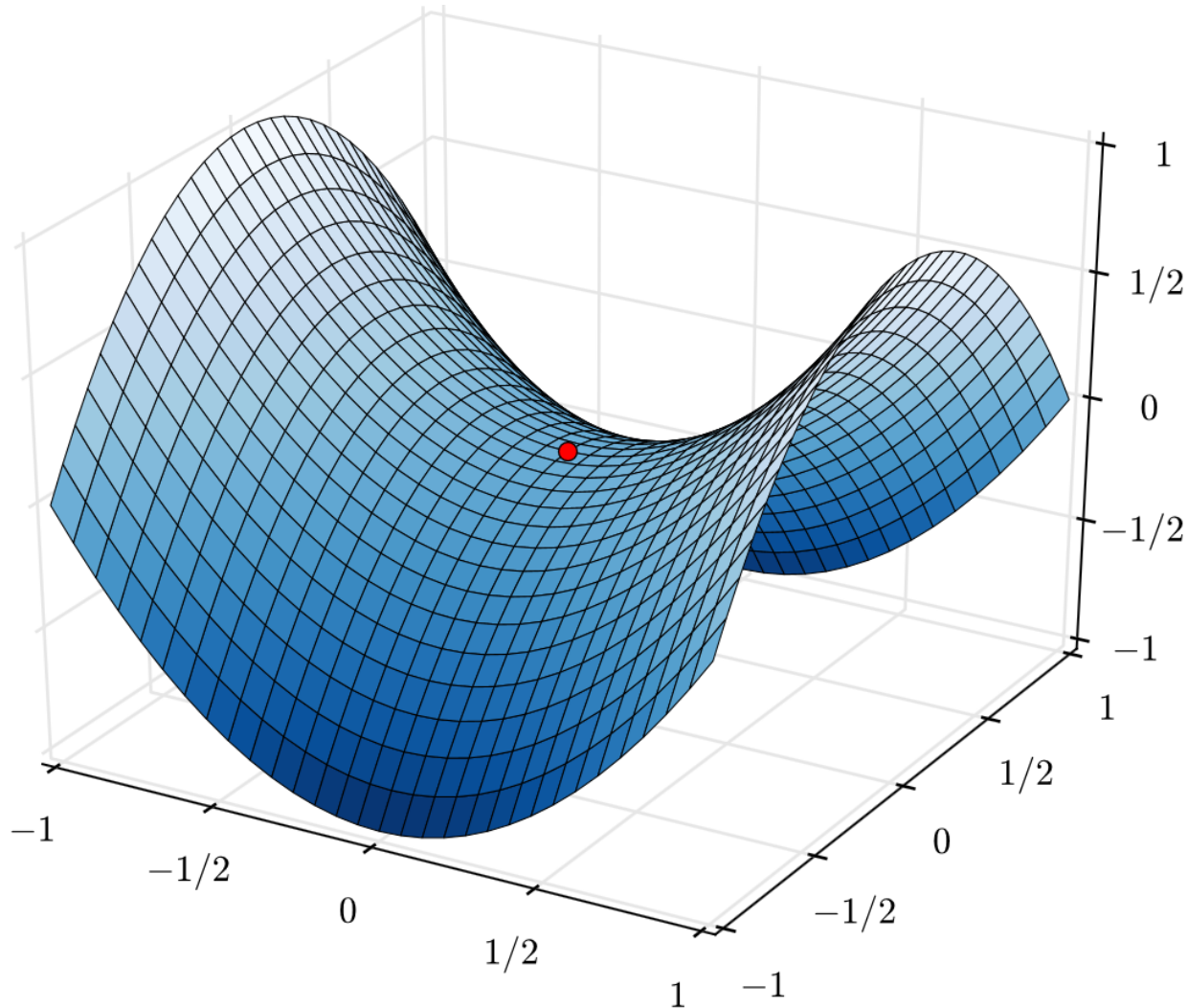
Plugging the first into the second gives:

$$\log l(\theta) = \left\langle \theta, \sum_m \sum_C f_C(x_C^m, y^m) \right\rangle - \sum_m \max_{q^m} \left[H(q^m) + \sum_C \sum_{x_C} q_C^m(x_C|y^m) \langle \theta, f_C(x_C, y^m) \rangle \right]$$

$$\max_{\theta} \log l(\theta) = \max_{\theta} \min_{q^1, \dots, q^M} \left[\left\langle \theta, \sum_c \sum_m \left(f_c(x_c^m, y^m) - \sum_{x_c} q_c^m(x_c | y^m) f_c(x_c, y^m) \right) \right\rangle - \sum_m H(q^m) \right]$$

- This is called a minimax or saddle-point problem
- When can we switch the order of the max and min?
 - The function is linear in theta, so there is an advantage to swapping the order

Saddle Point



Sion's Minimax Theorem



Let X be a compact convex subset of R^n and Y be a convex subset of R^m

Let f be a real-valued function on $X \times Y$ such that

- $f(x, \cdot)$ a continuous concave function over Y for each $x \in X$
- $f(\cdot, y)$ a continuous convex function over X for each $y \in Y$

then

$$\sup_y \min_x f(x, y) = \min_x \sup_y f(x, y)$$

$$\max_{\theta} \min_{q^1, \dots, q^M} \left[\left\langle \theta, \sum_C \sum_m \left(f_C(x_C^m, y^m) - \sum_{x_C} q_C^m(x_C | y^m) f_C(x_C, y^m) \right) \right\rangle - \sum_m H(q^m) \right]$$

is equal to

$$\min_{q^1, \dots, q^M} \max_{\theta} \left[\left\langle \theta, \sum_C \sum_m \left(f_C(x_C^m, y^m) - \sum_{x_C} q_C^m(x_C | y^m) f_C(x_C, y^m) \right) \right\rangle - \sum_m H(q^m) \right]$$

Solve for θ ?

$$\max_{q^1, \dots, q^M} \sum_m H(q^m)$$

such that the moment matching condition is satisfied

$$\sum_m f(x^m, y^m) = \sum_m \sum_x q^m(x|y^m) f(x, y^m)$$

and q^1, \dots, q^m are discrete probability distributions

- Instead of maximizing the log-likelihood, we could maximize the entropy over all approximating distributions that satisfy the moment matching condition

- We can compute the partition function in linear time over trees using belief propagation
 - We can use this to learn the parameters of tree-structured models
- What if the graph isn't a tree?
 - Use variable elimination to compute the partition function (exact but slow)
 - Use importance sampling to approximate the partition function (can also be quite slow; maybe only use a few samples?)
 - Use loopy belief propagation to approximate the partition function (can be bad if loopy BP doesn't converge quickly)

- Practical wisdom:
 - If you are trying to perform some prediction task (i.e., MAP inference to do prediction), then it is better to learn the “wrong model”
 - Learning and prediction should use the same approximations
- What people actually do:
 - Use a few iterations of loopy BP or sampling to approximate the marginals
 - Approximate marginals give approximate gradients (recall that the gradient only depended on the marginals)
 - Perform approximate gradient descent and hope it works

- Other options
 - Replace the true entropy with the Bethe entropy and solve the approximate dual problem
 - Use fancier optimization techniques to solve the problem faster
 - e.g., the method of conditional gradients

Course Project



- Pick a group (1-4) students
- Write a brief proposal and email it to me and Yibo
- Do the project
 - Collect/find a dataset
 - Build a graphical model
 - Solve approximately/exactly some inference or learning task
- Demo the project for the class (~15 mins during last 2-3 weeks)
 - Show your results
- Turn in a short write-up describing your project and results (due May 3)

Course Project



- Meet with me and Shahab (more if needed)
 - We'll help you get started and make sure you picked a hard/easy enough goal
- For one person:
 - Pick a small data set (or generate synthetic data)
 - Formulate a learning/inference problem using MRFs, CRFs, Bayesian networks
 - Example: SPAM filtering with a Bayesian network using the UCI spambase data set (or other data sets)
 - Compare performance across data sets and versus naïve algorithms

- For four people:
 - Pick a more complex data set
 - The graphical model that you learn should be more complicated than a simple Bayesian network
 - Ideally, the project will involve both learning and prediction using a CRF or an MRF (or a Bayesian network with hidden variables)
 - Example: simple binary image segmentation on smallish images
 - Be ambitious but cautious, you don't want to spend a lot of time formatting the data or worrying about feature selection

- Lots of other projects are possible
 - Read about, implement, and compare different approximate MAP inference algorithms (loopy BP, tree-reweighted belief propagation, max-sum diffusion)
 - Compare different approximate MLE schemes on synthetic data
 - Perform a collection of experiments to determine when the MAP LP is tight across a variety of pairwise, non-binary MRFs
 - If you are stuck, have a vague idea, ask me about it!

- What you need to do now
 - Find some friends (you can post on Piazza if you need friends)
 - Pick a project
 - Email me and Shahab (with all of your group members cc'd) by 3/23
- Grade will be determined based on the demo, final report, and project difficulty