Bayesian Learning in Undirected Graphical Models

Zoubin Ghahramani

Gatsby Computational Neuroscience Unit
University College London, UK
http://www.gatsby.ucl.ac.uk/

and

Center for Automated Learning and Discovery
Carnegie Mellon University, USA
http://www.cs.cmu.edu/∼zoubin/

Work with: Iain Murray and Hyun-Chul Kim

Aug 2003, CMU ML Lunch

Undirected Graphical Models

An Undirected Graphical Model (UGM; or Markov Network) is a graphical representation of the dependence relationships between a set of random variables. In an UGM, the joint probability over M variables $\mathbf{x} = [x_1, \ldots, x_M]$, can be written in a factored form:

$$p(\mathbf{x}) = \frac{1}{Z} \prod_{j=1}^{J} g_j(\mathbf{x}_{C_j})$$

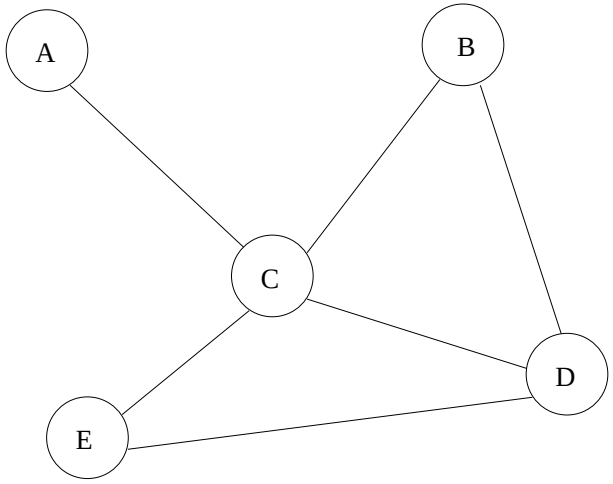
Here the g_j are **non-negative potential functions** over subsets of variables $C_j \subseteq \{1, \ldots, M\}$ and the notation: $\mathbf{x}_S \equiv [x_m : m \in S]$.

The **normalization constant** (a.k.a. partition function) is $Z = \sum_{\mathbf{x}} \prod_j g_j(\mathbf{x}_{C_j})$

We represent this type of probabilistic model graphically.

Graph Definition: Let each variable be a node. Connect nodes i and k if there exists a set C_j such that both $i \in C_j$ and $k \in C_j$. These sets form the *cliques* of the graph (fully connected subgraphs).

Undirected Graphical Models: An Example



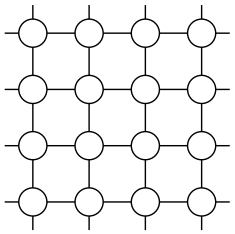
$$p(A, B, C, D, E) = \frac{1}{Z}\, g(A, C) g(B, C, D) g(C, D, E)$$

Markov Property: Every node is conditionally independent from its non-neighbors given its neighbors.

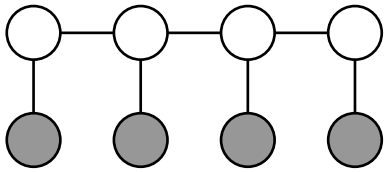
Conditional Independence: $A \perp\!\!\!\perp B | C \Leftrightarrow p(A|B, C) = p(A|C)$ for $p(B, C) > 0$
also $A \perp\!\!\!\perp B | C \Leftrightarrow p(A, B|C) = p(A|C)p(B|C)$.

Applications of Undirected Graphical Models

- Markov Random Fields in Vision, Bioinformatics

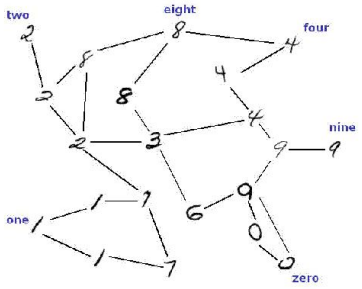


- Conditional Random Fields, and Exponential Language Models, e.g.:



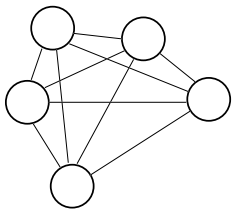
$$p(s) = \frac{1}{Z} p_0(s) \exp\left\{ \sum_i \lambda_i f_i(s) \right\}$$

- Products of Experts: $p(\mathbf{x}) = \frac{1}{Z} \prod_j p_j(\mathbf{x}|\theta_j)$



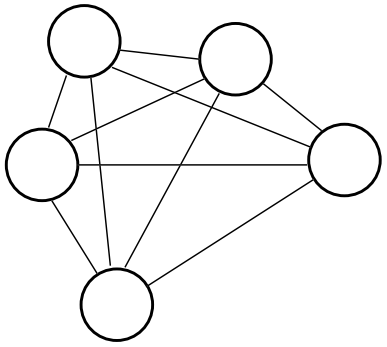
- Semi-Supervised Learning:

⋆ Boltzmann Machines



Boltzmann Machines

Undirected graph over a vector of binary variables $s_i \in \{0, 1\}$. Variables can be hidden or visible (observed).



$$p(\mathbf{s}|W) = \frac{1}{Z} \exp\left\{ \sum_{j<i} W_{ij} s_i s_j \right\}$$

where Z is the *partition function* (normalizer)

Maximum Likelihood Learning Algorithm: a gradient version of EM

- **E step** involves computing averages w.r.t. $p(\mathbf{s}_H|\mathbf{s}_V, W)$ ("clamped phase"). This could be done via an exact message passing algorithm (e.g. Junction Tree) or more usually an approximate method such as Gibbs sampling.
- **M step** also requires gradients w.r.t. Z, which can be computed by averages w.r.t. $p(\mathbf{s}|W)$ ("unclamped phase").

Hebbian and anti-Hebbian rule: $\Delta W_{ij} = \eta[\langle s_i s_j \rangle_c - \langle s_i s_j \rangle_u]$

Bayesian Learning

Prior over parameters: $p(W)$

Posterior over parameters, given data set $\mathcal{S} = \{\mathbf{s}^{(1)}, \ldots \mathbf{s}^{(N)}\}$,

$$p(W|\mathcal{S}) = \frac{p(W)p(\mathcal{S}|W)}{p(\mathcal{S})}$$

Model Comparison (for example for two different graph structures m, m') using Bayes factors:

$$\frac{p(m|\mathcal{S})}{p(m'|\mathcal{S})} = \frac{p(m)}{p(m')} \frac{p(\mathcal{S}|m)}{p(\mathcal{S}|m')}$$

where the marginal likelihood is:

$$p(\mathcal{S}|m) = \int p(\mathcal{S}|W, m)p(W|m)\ dW$$

Why Bayesian Learning?

- Useful prior knowledge can be included (e.g. sparsity, domain knowledge)
- Avoids overfitting (because nothing needs to be fit)
- Error bars on all parameters, and predictions
- Model and feature selection

A Simple Idea

Define the following **joint distribution** of weights W and matrix of binary variables S, organized into N rows (data vectors) and M columns (features, variables). Some variables on some data points may be hidden and some may be observed.

$$p(S, W) = \frac{1}{Z} \exp \left\{ -\frac{1}{2\sigma^2} \sum_{i,j=1}^{M} W_{ij}^2 + \sum_{n=1}^{N} \sum_{j<i}^{M} W_{ij} s_{ni} s_{nj} \right\}$$

Where $Z = \int dW \sum_S \exp\{\ldots\}$ is a nasty partition function.

Gibbs sampling in this model is very easy!

- Gibbs sample s_{ni} given all other $\mathbf{s}$ and W: Bernouilli, easy as usual.
- Gibbs sample W given s: diagonal multivariate Gaussian, easy as well.

What is wrong with this approach?

...a Strange Prior on W

$$p(S,W) = \frac{1}{Z}\exp\left\{-\frac{1}{2\sigma^2}\sum_{i,j=1}^{M} W_{ij}^2 + \sum_{n=1}^{N}\sum_{j<i}^{M} W_{ij}s_{ni}s_{nj}\right\}$$

This defines a Boltzmann machine for the data given W, but defines a **somewhat strange** and hard to compute "prior" on the weights.
What is the prior on W?

$$p(W) = \sum_S p(S,W) \propto N(0,\sigma^2 I)\sum_S \exp\left\{\sum_{n,j<i} W_{ij}s_{ni}s_{nj}\right\}$$

where the second factor is data-size dependent, so it's not a valid hierarchical Bayesian model of the kind $W \to S$. The second factor can be written as:

$$\sum_S \exp\left\{\sum_{n,j<i} W_{ij}s_{ni}s_{nj}\right\} = \left(\sum_s \exp\left\{\sum_{j<i} W_{ij}s_i s_j\right\}\right)^N = Z(W)^N$$

This will not work!

Three Families of Approximations

In order to do Bayesian inference in undirected models with nontrivial partition functions we can develop three classes of methods:

- Approximate Partition Function: $Z(W) = \sum_{\mathbf{s}} \exp\left\{ \sum_{j<i} W_{ij} s_i s_j \right\}$

- Approximate Ratio of Partition Functions.

$$\frac{Z(W)}{Z(W')} = \sum_{\mathbf{s}} p(\mathbf{s}|W) \left[\exp\left\{ \sum_{j<i} (W_{ij} - W'_{ij})\ s_i s_j \right\} \right]$$

- Approximate Gradients. $\frac{\partial \ln Z(W)}{\partial W_{ij}} = \sum_{\mathbf{s}} p(\mathbf{s}|W)\ s_i s_j$

The above quantities can be approximated using modern tools developed in the machine learning/statistics/physics communities.

Surprisingly, none of the following methods have been explored!

I. Metropolis with Nested Sampling

Simplest sampling approach: Metropolis Sampling

- Start with random weight matrix W
- Perturb it with a small-radius Gaussian proposal distribution $W \to W'$
- Accept the change with probability $\min[1, a]$, where

$$a = \frac{p(S|W')\,p(W')}{p(S|W)\,p(W)} = \left(\frac{Z(W)}{Z(W')}\right)^N \exp\left\{\sum_{n,i<j}\left(W'_{ij} - W_{ij}\right) s_i^{(n)} s_j^{(n)}\right\} \frac{p(W')}{p(W)}$$

The partition function ratio is nasty.
But one can estimate it using an MCMC sampling inner loop:

$$\frac{Z(W)}{Z(W')} = \frac{\sum_s \exp\left\{\sum_{j<i} W_{ij} s_i s_j\right\}}{\sum_s \exp\left\{\sum_{j<i} W'_{ij} s_i s_j\right\}} = \left\langle \exp\left\{\sum_{j<i}(W_{ij} - W'_{ij}) s_i s_j\right\}\right\rangle_{p(s|W')}$$

too slow: inner loop can take exponential time

II. Naive Mean-Field Metropolis

Same as above, but use naive mean-field to estimate the partition function. Jensen's inequality gives us:

$$\begin{aligned} \ln Z(W) &= \ln \sum_{\mathbf{s}} \exp\{\sum_{j<i} W_{ij} s_i s_j\} \\ &\geq \sum_{\mathbf{s}} q(\mathbf{s}) \sum_{j<i} W_{ij} s_i s_j + \mathcal{H}(q) = F(W, q) \end{aligned}$$

where $q(s) = \prod_i m_i^{s_i}(1 - m_i)^{(1-s_i)}$ and $\mathcal{H}$ is the entropy.

Gradient-based variant: use expectations to compute approximate gradients

III. Tree Mean-Field Metropolis

Same as above, but use tree-structured mean-field to estimate the partition function. Jensen's inequality gives us:

$$\begin{aligned} \ln Z(W) &= \ln \sum_{\mathbf{s}} \exp\{\sum_{j<i} W_{ij} s_i s_j\} \\ &\geq \sum_{\mathbf{s}} q(\mathbf{s}) \sum_{j<i} W_{ij} s_i s_j + \mathcal{H}(q) = F(W, q) \end{aligned}$$

where $q(s) \in \mathcal{Q}_{\mathbf{t}_{ree}}$, the set of tree-structured distributions and $\mathcal{H}$ is the entropy.

Gradient-based variant: use expectations to compute approximate gradients

IV. Loopy Metropolis

Belief Propagation (BP) is an exact method for inference on trees. Run belief propagation (BP) on the (loopy) graph and use the **Bethe free energy** as an estimate of $Z(W)$. Loopy BP provides on non-trees:

1. approximate marginals $b_i \approx p\left(s_i|W\right)$
2. approximate pairwise marginals $b_{ij} \approx p\left(s_i, s_j|W\right)$

These marginals are fixed points of the Bethe Free energy

$$F_{\text{Bethe}} = U - \mathcal{H}_{\text{Bethe}} \approx -\log Z(W)$$

where U is the expected energy and the approximate entropy is:

$$\mathcal{H}_{\text{Bethe}} = -\sum_{(ij)} \sum_{s_i, s_j} b_{ij}(s_i, s_j) \log b_{ij}(s_i, s_j) - \sum_i (1 - \text{ne}(i)) \sum_{s_i} b_i(s_i) \log b_i(s_i).$$

Gradient-based variant: use expectations to compute approximate gradients

V. The Langevin MCMC Sampling Procedure

So far, we've been describing Metropolis procedures, but these suffer from random walk behaviour.

Langevin makes use of gradient information and resembles noisy steepest descent. This is uncorrected Langevin:

$$W'_{ij} = W_{ij} + \frac{\epsilon^2}{2} \frac{\partial}{\partial W_{ij}} \log p(S, W) + \epsilon \, n_{ij}$$

where $n \sim \mathcal{N}(0, 1)$.

There are many ways of estimating gradients, but we use a method based on Contrastive Divergence (Hinton, 2000).

VI. Pseudo-Likelihood Based Approximations

$$p(\mathbf{s}|W) = \frac{1}{Z} \exp\left\{ \sum_{j<i} W_{ij} s_i s_j \right\}$$

The pseudo-likelihood is defined as

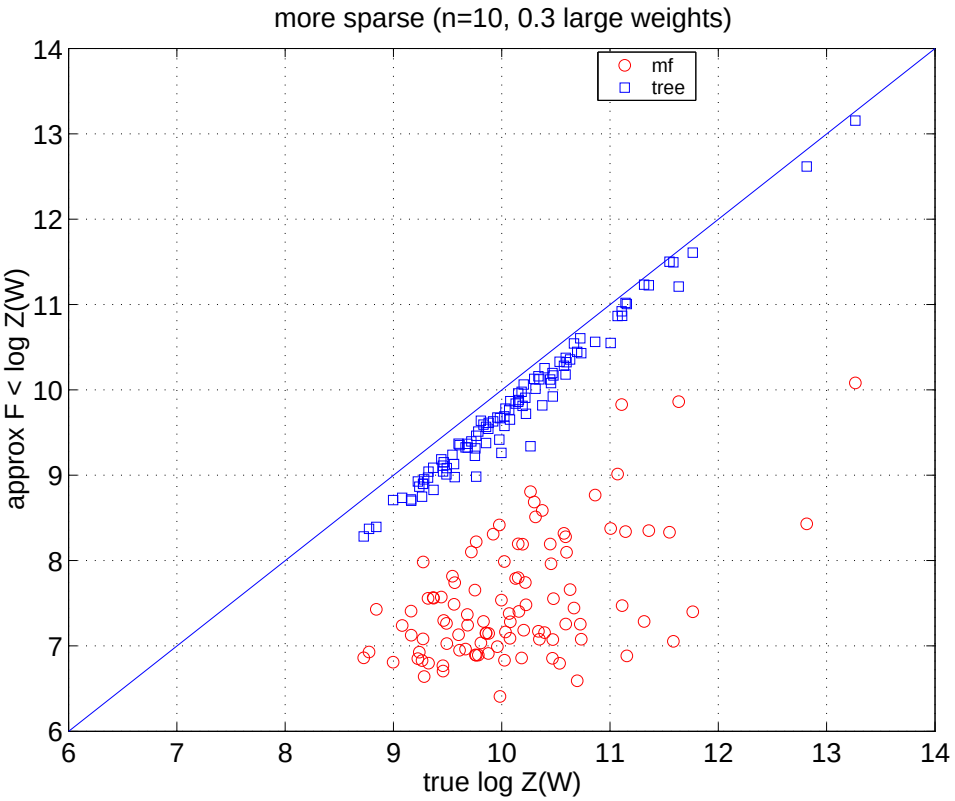
$$\begin{aligned} p(\mathbf{s}|W) &\approx \prod_i p(s_i|\mathbf{s}_{\setminus i}, W) = \prod_i \frac{\exp\{\frac{1}{2} s_i \sum_{j\neq i} W_{ij} s_j\}}{1 + \exp\{\frac{1}{2} \sum_{j\neq i} W_{ij} s_j\}} \\ &= \left[\frac{1}{\prod_i (1 + \exp\{\frac{1}{2} \sum_{j\neq i} W_{ij} s_j\})} \right] \exp\left\{ \sum_{j<i} W_{ij} s_i s_j \right\} \end{aligned}$$

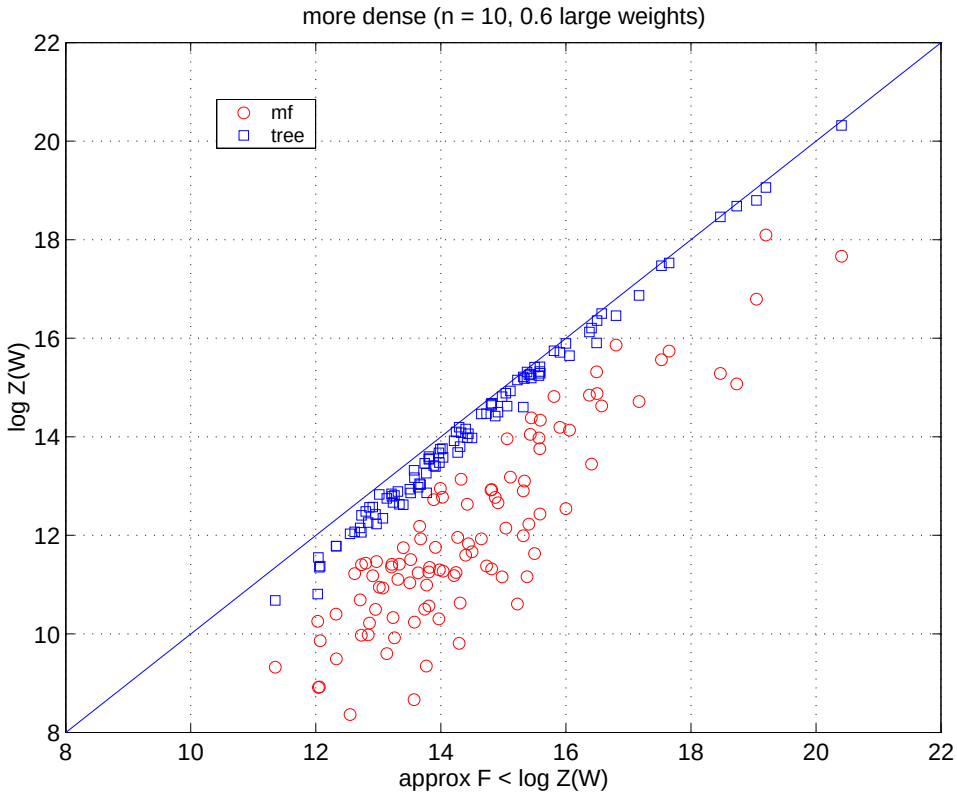
Therefore the use of pseudo-likelihood corresponds to:

$$Z(W) \approx \prod_i \left(1 + \exp\left\{ \frac{1}{2} \sum_{j\neq i} W_{ij} s_j \right\} \right)$$

Has not been tried yet—one can design and compare many other approaches.

Naive Mean Field vs Tree Mean Field Approximation

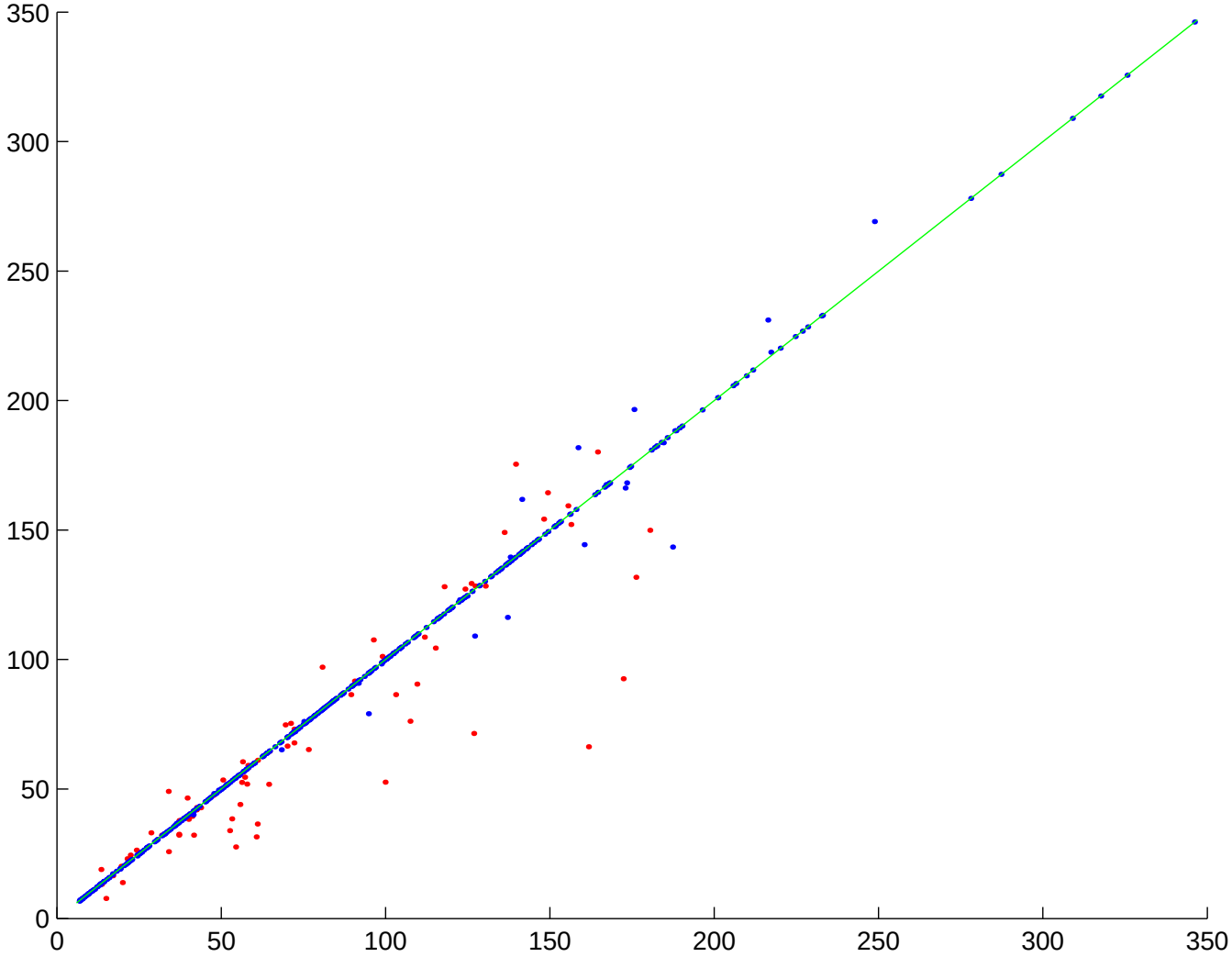




The tree based approximation found an MST and then used Wiegerinck's (UAI, 2000) variational approximation.

Bethe Free Energy

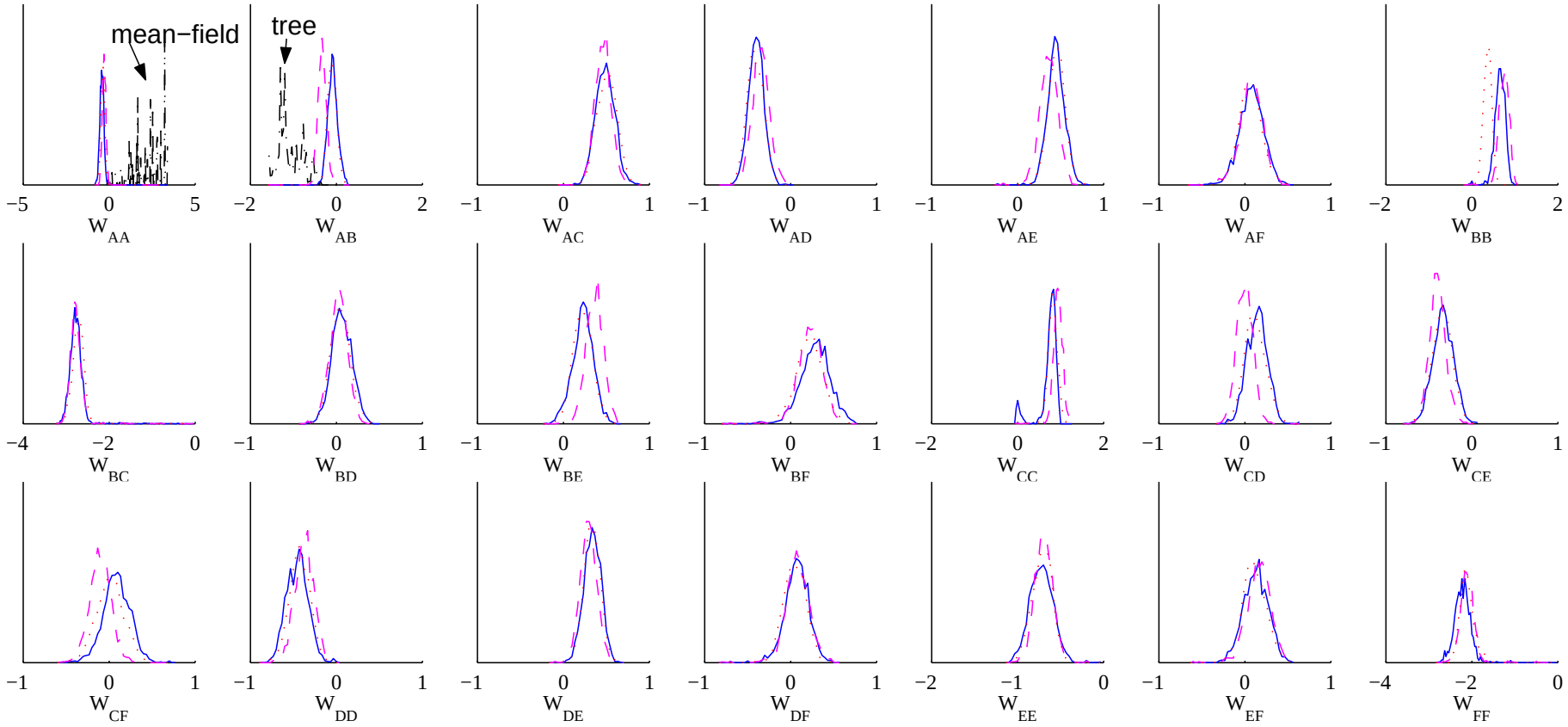
Plots of Z_{Bethe} vs Z_{true} for some independently drawn Boltzmann machines.



Points in red show where belief propagation failed to converge. No hacks were applied to fix up the results; there are ways in the literature.

Results on Coronary Heart Disease Data

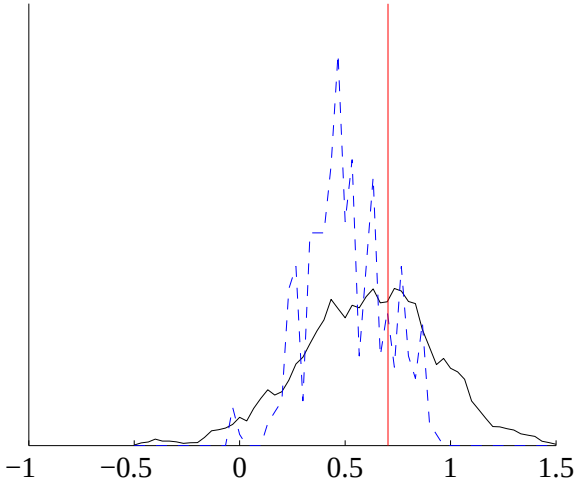
Classic data set of 6 binary variables detailing risk factors for coronary heart disease in 1841 men. Small enough exact $Z(W)$ can be computed.[1] Blue: exact; Red: CD Langevin; Purple: loopy Metropolis.



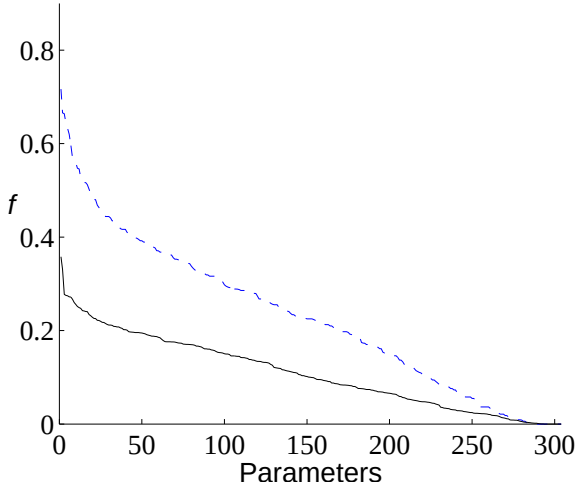
[1] 100000 samples; local Metropolis proposals 0.01 variance; CD Langevin step = 0.01.

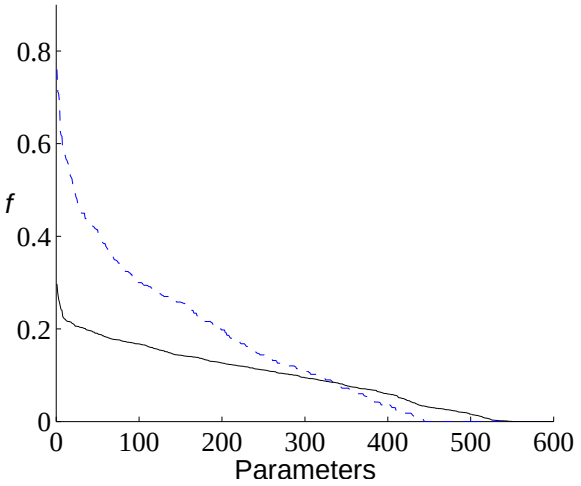
Results on Synthetic Data Sets

100 node random network. 204 and 500 edge systems. Weights $\sim \mathcal{N}(0, 1)$. 100 data points. Dashed Blue: Loopy Metropolis; Black: CD Langevin; Red: true.



f is fraction of samples within ± 0.1 of true parameter value (higher is better):





Part II: Summary and Future Directions

- The problem of Bayesian learning in large tree-width undirected models (log-linear models) appears to have been **completely overlooked** (!?)
- Standard MCMC procedures are **intractable** due to the need to compute partition functions at *each step*.
- This problem offers a natural opportunity for combining modern deterministic approximations with MCMC.
- We have proposed a variety of **novel methods** for approximate MCMC sampling for parameters of undirected models, based on **known ideas**.
- Naive mean field and tree-based mean field Metropolis do not seem to work. Trapped by areas of poor approximation (loose bound).
- The loopy Metropolis and contrastive Langevin both seem to work well. We found Langevin to be more robust.
- Other methods need to be compared.
- Potential applications to text modelling and computer vision.
- There is still **a lot to do** in this area!

End of Talk

Please allow me one more slide...

My Research Interests

- Modelling complex multivariate time series
- Learning Bayesian networks
- Causality
- Semi-supervised learning
- Active learning
- Non-parametric Bayesian methods
- Decision making and control under uncertainty
- Model selection
- Kernel methods
- Sensory-motor control
- Bioinformatcs

I'm looking to co-supervise one of more students in machine learning. Specifically on a project involving modelling the rich multivariate time line of a user's activities on a computer, so as to anticipate user actions and needs. Part of larger Enduring Personalized Cognitive Assistants (EPCA) project at CMU.

Email me: zoubin@cs.cmu.edu

Appendix

Contrastive Divergence[2]

The gradient for maximum likelihood learning:

$$\frac{\partial \log p\left(\mathbf{s}|W\right)}{\partial W_{kl}} \propto \left\langle s_k s_l \right\rangle_{\text{Data}} - \left\langle s_k s_l \right\rangle_{p(\mathbf{s}|W)}$$

becomes

$$\begin{aligned}\frac{\partial \log p\left(\mathbf{s}|W\right)}{\partial W_{kl}} &\propto \left\langle s_k s_l \right\rangle_{p_0(W)} - \left\langle s_k s_l \right\rangle_{p_\infty(W)} \\ &\approx \left\langle s_k s_l \right\rangle_{p_0(W)} - \left\langle s_k s_l \right\rangle_{p_1(W)}\end{aligned}$$

where $p_n(W)$ is defined to be the distribution obtained at the $n^{\text{t}h}$ step of Gibbs sampling *starting from the data*.

[2] Hinton (2000)

Contrastive Divergence for Bayesian Learning

A pretty accurate Taylor expansion makes the comparison easier:

$$\log a + \log \frac{p\left(W\right)}{p\left(W'\right)} = N\left\{\delta\left\langle s_k s_l\right\rangle_{p_0(W)} - \log\left\langle \exp \delta s_k s_l, \right\rangle_{p_\infty(W)}\right\}$$

$$\approx N\delta\left\{\left\langle s_k s_l\right\rangle_{p_0(W)} - \left\langle s_k s_l, \right\rangle_{p_\infty(W)}\right\}$$

It is now tempting to try:

$$\log a + \log \frac{p\left(W\right)}{p\left(W'\right)} = N\delta\left\{\left\langle s_k s_l\right\rangle_{p_0(W)} - \left\langle s_k s_l, \right\rangle_{p_1(W)}\right\}$$

We will call this *contrastive sampling*.