# CPSC 540: Machine Learning

## Log-Linear Models

Mark Schmidt

University of British Columbia

Winter 2020

# Last Time: Approximate Inference

- We've been discussing graphical models for density estimation,

$$p(x_1, x_2, \ldots, x_d) = \prod_{j=1}^{d} p(x_j \mid x_{\mathsf{pa}(j)}), \quad p(x_1, x_2, \ldots, x_d) \propto \prod_{c \in \mathcal{C}} \phi_c(x_c),$$

  where are natural and widely-used models for many phenomena.
  - These will also be among ingredients of more advanced models we'll see later.

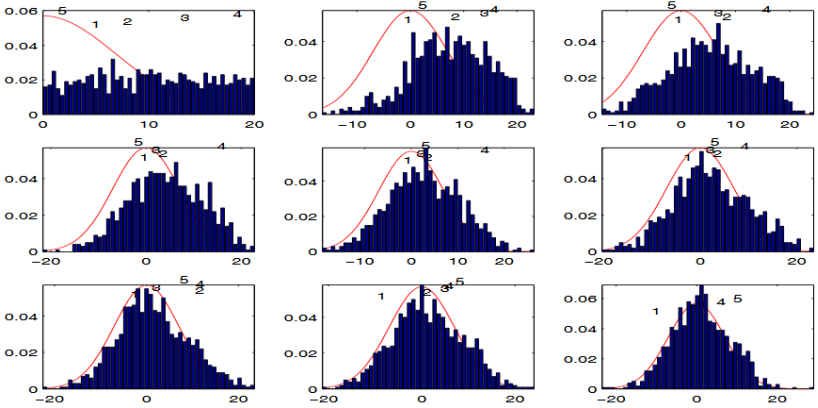
- But most calculations involving graphical models are typically NP-hard.
  - We can convert to DAGs to UGMs, so we'll just study UGMs.

- We considered approximate inference in discrete UGMs:
  1. Iterated conditional mode (ICM) applies coordinate-wise optimization.
  2. Gibbs sampling applies coorrdinate-wise sampling.
     - A special case of Markov chain Monte Carlo (MCMC).

# Markov Chain Monte Carlo

MCMC sampling from a Gaussian:

From top left to bottom right: histograms of 1000 independent Markov chains with a normal distribution as target distribution.



http://www.cs.ubc.ca/~arnaud/stat535/slides10.pdf

# MCMC Implementation Issues

- Basic idea of Markov Chain Monte Carlo (MCMC) method:
  - Design a Markov chain that has $\pi(x) = p(x)$.
  - Use these samples within a Monte Carlo estimator,

$$\mathbb{E}[g(x)] \approx \frac{1}{n} \sum_{t=1}^{n} g(x^i).$$

- In practice, we often don't take all samples in our Monte Carlo estimate:
  - Burn in: throw away the initial samples when we haven't converged to stationary.
  - Thinning: only keep every $k$ samples, since they will be highly correlated.

# MCMC Implementation Issues

- Two common ways that MCMC is applied:
  1. Sample from a huge number of Markov chains for a long time, use final states.
     - Great for parallelization.
     - No need for thinning, if chains are independently initialized.
     - Need to worry about burn in.
  2. Sample from one Markov chain for a really long time, use states across time.
     - Less worry about burn in.
     - Need to worry about thinning.

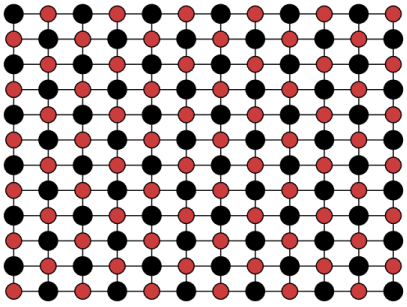
- It can very hard to diagnose if we have reached stationary distribution.
  - Recent work showed that this is P-space hard (*not* polynomial-time even if P=NP).
  - Various heuristics exist.

# Block-Structured Approximate Inference

- Basic approximate inference methods like ICM and Gibb sampling:
  - Update one $x_j$ at a time.
  - Efficient because conditional UGM is 1 node.
- Better approximate inference methods use block updates:
  - Update a block of $x_j$ values at once.
  - Efficient if conditional UGM allows exact inference.
- If we choose the blocks cleverly, this works substantially better.

# Block-Structured Approximate Inference

- Consider a lattice-structure and the following two blocks ("red-black ordering"):

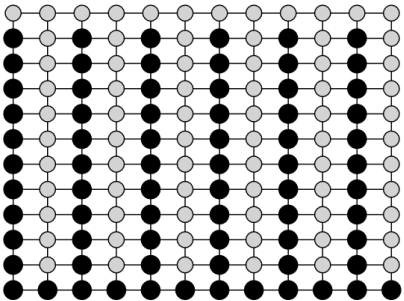


- Given black nodes, conditional UGM on red nodes is a disconnected graph.
  - "I can optimally update the red nodes given the black nodes" (and vice versa).
    - You update $d/2$ nodes at once for cost of this is $O(dk)$, and easy to parallelize.

- Minimum number of blocks to disconnect the graph is graph colouring.

# Block-Structured Approximate Inference

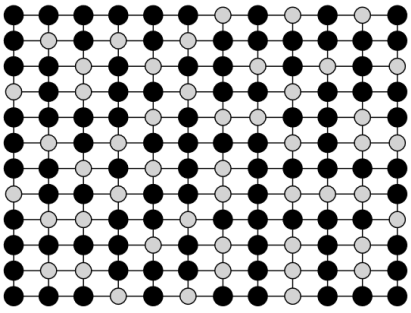
- We could also consider general forest-structured blocks:



- We can still optimally update the black nodes given the gray nodes in $O(dk^2)$.
  - This works much better than "one at a time".

# Block-Structured Approximate Inference

- Or we could define a new tree-structured block on each iteration:

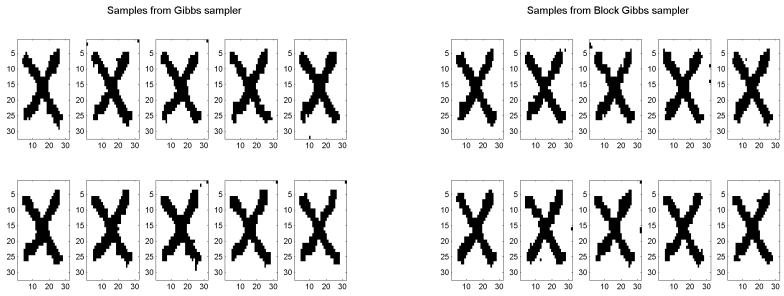


- The above block updates around two thirds of the nodes optimally.

(Here we're updating the black nodes.)

# Block Gibbs Sampling in Action

- Gibbs vs. tree-structured block-Gibbs samples:




- With block sampling, the samples are far less correlated.
- We can also do tree-structured block ICM.
  - Harder to get stuck if you get to update entire trees.

# Block ICM Based on Graph Cuts

- Consider a binary pairwise UGMs with "attractive" potentials,

$$\log \phi_{ij}(1,1) + \log \phi_{ij}(2,2) \geq \log \phi_{ij}(1,2) + \log \phi_{ij}(2,1).$$

- In words: "neighbours prefer to have similar states".

- In this setting exact decoding can be formulated as a max-flow/min-cut problem.
  - Can be solved in polynomial time.

- This is widely-used computer vision:
  - Want neighbouring pixels/super-pixels/regions to be more likely to get same label.

# Graph Cut Example: "GrabCut"

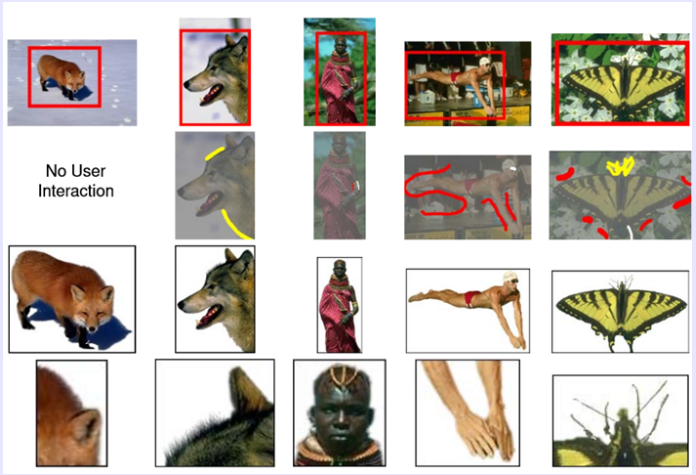
Figure 1: **Three examples of GrabCut.** The user drags a rectangle loosely around an object. The object is then extracted automatically.

http://cvg.ethz.ch/teaching/cvl/2012/grabcut-siggraph04.pdf

1. User draws a box around the object they want to segment.
2. Fit Gaussian mixture model to pixels inside the box, and to pixels outside the box.
3. Construct a pairwise UGM using:
   - $\phi_i(x_i)$ set to GMM probability of pixel $i$ being in class $x_i$.
   - $\phi_{ij}(x_i, x_j)$ set to Ising potential times RBF based on spatial/colour distance.
     - Use $w_{ij} > 0$ so the model is "attractive".
4. Perform exact decoding in the binary attractive model using graph cuts.

# Graph Cut Example: "GrabCut"

- GrabCut with extra user interaction:




http://cvg.ethz.ch/teaching/cvl/2012/grabcut-siggraph04.pdf

# Alpha-Beta Swap and Alpha-Expansions: ICM with Graph Cuts

- If we have more than 2 states, we can't use graph cuts.
- Alpha-beta swaps are an approximate decoding method for "pairwise attractive",

$$\log \phi_{ij}(\alpha, \alpha) + \log \phi_{ij}(\beta, \beta) \geq \log \phi_{ij}(\alpha, \beta) + \log \phi_{ij}(\beta, \alpha).$$

  - Each step choose an $\alpha$ and $\beta$, optimally "swaps" labels among these nodes.
- Alpha-expansions are another variation based on a slightly stronger assumption,

$$\log \phi_{ij}(\alpha, \alpha) + \log \phi_{ij}(\beta_1, \beta_2) \geq \log \phi_{ij}(\alpha, \beta_1) + \log \phi_{ij}(\beta_2, \alpha).$$

  - Steps choose label $\alpha$, and consider replacing the label of any node not labeled $\alpha$.

# Alpha-Beta Swap and Alpha-Expansions: ICM with Graph Cuts

- These don't find global optima in general, but make huge moves:

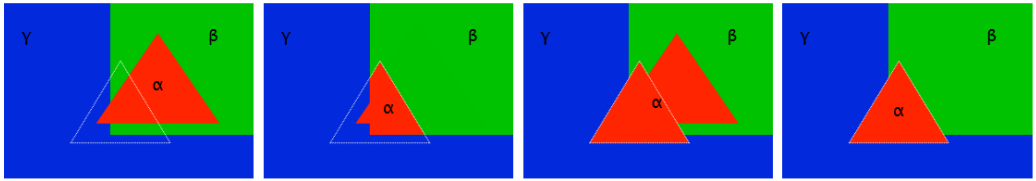



Figure 1: From left to right: Initial labeling, labeling after $\alpha\beta$-swap, labeling after $\alpha$-expansion, labeling after $\alpha$-expansion $\beta$-shrink. The optimal labeling of the $\alpha$ pixels is outlined by a white triangle, and is achieved from the initial labeling by one $\alpha$-expansion $\beta$-shrink move.

- A somewhat-related MCMC method is the Swendson-Wang algorithm.

# Example: Photomontage

- Photomontage: combining different photos into one photo:

http://vision.middlebury.edu/MRF/pdf/MRF-PAMI.pdf

- Here, $x_i$ corresponds to identity of original image at position $i$.

# Example: Photomontage

- Photomontage: combining different photos into one photo:

http://vision.middlebury.edu/MRF/pdf/MRF-PAMI.pdf

# Outline

# Structured Prediction with Undirected Graphical Models

- Consider a pairwise UGM,

$$p(x) = \frac{1}{Z}\left(\prod_{j=1}^{d}\phi_j(x_j)\right)\left(\prod_{(j,k)\in E}\phi_{jk}(x_j,x_k)\right).$$

- We've been focusing on the case where the potentials $\phi$ are known.
  - We've discussed decoding, marginalization, and sampling.
  - We've discussed [block-]coordinate approximate inference.

- We're now going to discuss learning the potentials $\phi$ from data.

- Unfortunately, $Z$ makes this complicated compared to DAGs.
  - You can't fit each potential independently.

# Naive Parameterization of UGMs

- We'll want to make the $\phi$ depend on a set of parameters $w$.

- As before, with $n$ IID training $x^i$ we can do MAP estimation,

$$w = \underset{w}{\text{argmin}} -\sum_{i=1}^{n} \log p(x^i \mid w) + \frac{\lambda}{2}\|w\|^2,$$

  where I've assumed an independent Gaussian prior on $w$.

- A naive parameterization is to just directly treat potentials as parameters:

$$\phi_j(s) = w_{j,s}, \quad \phi_{jk}(s, s') = w_{j,k,s,s'},$$

  so $w_{j,s}$ is "potential of node $j$ being in state $s$".
  - And optimize subject to all parameters being non-negative.
  - This unfortunately leads to a non-convex optimizaiton.

# Log-Linear Parameterization of UGMs

- Instead of using non-negative $w$, we can instead exponentiate $w$,

$$\phi_j(s) = \exp(w_{j,s}), \quad \phi_{jk}(s, s') = \exp(w_{j,k,s,s'}).$$

- This gives a log-linear model,

$$p(x \mid w) \propto \left(\prod_{j=1}^{d} \phi_j(x_j)\right)\left(\prod_{(j,k)\in E} \phi_{jk}(x_j, x_k)\right)$$
$$= \exp\left(\sum_{j=1}^{d} w_{j,x_j} + \sum_{(j,k)\in E} w_{j,k,x_j,x_k}\right),$$

and leads to a convex NLL.

- Normally, exponentiating to get non-negativity introduces local minima.

# Parameter Tieing in UGMs

- So our log-linear parameterization has the form

$$\log \phi_j(s) = w_{j,s}, \quad \log \phi_{jk}(s, s') = w_{j,k,s,s'},$$

  which can represent any positive pairwise potentials.

- There exist many common variations on parameter tieing:
  - We might want $w_{j,x_j}$ to be the same for all $j$ (all nodes use same potentials).
    - You can similarly tie the edge parameters across all edges.
    - This is similar to homogenous Markov chains.
  - In the Ising model we tied across states: $w_{j,k,1,1} = w_{j,k,2,2}$ and $w_{j,k,1,2} = w_{j,k,2,1}$.
  - We could also have special potentials for the boundaries.
    - Many language models are homogeneous, except for start/end of sentences.

# Energy Function and Feature Vector Representation

- Recall that we use $\tilde{p}(x)$ for the unnormalized probability,

$$p(x) = \frac{\tilde{p}(x)}{Z}.$$

- In physics, the value $E(x) = -\log \tilde{p}(x)$ is called the energy function.

- With the log-linear parameterization, the energy function is linear,

$$-E(X) = \sum_j w_{j,x_j} + \sum_{(j,k)\in E} w_{j,k,x_j,x_k}.$$

- To account for parameter tieing, we often write

$$-E(x) = w^T F(x), \quad \text{or equivalently} \quad p(x) \propto \exp(w^T F(x)),$$

where feature function $F$ counts number of times we use each parameter.

# Example of Feature Function

- Consider the 2-node 1-edge UGM (1)–(2), where each state has 2 values.
  - So we have potentials $\phi_1(x_1)$, $\phi_2(x_2)$, and $\phi_{12}(x_1, x_2)$ and want to have

$$w^T F(x) = w_{1,x_1} + w_{2,x_2} + w_{1,2,x_1,x_2}.$$

- With no parameter tieing and $x = \begin{bmatrix} 2 & 1 \end{bmatrix}$, our parameter vector and features are

$$w = \begin{bmatrix} w_{1,1} \\ w_{1,2} \\ w_{2,1} \\ w_{2,2} \\ w_{1,2,1,1} \\ w_{1,2,1,2} \\ w_{1,2,2,1} \\ w_{1,2,2,2} \end{bmatrix}, \quad F(x) = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix},$$

# Example of Feature Function

- If we instead had Ising potentials (just measuring whether $x_1 = x_2$) we would have

$$w^T F(x) = w_{1,x_1} + w_{2,x_2} + w_{1,2,\text{same}},$$

where $w_{1,2,\text{same}}$ is the parameter specifying how much we want $x_1 = x_2$.

- With no parameter tieing and $x = \begin{bmatrix} 2 & 1 \end{bmatrix}$, our parameter vector and features are

$$w = \begin{bmatrix} w_{1,1} \\ w_{1,2} \\ w_{2,1} \\ w_{2,2} \\ w_{1,2.\text{same}} \end{bmatrix}, \quad F(x) = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix},$$

# UGM Training Objective Function

- With log-linear parameterization, NLL for IID training examples is

$$\begin{aligned} f(w) &= -\sum_{i=1}^{n} \log p(x^i \mid w) = -\sum_{i=1}^{n} \log \left( \frac{\exp(w^T F(x^i))}{Z(w)} \right) \\ &= -\sum_{i=1}^{n} w^T F(x^i) + \sum_{i=1}^{n} \log Z(w) \\ &= -w^T F(X) + n \log Z(w). \end{aligned}$$

  where the $F(X) = \sum_i F(x^i)$ are called the sufficient statistics of the dataset.
  - Given sufficient statistics $F(X)$, we can throw out the examples $x^i$.

    (only go through data once)
- Function $f(w)$ is convex (it's linear plus a big log-sum-exp function).
  - But notice that $Z$ depends on $w$

.

# Log-Linear UGM Gradient

- For 1 example $x$, we showed that NLL with log-linear parameterization is

$$f(w) = -w^T F(x) + \log Z(w).$$

- The partial derivative with respect to parameter $w_j$ has a simple form

$$\begin{aligned}\nabla_{w_j} f(w) &= -F_j(x) + \sum_x \frac{\exp(w^T F(x))}{Z(w)} F_j(x) \\ &= -F_j(x) + \sum_x p(x \mid w) F_j(x) \\ &= -F_j(x) + \mathbb{E}[F_j(x)].\end{aligned}$$

- Observe that derivative of $\log(Z)$ is expected value of feature.

# Summary

- Block approximate inference works better than single-variable methods.
  - Blocks could be defined by trees or to implement graph cuts.
- Log-linear parameterization can be used to learn UGMs:
  - Maximum likelihood is convex, but requires normalizing constant $Z$.
- Next time: the work that started the the modern deep learning movement.

# Example: Ising Model of Rain Data

- E.g., for the rain data we could parameterize our node potentials using

$$\log(\phi_i(x_i)) = \begin{cases} w_1 & \text{no rain} \\ 0 & \text{rain} \end{cases}.$$

- Why do we only need 1 parameter?
  - Scaling $\phi_i(1)$ and $\phi(2)$ by constant doesn't change distribution.

- In general, we only need $(k-1)$ parameters for a $k$-state variable.
  - But if we're using regularization we may want to use $k$ anyways (symmetry).

# Example: Ising Model of Rain Data

- The Ising parameterization of edge potentials,

$$\log(\phi_{ij}(x_i, x_j)) = \begin{cases} w_2 & x_i = x_j \\ 0 & x_i \neq x_j \end{cases}.$$

- Applying gradient descent gives MLE of

$$w = \begin{bmatrix} 0.16 \\ 0.85 \end{bmatrix}, \quad \phi_i = \begin{bmatrix} \exp(w_1) \\ \exp(0) \end{bmatrix} = \begin{bmatrix} 1.17 \\ 1 \end{bmatrix}, \quad \phi_{ij} = \begin{bmatrix} \exp(w_2) & \exp(0) \\ \exp(0) & \exp(w_2) \end{bmatrix} = \begin{bmatrix} 2.34 & 1 \\ 1 & 2.34 \end{bmatrix},$$

  preference towards no rain, and adjacent days being the same.
- Average NLL of 16.8 vs. 19.0 for independent model.

# Full Model of Rain Data

- We could alternately use fully expressive edge potentials

$$\log(\phi_{ij}(x_i, x_j)) = \begin{bmatrix} w_2 & w_3 \\ w_4 & w_5 \end{bmatrix},$$

  but these don't improve the likelihood much.

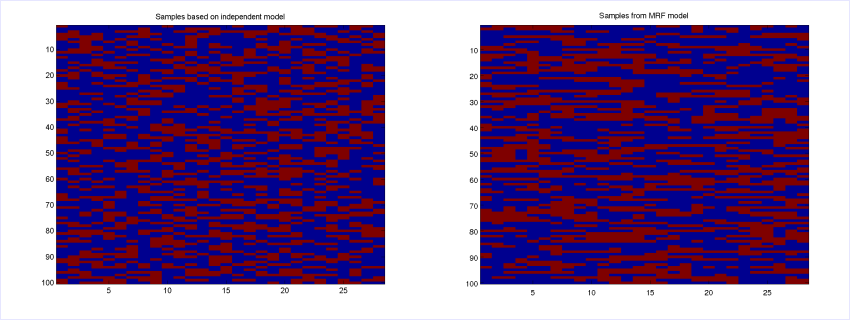
- We could fix one of these at 0 due to the normalization.
  - But we often don't do this when using regularization.

- We could also have special potentials for the boundaries.
  - Many language models are homogeneous, except for start/end of sentences.

# Example: Ising Model of Rain Data

Independent model vs. chain-UGM model with tied nodes and Ising tied edges:

- For this dataset, using untied or general edges doesn't change likelihood much.



# Example: Ising Model of Rain Data

Samples from Ising chain-UGM model if it rains on the first day: