# CS 6347

# Lecture 15

## Concave Entropy Approximations & Conditional Gradients

# Course Project

- Pick a group (1-4) students
- Write a brief proposal and email it to me and Travis
- 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 weeks)
  - Show your results
- Turn in a short write-up describing your project and results (due May 2)

# Course Project

- Meet with me and/or Travis about two times (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

# Course Project

- 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 or smallish images
  - Be ambitious but cautious, you don't want to spend a lot of time formatting the data or worrying about feature selection

# Course Project

- 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 (e.g., minimum s-t cuts)
  - 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!

# Course Project

- What you need to do now
  - Find some friends
  - Pick a project
  - Email me and Travis (with all of your group members cc'd) by 3/18

- Grade will be determined based on the demo, final report, and project difficulty

# Maximum Entropy

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

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 $q^1, \ldots, q^m$ 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!

# Regularized MLE

- $L_2$ regularizer with a constant $\lambda$
  - $\lambda$ is unknown and is chosen by cross-validation

Regularized log-likelihood:

$$\left\langle \theta, \sum_m \sum_C f_C(x_C^m, y^m) \right\rangle - \sum_m \log Z(\theta, y^m) - \frac{\lambda}{2} \|\theta\|_2^2$$

Regularized maximum entropy:

$$\max_{q^1,\ldots,q^m} \sum_m H(q^m) - \frac{1}{2\lambda} \left\| \sum_m f(x^m, y^m) - \sum_m \sum_x q^m(x|y^m) f(x, y^m) \right\|_2^2$$

# Bethe Entropy

$$H_B(\tau) = -\sum_{i \in V} \sum_{x_i} \tau_i(x_i) \log \tau_i(x_i) - \sum_C \sum_{x_C} \tau_C(x_C) \log \frac{\tau_C(x_C)}{\prod_{k \in C} \tau_k(x_k)}$$

- $\tau$ are pseudomarginals in the marginal polytope
- Not concave in general
  - Real entropy is concave
  - Can make it concave by “reweighting” some of the pieces

# Concave Entropy Approximations

$$H_\rho(\tau) = -\sum_{i\in V}\sum_{x_i}\tau_i(x_i)\log\tau_i(x_i) - \sum_C \rho_C \sum_{x_C}\tau_C(x_C)\log\frac{\tau_C(x_C)}{\prod_{k\in C}\tau_k(x_k)}$$

$$= -\sum_{i\in V}\sum_{x_i}\left(1-\sum_{C\supset i}\rho_C\right)\tau_i(x_i)\log\tau_i(x_i) - \sum_C\sum_{x_C}\tau_C(x_C)\log\tau_C(x_C)$$

- For each clique C, choose some real number $\rho_C \geq 0$
  - We can always choose the $\rho$ such that the resulting approximation is concave
  - Use this as a surrogate for the true entropy

# Reweighted Maximum Entropy

$$\max_{\tau^1,\ldots,\tau^M \in T} \sum_m H_\rho(\tau^m) - \frac{1}{2\lambda} \left\| \sum_m f(x^m, y^m) - \sum_m \sum_C \sum_{x_C} \tau_C^m(x_C|y^m) f_C(x_C, y^m) \right\|_2^2$$

- For appropriate choice of $\rho$ this is a constrained concave optimization problem
- How do we maximize constrained concave functions?
  - Gradient ascent can step outside of the constraint set...
    - Projecting back in can be computationally expensive

# Reweighted Maximum Entropy

$$\max_{\tau^1,\ldots,\tau^M \in T} \sum_m H_\rho(\tau^m) - \frac{1}{2\lambda}\left\| \sum_m f(x^m, y^m) - \sum_m \sum_C \sum_{x_C} \tau_C^m(x_C | y^m) f_C(x_C, y^m) \right\|_2^2$$

- This approximate maximum entropy optimization problem is dual to an approximate MLE optimization problem where we approximate $Z$ using the Bethe free energy with a concave entropy approximation
  - Note: duality holds when this problem is concave and you choose the same $\rho$ for both max-entropy and MLE

# Gradient Descent

- Let's suppose that we want to minimize a convex function $f(x)$ over a convex set $S$

- Start with an initial point $x^0 \in S$

$$x^t = x^{t-1} - \gamma_t \nabla f(x^{t-1})$$

  – $\gamma_t$ is a step size

- Idea: step along a decreasing direction

# Method of Conditional Gradients

- Also known as the Frank-Wolfe algorithm
- To minimize a convex function over a convex set, it suffices to solve a series of linear optimization problems
- Let's suppose that we want to minimize a convex function $f(x)$ over a convex set $S$
- Start with an initial point $x^0 \in S$

$$s^t = \arg\min_{x \in S} \langle x, \nabla f(x^{t-1}) \rangle$$

$$x^t = (1 - \gamma_t) x^{t-1} + \gamma_t s^t$$

# Method of Conditional Gradients

- Start with an initial point $x^0 \in S$

$$s^t = \arg\min_{x \in S} \langle x, \nabla f(x^{t-1}) \rangle$$

$$x^t = (1 - \gamma_t) x^{t-1} + \gamma_t s^t$$

- $\gamma_t$ is the step size
  - The algorithm is guaranteed to converge if $\gamma_t = \frac{2}{2+t}$
  - Other choices are also possible

# Reweighted Maximum Entropy

$$Ent(\tau^1, \dots, \tau^M) = \sum_m H_\rho(\tau^m) - \frac{1}{2\lambda} \left\| \sum_m f(x^m, y^m) - \sum_m \sum_C \sum_{x_C} \tau_C^m(x_C | y^m) f_C(x_C, y^m) \right\|_2^2$$

- To apply FW, need to compute the gradient with respect to $\tau^1, \dots, \tau^M$
- No matter what it ends up being, the optimization we need to solve is

$$\arg \max_{\mu^1, \dots, \mu^m \in T} \langle \mu, \nabla Ent(\tau^1, \dots, \tau^M) \rangle$$

- This is a linear programming problem over the local polytope
  - This means it corresponds to solving an approximate MAP problem!

# MAP LP

$$\max_{\tau} \sum_{i \in V} \sum_{x_i} \tau_i(x_i) \log \phi_i(x_i) + \sum_{(i,j) \in E} \sum_{x_i, x_j} \tau_{ij}(x_i, x_j) \log \psi_{ij}(x_i, x_j)$$

such that

$$\sum_{x_i} \tau_i(x_i) = 1$$ For all $i \in V$

$$\sum_{x_j} \tau_{ij}(x_i, x_j) = \tau_i(x_i)$$ For all $(i,j) \in E, x_i$

$$\tau_i(x_i) \in [0,1]$$ For all $i \in V, x_i$

$$\tau_{ij}(x_i, x_j) \in [0,1]$$ For all $(i,j) \in E, x_i, x_j$

# Reweighted Maximum Entropy

$$Ent(\tau^1, \dots, \tau^M) = \sum_m H_\rho(\tau^m) - \frac{1}{2\lambda} \left\| \sum_m f(x^m, y^m) - \sum_m \sum_C \sum_{x_C} \tau_C^m(x_C | y^m) f_C(x_C, y^m) \right\|_2^2$$

- Can solve this optimization problem just by solving a series of approximate MAP (linear programming problems)
  - Many general purpose solvers exist for LPs
  - Could use belief propagation!

# Reweighted Sum-Product

- We know that fixed points of loopy BP correspond to local optima of the Bethe free energy

- Is there an analog of sum-product for each choice of $\rho$?

  – Yes!

# Reweighted Sum-Product

- $p(x_1, \dots, x_n) = \frac{1}{Z} \prod_{i \in V} \phi_i(x_i) \prod_{(i,j) \in E} \psi_{ij}(x_i, x_j)$

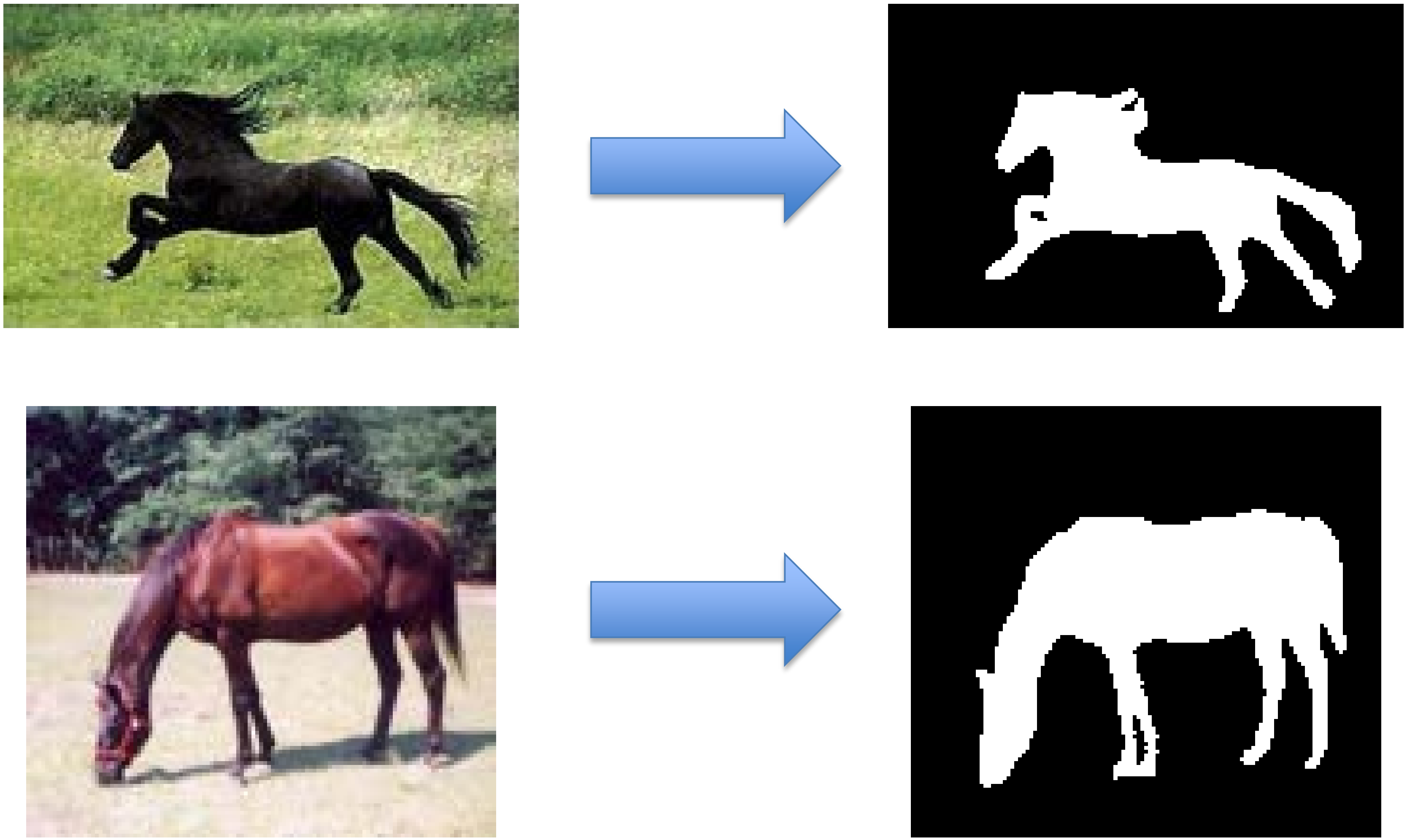
$$m_{i \to j}(x_j) = \sum_{x_i} \phi_i(x_i) \psi_{ij}(x_i, x_j)^{\frac{1}{\rho_{ij}}} \left[ \frac{\prod_{k \in N(i)} m_{k \to \mathrm{i}}(x_i)^{\rho_{ki}}}{m_{j \to i(x_i)}} \right]$$

- $\rho = \vec{1}$ is equal to regular belief propagation

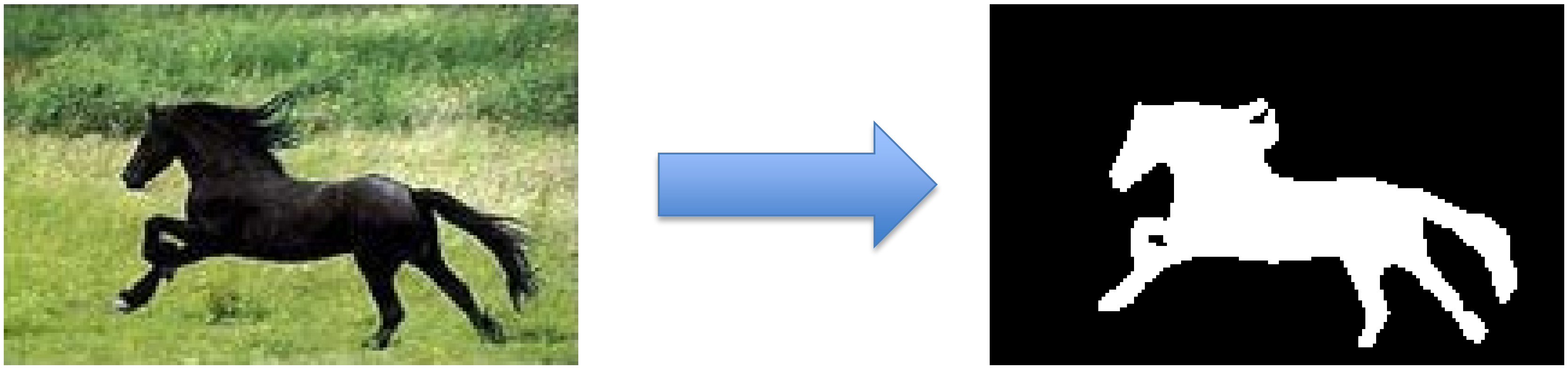


UTD

# Image Segmentation



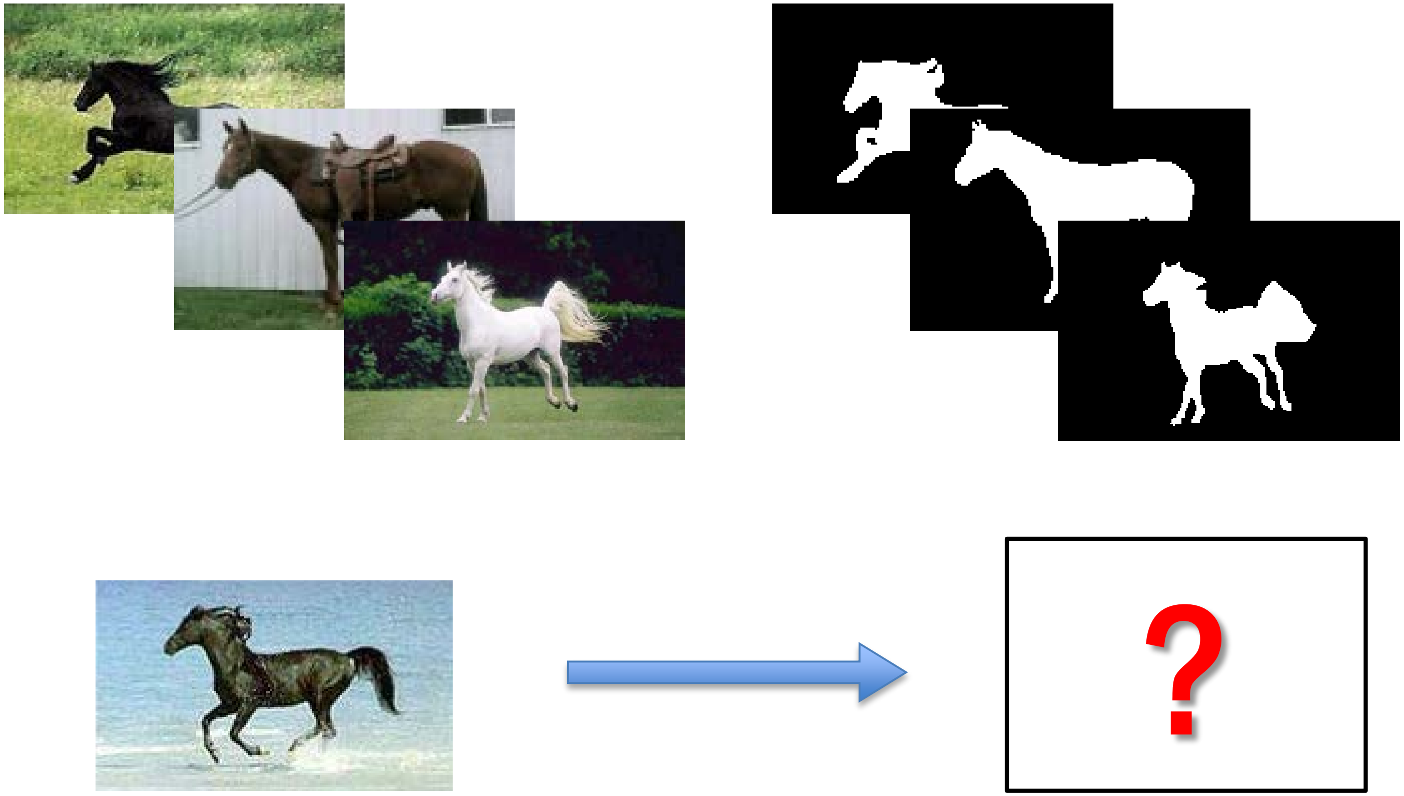
# Image Segmentation



This image is 159x100 = 15,900 pixels

$2^{15,900}$ different possible segmentations!

# Image Segmentation

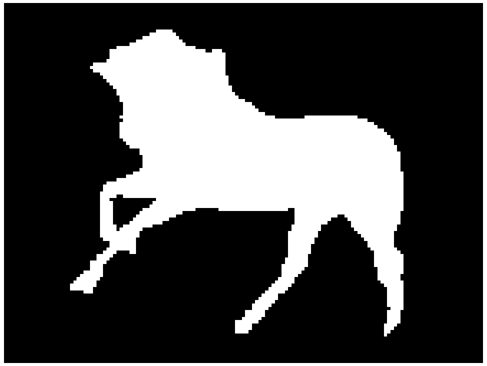



Given a set of labeled training examples, we want to learn the weights of an Ising model (with features) to correctly predict the segmentation of an unseen horse

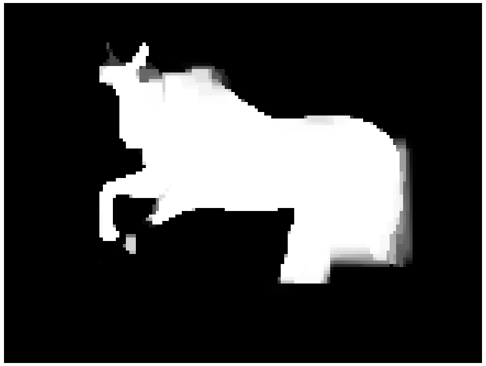
# Image Segmentation

Unseen Test Image

Ground Truth Segmentation



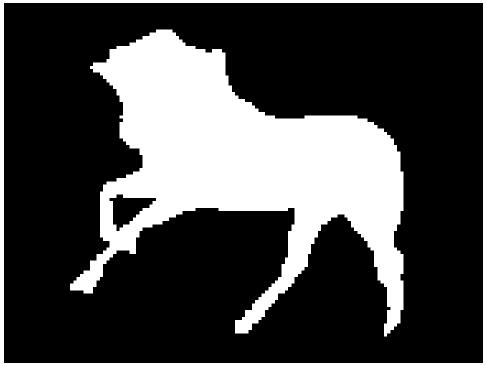
100 iterations
(9 mins)



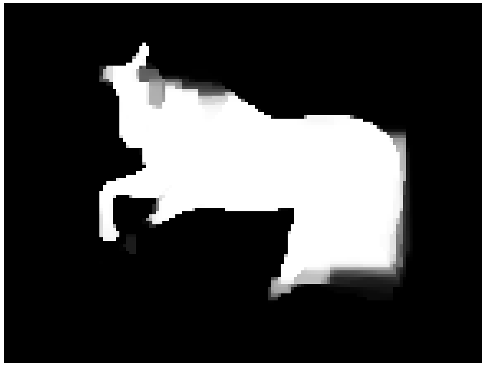
# Image Segmentation

Unseen Test Image

Ground Truth Segmentation



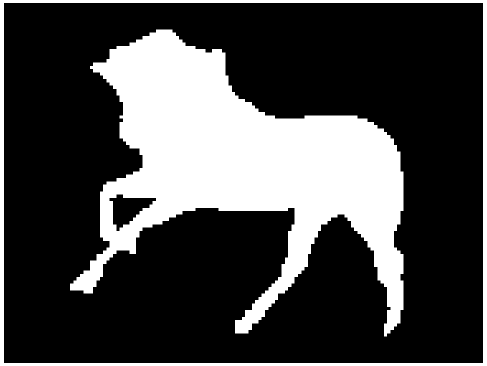
250 iterations



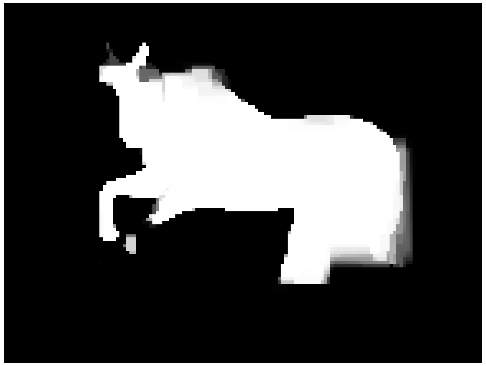
# Image Segmentation

Unseen Test Image

Ground Truth Segmentation



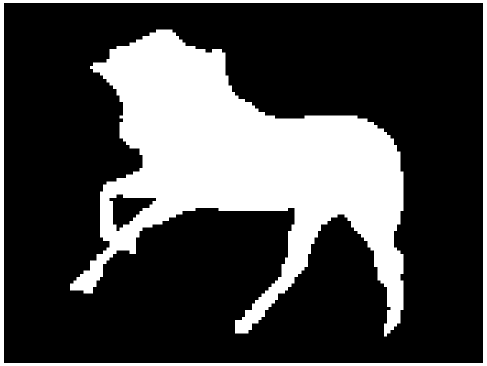
2,000 iterations



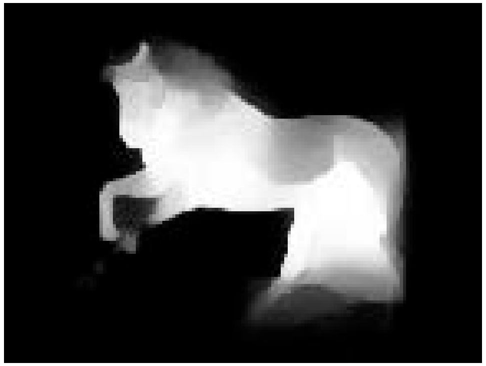
# Image Segmentation

Unseen Test Image

Ground Truth Segmentation



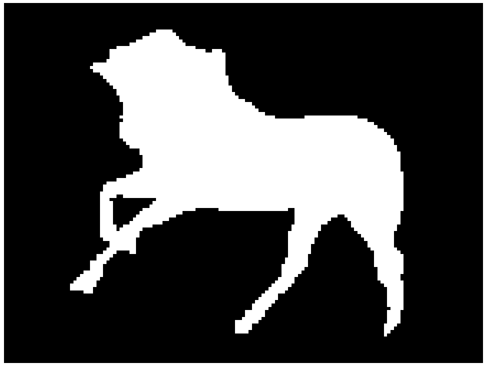
11,750 iterations



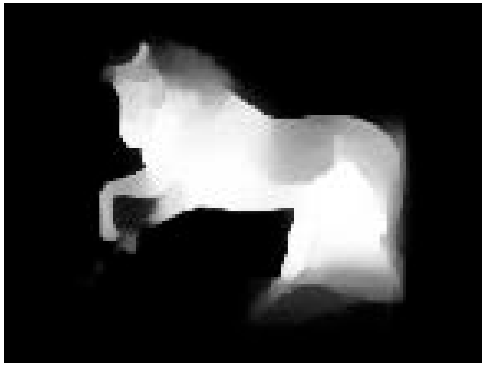
# Image Segmentation

Unseen Test Image

Ground Truth Segmentation



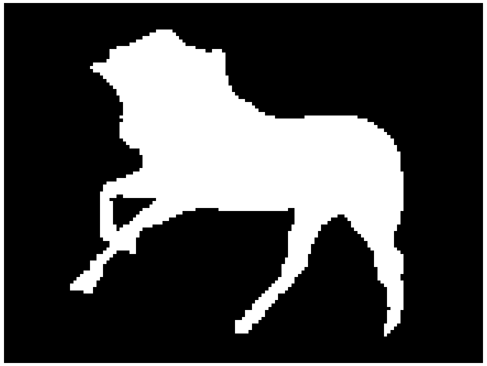
100,000 iterations



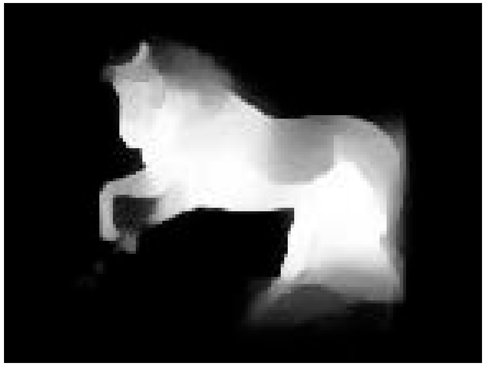
# Image Segmentation

Unseen Test Image

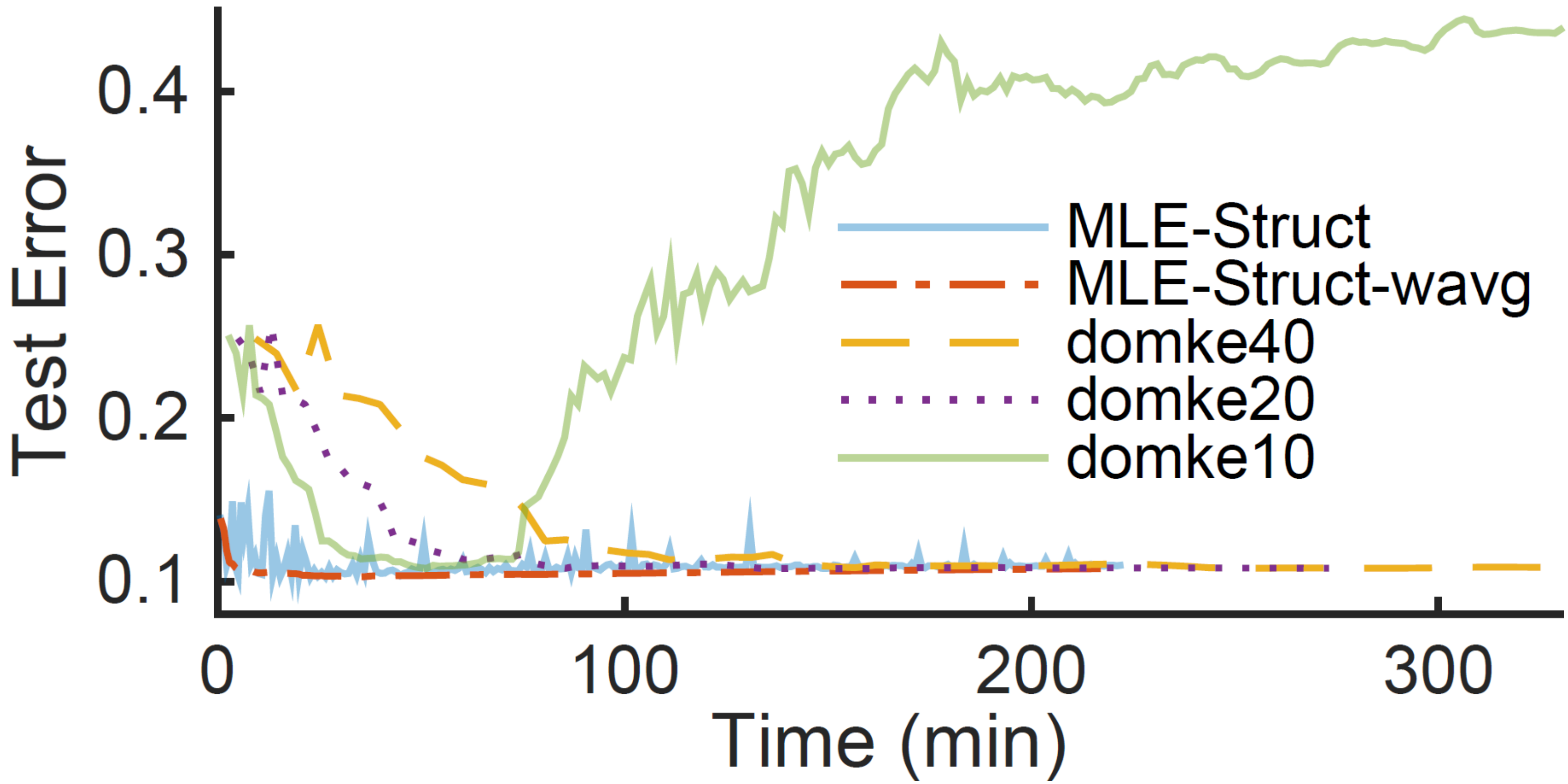
Ground Truth Segmentation



250,000 iterations
(3.7 hours)



# Test Error Over Time



# Hidden Variables

- So far, we've only considered the case where all of the variables in the model were fully observed

- How do we handle situations in which some of the variables are hidden?

- Given a MRF over observed variables $x$ and hidden variables $h$, we can still write down the log-likelihood

$$\log \ell(\theta) = \sum_m \log p(x^m|\theta)$$
$$= \sum_m \sum_h \log p(x^m, h|\theta)$$

# Hidden Variables

- So far, we've only considered the case where all of the variables in the model were fully observed

- How do we handle situations in which some of the variables are hidden?

- Given a MRF over observed variables $x$ and hidden variables $h$, we can still write down the log-likelihood

$$\log \ell(\theta) = \sum_m \log p(x^m|\theta)$$
$$= \sum_m \sum_h \log p(x^m, h|\theta)$$

NOT concave in $\theta$!