MCMC for UGMs

Jason Hartford

August, 2015

Outline

1. Gibbs for UGMs
   - Sampling from UGMs
   - Relation to MCMC

2. Using MCMC
   - Samples from Gibbs Sampling
   - Inference by averaging across samples
   - Decoding using the temperature parameter
   - Learning via Herding

Recall from yesterday...

- ICM: For each x_i find the mode of the energy function, conditional on $x_{\neg i}$
- Gibbs sampling uses the same idea except instead of approximately decoding, we sample from the conditional distribution

Gibbs Sampling

Basic idea: sample each variable in turn, conditional on the currents values of all other variables in the distribution. E.g. if $D = 3$

- $x_1^{s+1} \sim p(x_1 | x_2^s, x_3^s)$

Gibbs Sampling

Basic idea: sample each variable in turn, conditional on the currents values of all other variables in the distribution. E.g. if $D = 3$

- $x_1^{s+1} \sim p(x_1 | x_2^s, x_3^s)$
- $x_2^{s+1} \sim p(x_2 | x_1^{s+1}, x_3^s)$

Gibbs Sampling

Basic idea: sample each variable in turn, conditional on the currents values of all other variables in the distribution. E.g. if $D = 3$

- $x_1^{s+1} \sim p(x_1 | x_2^s, x_3^s)$
- $x_2^{s+1} \sim p(x_2 | x_1^{s+1}, x_3^s)$
- $x_3^{s+1} \sim p(x_3 | x_1^{s+1}, x_2^{s+1})$

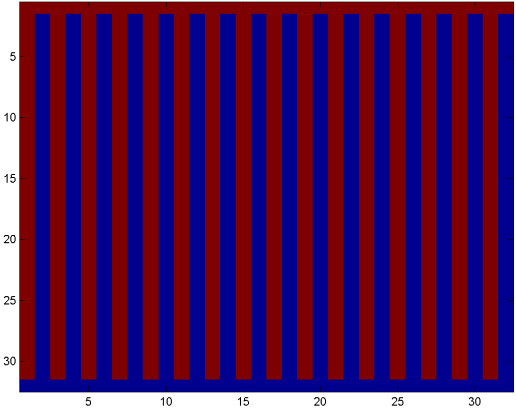
Gibbs Sampling (cont...)

Sampling from the conditional distribution is easy because...

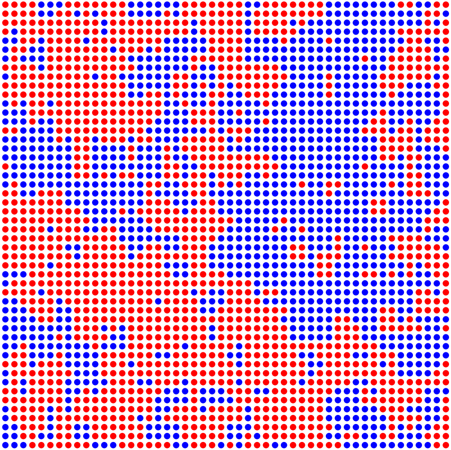
$$
\begin{aligned}
P(\mathbf{x}_i|\mathbf{x}_{\neg i}) &= \frac{P(X_i,\mathbf{x}_{\neg i})}{\sum_{x\in X_i} P(X_i = x,\mathbf{x}_{\neg i})} \\
&= \frac{\tilde{P}(X_i,\mathbf{x}_{\neg i})}{\sum_{x\in X_i} \tilde{P}(X_i = x,\mathbf{x}_{\neg i})} \\
&= \frac{\prod_{k=1}^{n} \phi_k(x_k) \prod_{k,j\in E} \phi_{k,j}(x_k,x_j)}{\sum_{x\in X_i} \prod_{k=1}^{n} \phi_k(x_k) \prod_{k,j\in E} \phi_{k,j}(x_k,x_j)} \\
&= \frac{\phi_i(x_i) \prod_{i,j\in \text{neigh of} i} \phi_{i,j}(x_i,x_j)}{\sum_{x\in X_i} \phi_i(x_i) \prod_{i,j\in \text{neigh of} i} \phi_{i,j}(x_i,x_j)}
\end{aligned}
$$

Block Gibbs

Similarly to yesterday, we can sample from more than one variable at a time by splitting our graph into trees and using the methods from last week to sample.



Block Gibbs



See also Swendson-Wang (1987) for an alternate way of proposing samples with large moves.

Markov Chain Refresher

Great - but how do we know that the samples come from our target distribution? Let's examine the relationship with MCMC. First, a refresher on Markov chains...

Markov Chain Refresher

Great - but how do we know that the samples come from our target distribution? Let's examine the relationship with MCMC. First, a refresher on Markov chains...

Recall from last week:

- A regular Markov chain with transition matrix T converges to an invariant distribution $P^{\star}(x)$ such that $TP^{\star}(x) = P^{\star}(\mathrm{x})$.

Markov Chain Refresher

Great - but how do we know that the samples come from our target distribution? Let's examine the relationship with MCMC. First, a refresher on Markov chains...

Recall from last week:

- A regular Markov chain with transition matrix T converges to an invariant distribution $P^{\star}(x)$ such that $TP^{\star}(x) = P^{\star}(\mathrm{x})$.
- Idea - let's construct a Markov Chain s.t. its invariant distribution is our distribution of interest $P(\mathbf{x})$, then can sample from it by repeatedly applying T.

Detailed Balance

A sufficient (but not necessary) condition for ensuring $P(x)$ is invariant is to choose $T(\cdot)$ such that:

- $T(x' \leftarrow x)P^{\star}(x) = T(x \leftarrow x')P^{\star}(x') \quad \forall x, x'$

Detailed Balance

A sufficient (but not necessary) condition for ensuring $P(x)$ is invariant is to choose $T(\cdot)$ such that:

- $T(x' \leftarrow x)P^{\star}(x) = T(x \leftarrow x')P^{\star}(x') \quad \forall x, x'$
- Proof:

$$\sum_{x'} T(x \leftarrow x')P^{\star}(x') = \sum_{x'} T(x' \leftarrow x)P^{\star}(x)$$

$$P^{\star}(x)\sum_{x'} P(x'|x) = P^{\star}(x)$$

$$TP^{\star}(x) = P^{\star}$$

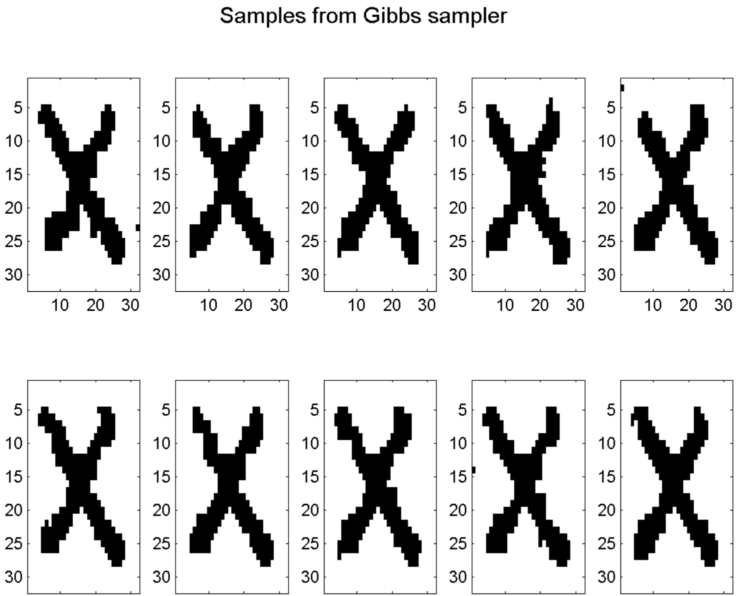
Gibbs Sampling

For Gibbs Sampling we apply a series of transition operators T_i.

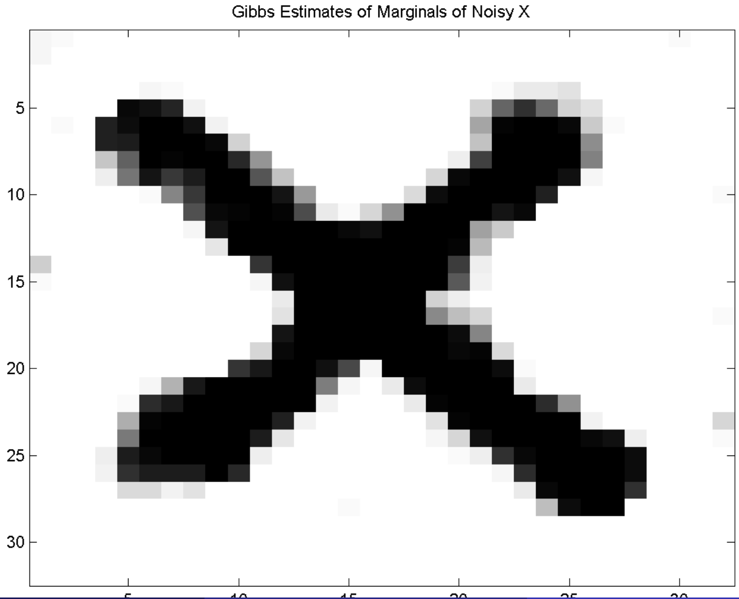
$$T_i(x' \leftarrow x) = P(x_i = x' | x_{\neg i})$$

So detailed balance applies for each component, and by combining operators $T_1, T_2, \dots T_n$ we can reach any $\mathbf{x}$.

Samples



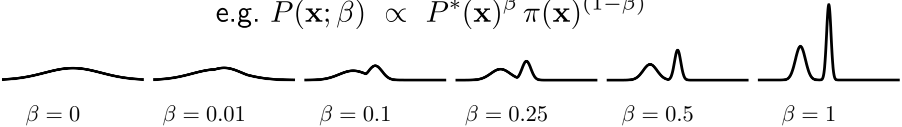
Samples

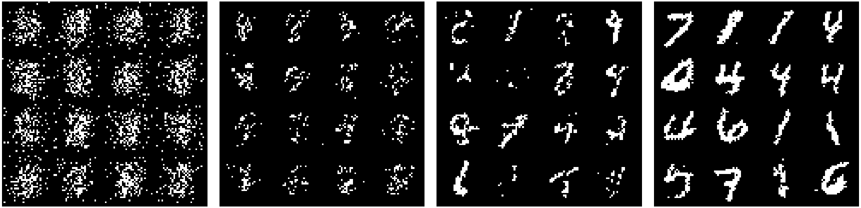

Gibbs Estimates of Marginals of Noisy X
5
10
15
20
25
30
5
10
15
20
25
30

Decoding using the temperature parameter

For some temperature $T = \frac{1}{\beta}$

e.g. $P(\mathbf{x}; \beta) \;\propto\; P^*(\mathbf{x})^{\beta}\, \pi(\mathbf{x})^{(1-\beta)}$





Decoding using the temperature parameter

So simulated annealing works as follows:

- Draw sample using Gibbs with temperature T
- Reduce T by some ϵ

Learning via Herding

We focus on herding for structured prediction. Recall from Monday:

- Given data $\mathcal{D} = \{(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})\}_{n=1}^{N}$ drawn from $P(\mathbf{x}, \mathbf{y})$ we learn $f : \mathcal{X} \to \mathcal{Y}$ where $\mathbf{y} = (y_1, \dots, y_m)$ and each $y_i \in \{1, \dots, K\}$. Finally $\mathbf{y}_\alpha$ denotes some subset of $\mathbf{y}$
- We learn a linear prediction rule of the form:

$$\hat{\mathbf{y}} = f(\mathbf{x}, \mathbf{y}) = \arg\max_{y \in \mathcal{Y}} \sum_{\alpha} w_\alpha \psi_\alpha(\mathbf{y}_\alpha, \mathbf{x})$$

- Running example. y_i is one of K labels associated with each pixel i and we have unary features $\psi(y_i, \mathbf{x})$ for each pixel and pairwise features for each adjacent pixel $\psi_{i,j}(y_i, y_j \mathbf{x})$.
- Standard learning goal: find $\mathbf{w}^\star$

$$\mathbf{w}^\star = \arg\min_{w} \mathcal{L}(\mathcal{D}, \mathbf{w}) = \arg\min_{w} \frac{1}{N} \sum_{n=1}^{N} l(\mathbf{x}^{(n)}, \mathbf{y}^{(n)}, \mathbf{w})$$

Learning via Herding

Consider a conditional Gibbs distribution:

$$p_\tau(\mathbf{y}|\mathbf{x}, \mathbf{w}) = \frac{1}{Z_\tau(\mathbf{x}, \mathbf{w})} \exp\left(\frac{1}{\tau}\sum_\alpha w_\alpha \psi_\alpha(\mathbf{y}_\alpha, \mathbf{x})\right)$$

$$Z_\tau(\mathbf{x}, \mathbf{w}) = \sum_{y \in \mathcal{Y}} \exp\left(\frac{1}{\tau}\sum_\alpha w_\alpha \psi_\alpha(\mathbf{y}_\alpha, \mathbf{x})\right)$$

With $\tau = 1$ this is the CRF Mark presented on Monday. We learn by minimising the loss:

$$l_{\tau,LL}(\mathbf{x}, \mathbf{y}, \mathbf{w}) = -\tau \log p_\tau(\mathbf{y}|\mathbf{x}, \mathbf{w}) = -\sum_\alpha w_\alpha \psi_\alpha(\mathbf{y}_\alpha, \mathbf{x})) + \tau \log Z_\tau(\mathbf{x}, \mathbf{w})$$

Learning via Herding

If we wanted to find the optimal $\mathbf{w}^{\star}$, our gradient update is:

$$w_{\alpha}^{t} = w_{\alpha}^{t-1} + \eta_{\alpha,t}\left(\mathbb{E}_{\hat{P}}[\psi_{\alpha}] - \frac{1}{N}\sum_{n}\sum_{\mathbf{y}'} p_{\tau}(\mathbf{y}'|\mathbf{x}^{(n)}, \mathbf{w}^{t-1})\psi_{\alpha}(\mathbf{y}_{\alpha}, \mathbf{x})\right)$$

Which reaches the optimal $\mathbf{w}^{\star}$ when $\mathbb{E}_{\hat{P}}[\psi_{\alpha}]$ equals the observed moments.

Learning via Herding

If we wanted to find the optimal $\mathbf{w}^\star$, our gradient update is:

$$w_\alpha^t = w_\alpha^{t-1} + \eta_{\alpha,t}\left(\mathbb{E}_{\hat{P}}[\psi_\alpha] - \frac{1}{N}\sum_n \sum_{\mathbf{y}'} p_\tau(\mathbf{y}'|\mathbf{x}^{(n)}, \mathbf{w}^{t-1})\psi_\alpha(\mathbf{y}_\alpha, \mathbf{x})\right)$$

Which reaches the optimal $\mathbf{w}^\star$ when $\mathbb{E}_{\hat{P}}[\psi_\alpha]$ equals the observed moments. But with herding... we don't want to find the optimal $\mathbf{w}^\star$.

Learning via Herding

Instead we take the limit $\tau \to 0$ to get the herding loss:

$$l_{Herd}(\mathbf{x}, \mathbf{y}, \mathbf{w}) = -\sum_{\alpha} w_\alpha \psi_\alpha(\mathbf{y}_\alpha, \mathbf{x}) + \max_{\mathbf{y}'} \left[\sum_{\alpha} w_\alpha \psi_\alpha(\mathbf{y'}_\alpha, \mathbf{x}) \right]$$

Learning via Herding

Instead we take the limit $\tau \to 0$ to get the herding loss:

$$l_{Herd}(\mathbf{x}, \mathbf{y}, \mathbf{w}) = -\sum_{\alpha} w_\alpha \psi_\alpha(\mathbf{y}_\alpha, \mathbf{x}) + \max_{\mathbf{y}'} \left[\sum_{\alpha} w_\alpha \psi_\alpha({\mathbf{y}'}_\alpha, \mathbf{x}) \right]$$

Notice that the above is minimised when $\mathbf{w} = 0$. So it seems pointless to apply subgradient updates... but this is exactly what herding does!

Learning via Herding

Instead we take the limit $\tau \rightarrow 0$ to get the herding loss:

$$l_{Herd}(\mathbf{x}, \mathbf{y}, \mathbf{w}) = -\sum_{\alpha} w_\alpha \psi_\alpha(\mathbf{y}_\alpha, \mathbf{x}) + \max_{\mathbf{y}'} \left[\sum_{\alpha} w_\alpha \psi_\alpha(\mathbf{y'}_\alpha, \mathbf{x}) \right]$$

Notice that the above is minimised when $\mathbf{w} = 0$. So it seems pointless to apply subgradient updates... but this is exactly what herding does!
We update as follows:

$$\hat{\mathbf{y}}^{(n),t} = \arg\max_{\mathbf{y}'} \sum_{\alpha} w_\alpha \psi_\alpha(\mathbf{y'}_\alpha, \mathbf{x})$$

$$w_\alpha^t = w_\alpha^{t-1} + \eta_\alpha \left(\mathbb{E}_{\hat{P}}[\psi_\alpha] - \frac{1}{N} \sum_{n} \psi_\alpha(\hat{\mathbf{y}}_\alpha^{(n),t}, \mathbf{x}^{(n)}) \right)$$

Learning via Herding

The sequence of updates will not converge as long as at least one incorrect prediction is made at every iteration. i.e. $\hat{\mathbf{y}}^{(n),t} \neq y^(n)$ and the sequence $\ldots, \mathbf{w}^{t-1}, \mathbf{w}^{t}, \mathbf{w}^{t+1}, \ldots$ will not diverge as long as some simple conditions are met.

Herding returns $\ldots, \mathbf{w}^{t-1}, \mathbf{w}^{t}, \mathbf{w}^{t+1}, \ldots$ and $\ldots, \mathbf{y}^{t-1}, \mathbf{y}^{t}, \mathbf{y}^{t+1}, \ldots$ such that,

$$\left| \mathbb{E}_{\hat{P}}[\psi_\alpha] - \frac{1}{T}\sum_{t=1}^{T}\frac{1}{N}\sum_{n}\psi_\alpha(\hat{\mathbf{y}}_\alpha^{(n),t}, \mathbf{x}^{(n)}) \right| = \mathcal{O}(\frac{1}{T}) \quad \forall \alpha$$

Herding produces samples converge faster than the $\mathcal{O}(\frac{1}{\sqrt{T}})$ convergence rates of normal Monte Carlo samples from $\hat{P}$, despite not returning $\mathbf{w}^\star$.

Learning via Herding

Is not having $\mathbf{w}^\star$ a problem? Depends on the application.
Works well in image segmentation where you can just average $\mathbf{y}^t$ for some T.

Herding

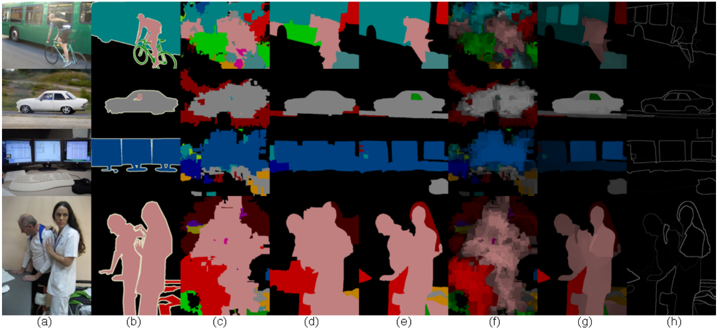
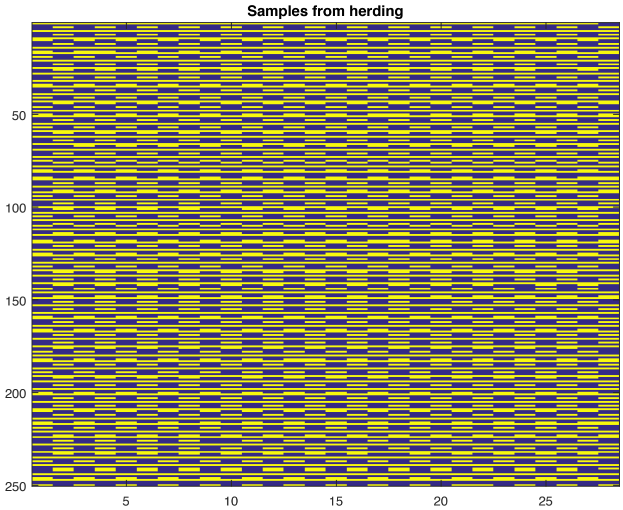


Figure 1.1: Examples of segmentation on Pascal VOC 2007 data set. Images on each line starting from left to right are respectively: (a) the original image, (b) ground truth segmentation, results of (c) local classifier, (d) CRF and (e) Herding, results with intensity proportional to the posterior probability of the (f) local classifier and (g) Herding, and (h) the Herding estimate of the pairwise probability of the existence of a boundary (the corresponding posterior probability for CRF cannot be easily obtained). Neighboring superpixels of a distance up to 3 hops are used for training local SVM. Best viewed in color.

Herding

Not so well for the rain dataset...

Herding



Summary

- We can easily sample from UGM's using conditioning to make Gibbs moves.
- These samples can also be used for decoding and inference.
- Herding is an alternate way of sampling without the intermediate learning step.