Structured Probabilistic Models and Monte Carlo

Deep Learning Decal

Hosted by Machine Learning at Berkeley

ML@B

Agenda

Background

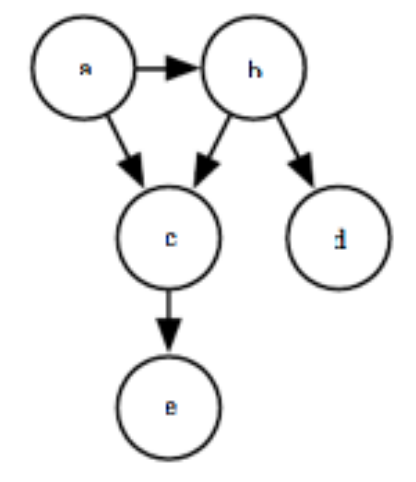
Algorithms

Questions

Background

Purpose of Structured Probabilistic Models

- Split a complicated probability distribution function into a function made of simpler functions based on specific subsets of random variables.
- Random variables $a \sim N(0, 1)$, $b \sim a + N(1, 2)$, $c \sim a + b + N(2, 3)$, $d \sim b + N(3, 4)$, $e \sim c + N(4, 5)$.



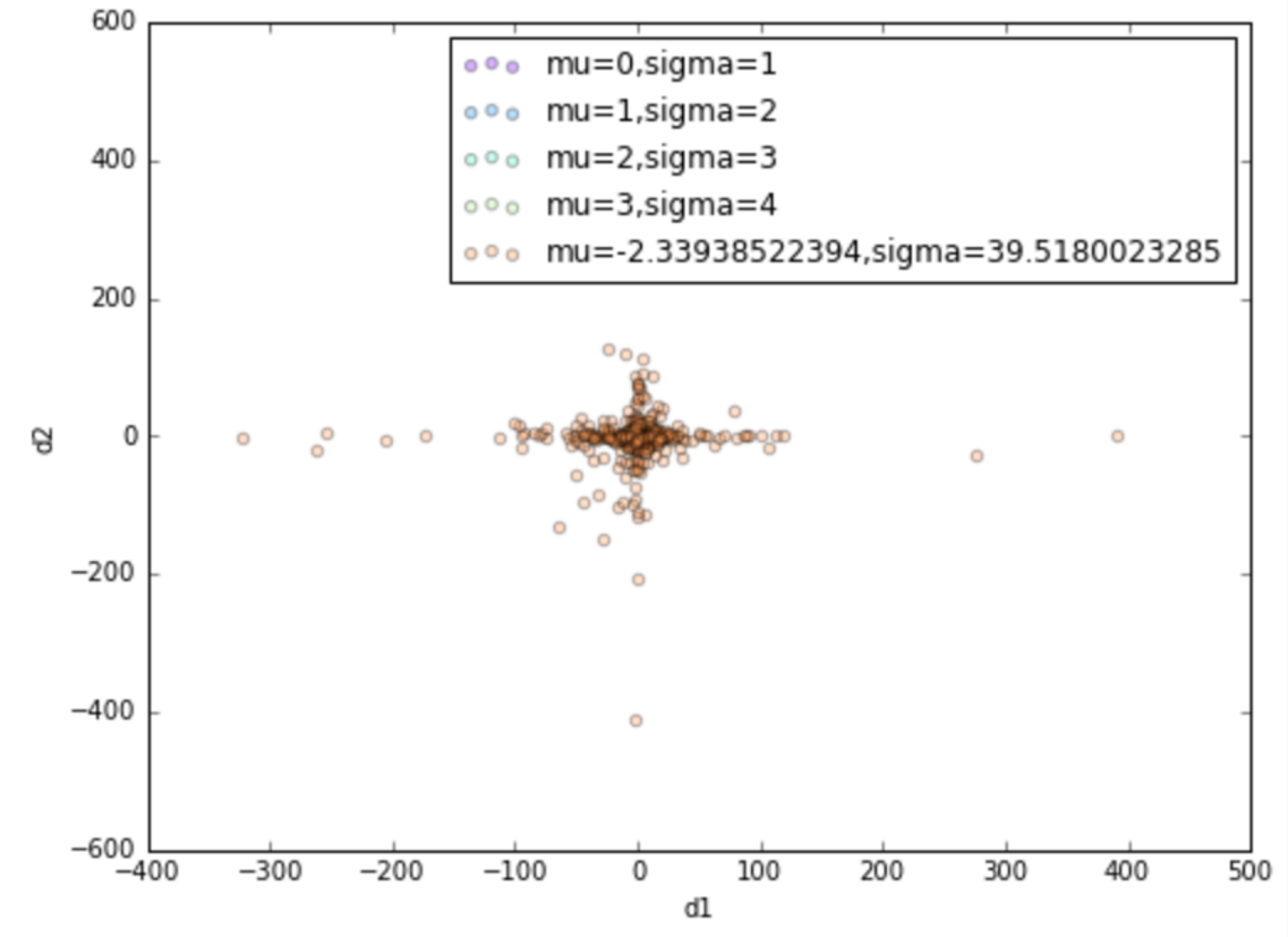
Purpose of Structured Probabilistic Models

- Ex) $p(a, b, c, d) = p(a)p(b|a)p(c|a, b)p(d|b)p(e|c)$.
- Normalize the distribution to sum to 1. z = normalizing constant. $p(x) = \frac{1}{z} \prod_i \phi^{(i)}(C^{(i)})$ where ϕ is a chosen function.

Purpose of Structured Probabilistic Models

ML@B

- unnormalized



- Uses: density estimation, denoising, Missing value imputation, sampling.
- Cons: memory, statistical efficiency, runtime (cost of inference), runtime (cost of sampling).
- A distribution has few parameters if the random variable (node) has few parents in the graph.

Monte Carlo Methods

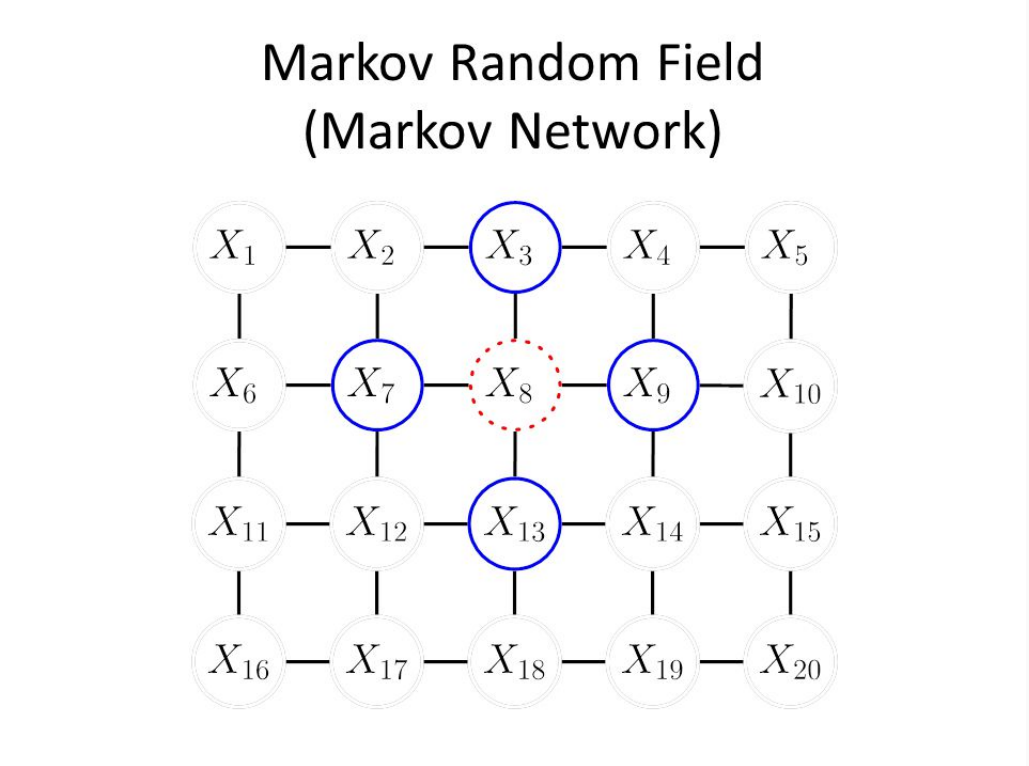
ML@B

- Randomized algorithms that return answers with a random amount of error. Increasing resources (ex: time, memory) can decrease error.
- Used in sampling for approximations, speedup.

Algorithms

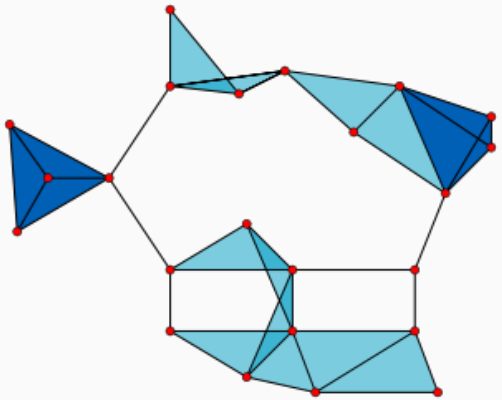
Undirected Models

- Markov random fields (MRFs) or Markov Networks: For when there is a bidirectional relationship (or lack of unidirectional relationship) between nodes.
- Undirected notation includes: -; $<->$; a $->$ b, b $->$ a.



Undirected Models

- An unnormalized probability distribution can be represented by each clique $\mathbf{C}$ in graph $\mathbf{G}$. Let clique potential be a factor $\phi(\mathbf{C})$. Clique potential measures the likelihood of each random variable's state for every configuration (joint state) in the clique.
- $\tilde{p}(x) = \prod_{\mathbf{C} \in \mathbf{G}} \phi(\mathbf{C})$.
- Gibbs distribution: $p(x) = \frac{1}{Z}\tilde{p}(x)$. $Z = \int \tilde{p}(x)dx$



Approximating Z

ML@B

- It is possible to specify variables st Z doesn't exist. This happens if $Z = \int \tilde{p}(x)dx$ diverges.
- Conditions for divergence:

  1. Is an improper integral: a definite integral that has either or both limits infinite or an integrand that approaches infinity at one or more points in the range of integration.
  2. The limit does not exist or it is infinite.

Approximating Z

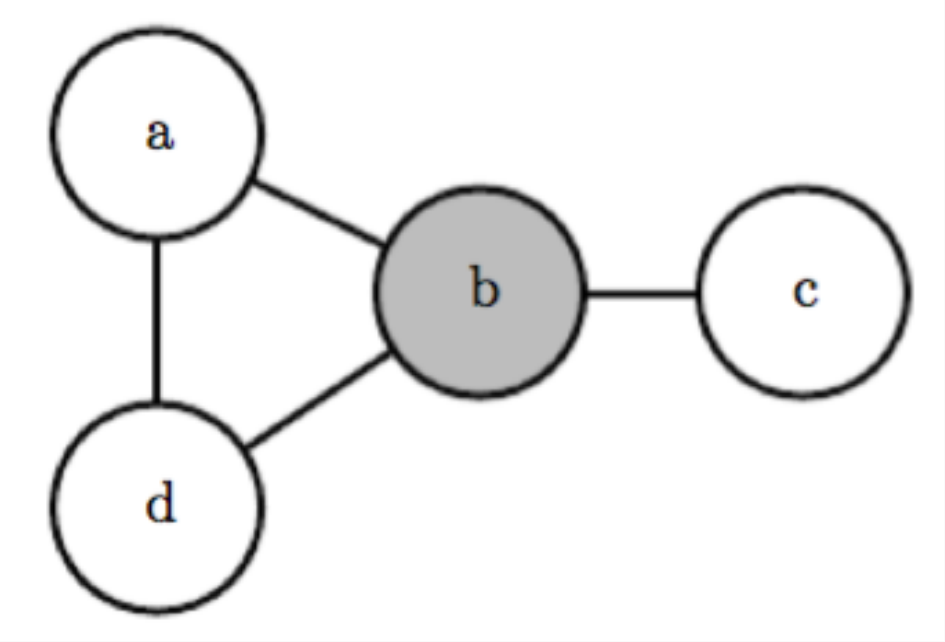
- The domain of each variable can significantly change the type of probability distribution in a clique.
- Ex)
  n-dimensional rv x.
  one clique for each element $x_i \in x \forall i = 1, ..., n$, where each element is mutually independent.
  Domain 1: $x \in \{0, 1\}^n$. $p(x) = \sum_{i=1}^{2n} p(x = pattern_i)$.
  Domain 2: set of n dimensional basis bectors.
  $p(x) = \sum_{i=1}^{n} p(x_i = 1)$.

Energy-Based Models

ML@B

- Assume $\forall x, \tilde{p}(x) > 0$. $\tilde{p}(x) = exp(-E(x))$ where E(x) is the energy function.
  Each clique has an $\phi(x)$ function,
  $p(x) = \prod exp(-E(x)) = \prod \phi(-E(x))$
- By $exp(a)exp(b) = exp(a + b)$,
  $p(x) = exp(\sum -E(x)) = \phi(\sum -E(x))$.
- Also called a product of experts.
- $\mathbf{F}(x) = -log \sum_h exp(-E(x, h))$.

Separation and D-Separation

- dependence-separation
- 

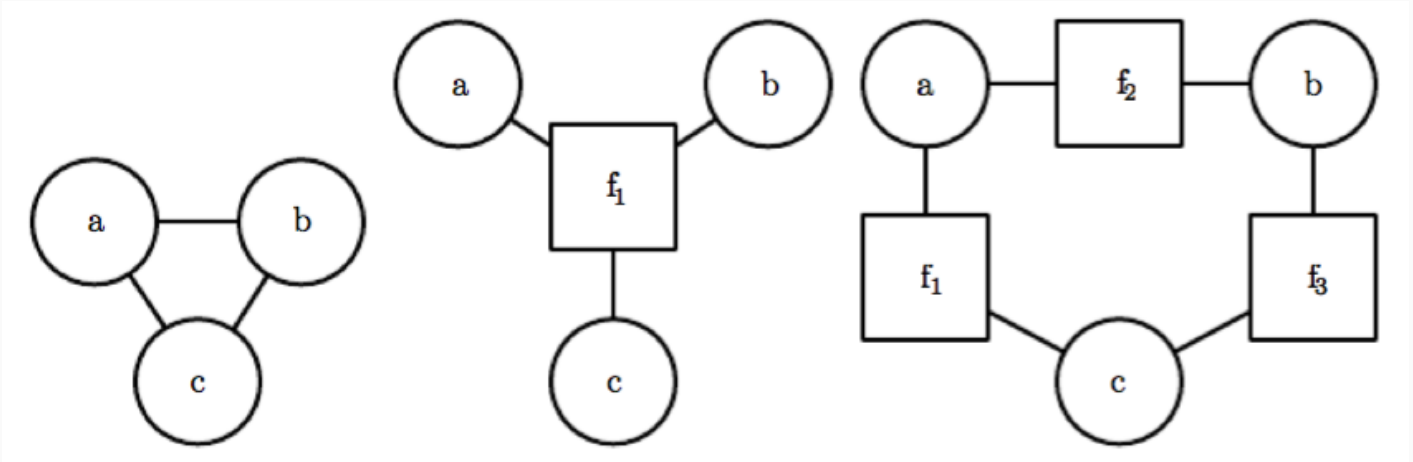
- Given B, A and C are separate (implied-not guaranteed independent). A and D are still connected (implied dependent).
- Choosing graph structure (ex: directed or undirected) depends on the problem.

Factor Graphs

ML@B

- A graphical representation of an undirected model that consists of a bipartite undirected graph.
- Clarifies the undirected graph.
- Rules:
  May be connected with undirected edges.
  A variable and a factor are connected iff the variable is an argument to the factor in the unnormalized probability distribution.
  A factor cannot be connected to any factor.
  A variable cannot be connected to a variable.

Factor Graphs

- 

- Left: undirected graph. Middle and Right: different configurations of the same undirected graph.

Sampling from Graphical Models

- Ancestral sampling: Topological sort variables x_i st for all i and j, j ¿ i if x_i is a parent of x_j. Then sample in order.
- Ex: Let $Pa_G(x_i)$ be the set of random variables (already sampled) that are the ancestors of random variable x_i.

$$\begin{bmatrix} x_1 \sim P(x_1) \\ x_2 \sim P(x_2|Pa_G(x_2) \\ \dots \\ x_n \sim P(x_n|Pa_G(x_i)) \end{bmatrix}$$

Sampling from Graphical Models

ML@B

- Cons: Only for directed graphs, does not support every conditional sampling operation.
- Gibbs sampling: for undirected graphs. Let x be an n-dimensional vector where each x_i is a node in the graph. Iteratively visit each x_i and sample with condition: $p(x_i|x_{i-1})$.

Pros of Structured Modeling

ML@B

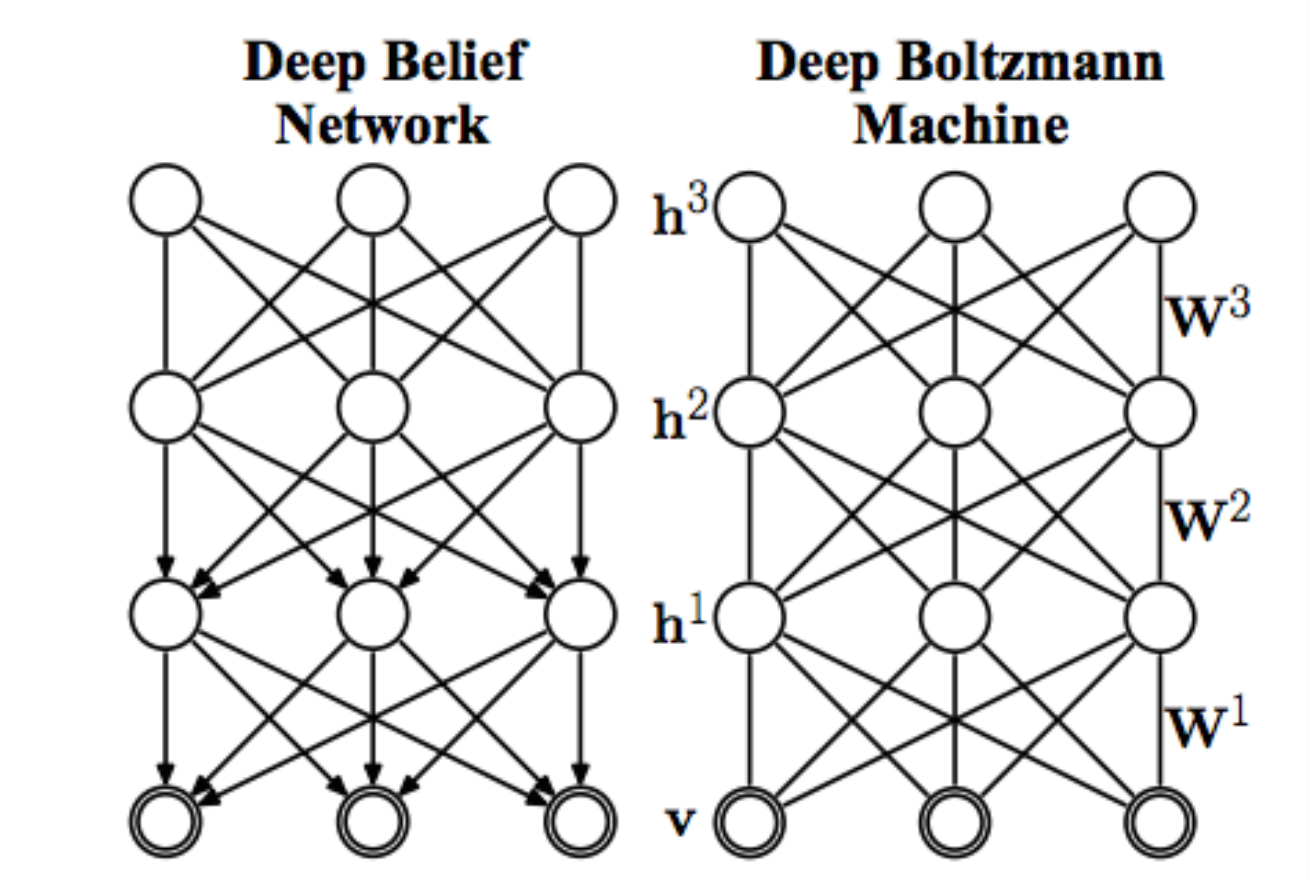
- Reduce cost of learning, representing, and inference from probability distributions.
- Explicit representation of learning and inference $\rightarrow$ Easier to develop and debug.

Deep Learning Approach to Structured Probabilistic Models

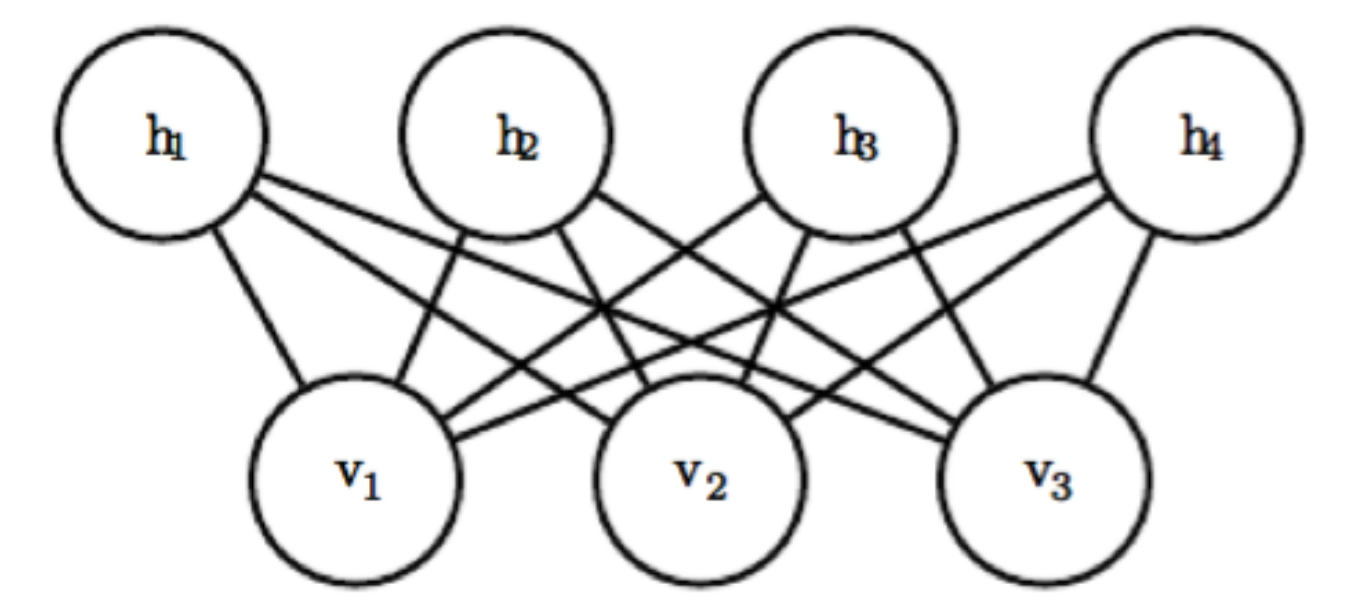
- Let latent variable h_i be at depth j if the shortest path from h_i to an observed variable is j steps. Depth of a model is now max j.
- Use distributed representations. More latent variables.
- Latent variables are learned aren't assigned specific semantics ahead of time. Many latent variables, so algorithms should be efficient.
- Many connections between units.

Restricted Boltzmann Machine

- Energy-based model with binary units. Has a single layer of latent variables to learn a representation for the input.
- Energy function: $E(v, h) = -b^T v - c^T h - v^T W h$ where b, c, W are unconstrained, real, and learnable.



Restricted Boltzmann Machine

- Properties: $p(h|v) = \prod_i p(h_i|v)$, $p(v|h) = \prod_i p(v_i|h)$.
- Consequently, $P(h_i = 1|v) = \sigma(v^T W_{:,i} + b_i)$, $P(h_i = 0|v) = 1 - \sigma(v^T W_{:,i} + b_i)$ where σ is a probability distribution function.
- 

- $E_{h \sim p(h|v)}[h]$ is the set of features for v.

Monte Carlo Sampling

- To approximate a sum or integral, view it as an expectation under a distribution. Approximate the expectation by a corresponding average.
- $$s = \sum_x p(x) f(x) = E_p[f(x)]$$

  ,

  $$s = \int p(x) f(x) dx = E_p[f(x)]$$

  where s is what we're estimating.

Monte Carlo Sampling

- Draw n samples $x^(1), ..., x^(n)$ from p, then calculate the average.

$$\widehat{s}_n = \frac{1}{n} \sum_{i=1}^{n} f(x^{(i)})$$

- This is a reasonable approximation because:
  1. $\widehat{s}_n$ is unbiased. $E[\widehat{s}_n] = \frac{1}{n} \sum_{i=1}^{n} E[f(x^{(i)})] = \frac{1}{n} \sum_{i=1}^{n} s = s$.
  2. By Law of Large Numbers: $\widehat{s}_n$ will converge to s with $n \rightarrow$ inf samples given that the samples are iid and the variance of each $x^{(i)}$ is bounded.

Importance Sampling

- When we can't sample from p(x). Decide which part of integrand is the distribution p(x) and which part is the value f(x). We calculate expectation as:
- $$p(x)f(x) = q(x)\frac{p(x)f(x)}{q(x)}$$
- where we sample from p(x) and average $\frac{pf}{q}$.

Importance Sampling

- Transform the Monte Carlo estimator $\widehat{s}_p = \frac{1}{n}\sum_{i=1,x^{(i)}\sim p} f(x^{(i)})$ into $\widehat{s}_q = \frac{1}{n}\sum_{i=1,x^{(i)}\sim q} \frac{p(x^{(i)})f(X^{(i)})}{q(x^{(i)})}$.
- $E_q[\widehat{s}_q] = E_q[\widehat{s}_p] = s$.
- $Var[\widehat{s}_q] = Var[\frac{p(x)f(x)}{q(x)}]/n$. Minimum variance at: $q^*(x) = \frac{p(x)|f(x)|}{Z}$ where Z is the normalization constant.

Monte Carlo: Biased importance sampling

- Does not require normalized p or q.
- $\widehat{s}_{BIS} = \frac{\sum_{i=1}^{n} \frac{p(x^{(i)})}{q(x^{(i)})}(x^{(i)})}{\sum_{i=1}^{n} \frac{p(x^{(i)})}{q(x^{(i)})}} = \frac{\sum_{i=1}^{n} \frac{\widehat{p}(x^{(i)})}{\widehat{q}(x^{(i)})} f(x^{(i)})}{\sum_{i=1}^{n} \frac{\widehat{p}(x^{(i)})}{\widehat{q}(x^{(i)})}}$ where $\widehat{p}, \widehat{q}$ are unnormalized p, q respectively. $x^{(i)}$ are the samples from q.
- Choice of the q distribution affects efficiency.

Monte Carlo: Biased importance sampling

ML@B

- With high-dimensional x:
  1. q will not match to p or $p|f|$ (collect useless samples). $q(x^{(i)}) \gg p(x^{(i)})|f(x^{(i)})$.
  2. q will "match" to p or $p|f|$ too well (extreme overestimation). $q(x^{(i)}) \ll p(x^{(i)})|f(x^{(i)})$.

Markov Chain Monte Carlo Methods (MCMC)

ML@B

- Use Markov Chains to estimate with Monte Carlo.
- Example model: energy-based model (EBM) $p(x) \propto exp(-E(x))$. Drawing samples from EBM is difficult because it can be an undirected graph ($b \rightarrow a, a \rightarrow b$).
- Soln: Use a Markov chain: random state x and a transition distribution $T(x'|x)$ which is the probability that x will go to state x'.

Markov Chain Monte Carlo Methods (MCMC)

ML@B

- Have infinitely many MC run in parallel. Let t = time steps. The states of each MC is from distribution $q^{(t)}(x)$. $q^{(0)}$ is the initial distribution. $q^{(t)}$ inferred from all MC steps so far, so $q^{(t)}$ to converge to p(x).

Markov Chain Monte Carlo Methods (MCMC)

ML@B

- Ex: Positive integer rv x, probability distribution q, vector v. $q(x = i) = v_i$. Update to state x'. Probability of landing on state x' is $q^{(t+1)}(x') = \sum_x q^{(t)}(x)T(x'|x)$. Define matrix A st it describes the probability of state x to every state in the graph. $A_{i,j} = T(x' = i|x = j)$.
- Update rule for all MC run in parallel: $v^{(t)} = Av^{(t-1)} = A^t v^{(0)}$.
- A is a stochastic matrix.

Markov Chain Monte Carlo Methods (MCMC)

ML@B

- A Markov chain with transition operator T will converge under mild conditions to the point: $q'(x') = E_{x \sim q} T(x'|x)$.
- Deep Learning uses multiple MCs.
- Cons: MCs are expensive because of the time required to reach equilibrium (converge). We don't know how long it takes to converge.

Gibbs Sampling

ML@B

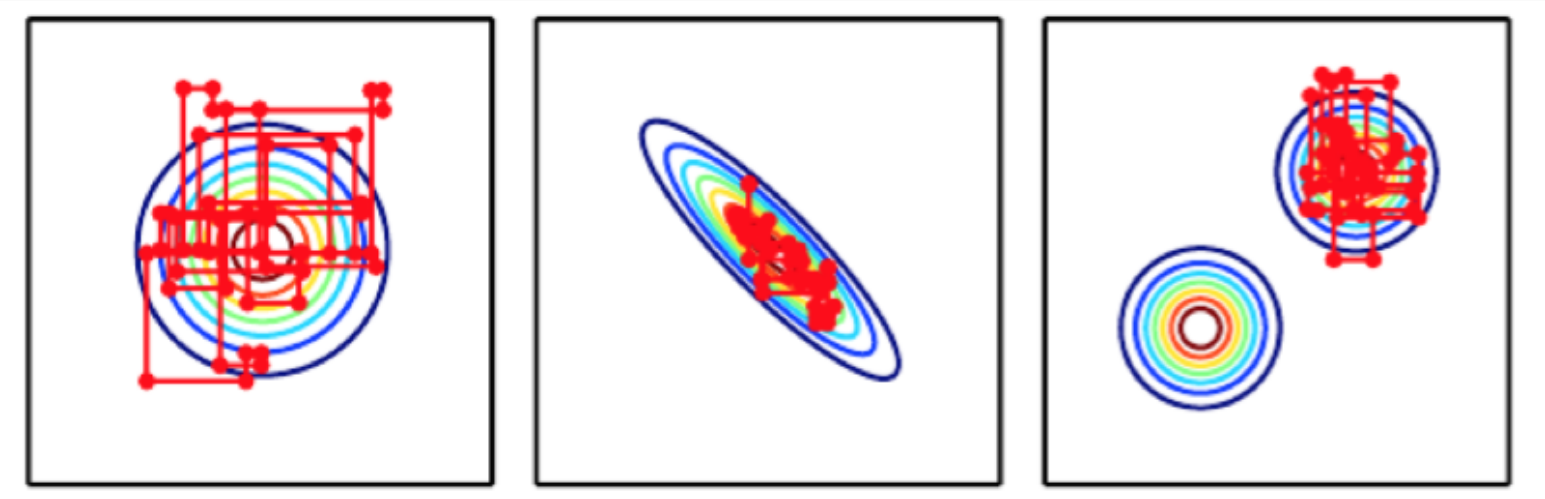
- Ensures that q(x) is a useful distribution by deriving T(x' — x) from the learned $p_{model}(x)$.
- Block Gibbs Sampling: Pick one variable x_i and sample it from p_{model} conditioned on its neighbors in undirected graph G. G defines the structure of the EBM p_{model} - in this example.

Challenge of Mixing between Separated Modes

- Ideally, samples from an MC to sample from p(x) would be iid. MCMC samples become very correlated in high dimensions.
- In terms of an EBM, T = energy barrier from x to x. If probability x to x' low $\rightarrow$ high energy barrier. So when multiple modes with high probability are separated by nodes of low probability, then converge slowly.

Challenge of Mixing between Separated Modes

- Ex:



- Left: Multivariate normal, 2 independent variables.
- Center: Multivariate normal, 2 highly correlated variables.
- Right: Mixture of Gaussians with very far modes that are not axis aligned.

Tempering to Mix between Modes

ML@B

- Based on making a smoother version of the target distribution.
- Ex: EBM with a smoothed distribution: $p_\beta(x) \propto exp(-\beta E(x))$ where E is the energy function, and β is the parameter to tune smoothness. Temper with $\beta < 1$.
- Parallel Tempering: MC simulates different states in parallel at different parameters.
- Caveat: Sometimes T has to be a small probability in order to converge.

Depth May Help Mixing

ML@B

- Let latent variable model p(h, x). Mixing will be poor if $p(h|x)$ encodes x too well.
- A soln: Let h be a deep representation such that $p(h|x)$ encodes x less strict.
- Ex: Autoencoders and RBMs usually have a marginal distribution over h that is more uniform and unimodal than x's distribution.

Monte Carlo Pi Example

- Animation

  `https://www.youtube.com/watch?v=VJTFfIqO4TU&t=25s`

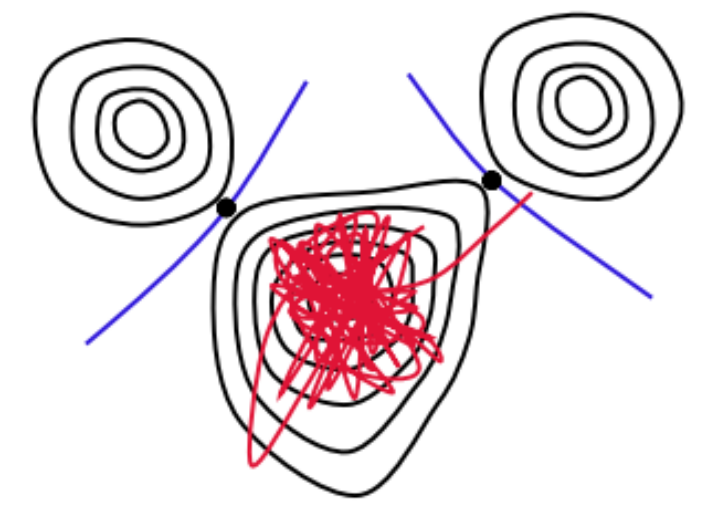
Paper: Kinetic Monte Carlo Method

ML@B

- Simulating atom movement. Total simulation time is usually less than one microsecond, which may not be useful for observing longer processes such as diffusion.
- Kinetic Monte Carlo tries to solve this time-scale problem by modeling state to state transitions in a Markov Model.

Kinetic Monte Carlo Method - Funny Contour Diagram

- Energy-Barrier-Limited Infrequent-Event System



Contour plot of the potential energy between states.

- Memoryless System: The transition probabilities (escape rate) for exiting state i have nothing to do with the history prior to entering state i.

Kinetic Monte Carlo Method - KMC Procedure

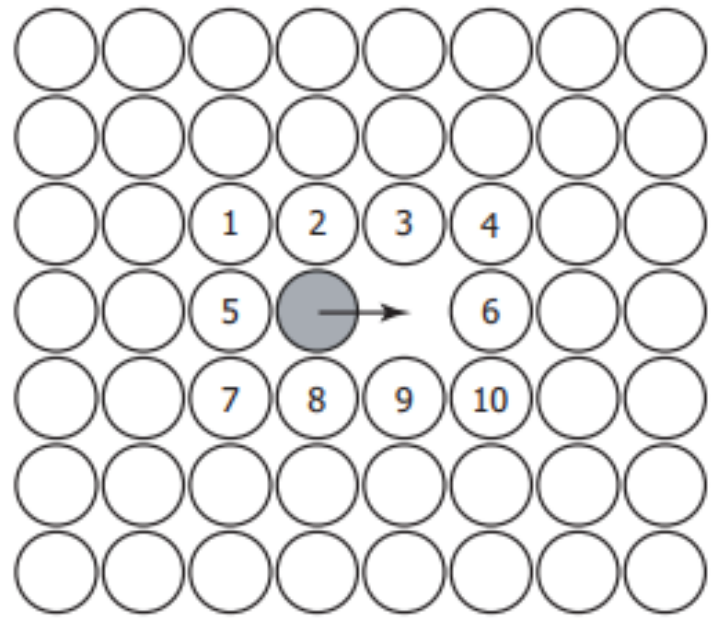
- In state i
- Set of pathways and associated rate constants kij
- We have the distribution for each pathway's first escape time.
- Exponentially distributed time t_j to transition.
- Find the pathway j_{min} with the lowest t_j, record only this step, advance the overall system clock by j_{min}.

Kinetic Monte Carlo Method - KMC Procedure (common)

- BKL or "n-fold way" algorithm:
- Have an object with length equal to rate constant k_{ij} for pathway i to j for all M escape pathways.
- Put all objects end to end, with total length k_{tot}.
- Choose 1 random position along this length, picking the pathway for the system to follow.
- Advance the clock by drawing a time from an exponential distribution with rate constant k_{tot}.

Kinetic Monte Carlo Method - Harmonic Transition State Theory

- Transition pathway has a saddle point on the potential energy surface (contours of the distribution of atoms leaving a state).
- Rate Catalog:



- Caveat: The real dynamics of a system can have unexpected and complex reaction pathways that cannot occur during a KMC simulation.

Kinetic Monte Carlo Method - Problems

- A low barrier can significantly decrease simulation time.
- The system repeatedly visits a subset of states.
- Enumerating all states is computationally expensive.

Kinetic Monte Carlo Method - Optimizing

- Object kinetic Monte Carlo
  -Higher level simulation.
  -Create state definitions and rate constants for atom clusters.
  -Pro: Can run for more time.
  -Con: Can miss important pathways with less detail. Similarly, relationships between states for clusters can become much more complex because of the missing (simple) states for atoms.
- F U T U R EEEEEE: On-the-fly Kinetic Monte Carlo

Paper: Monte Carlo Strength Evaluation

ML@B

- Simulate guessing attacks. Goal is to increase effort it takes for attackers to break a password.
- Sampling from a model to estimate password strength.

Monte Carlo Strength Evaluation - Attacks

ML@B

- Password Guessing
  -Dictionary, rainbow chains (efficiently memorize very large set of pre-computed password hashes)
  -Probabilistic attacks reduce number of password guesses. Ex: n-gram, PCFG, backoff models.
- Some Defenses: Salting, password strengthening, password stretching (hashing passwords using computationally expensive functions)

Monte Carlo Strength Evaluation - Compute Password Strength

- Problem: Strength meters generally based on heuristics that don't accurately reflect a password's resistance to guessing attacks.

Monte Carlo Strength Evaluation - Attack Model 1

- N-gram model: probability of a password $c_1 \ldots c_l$ is

$$p_{n-gram}(c_1 \ldots c_l) = \prod_{i=1}^{l+1} P(c_i | c_{i-n+1} \ldots c_{i-1})$$

Algorithm 1 Password generation for n-grams.

```
def starting_state():
    return "⊥ ... ⊥" with length n − 1
def update_state(s, t):
    drop the first character from s
    return s + t # concatenation
s ←starting_state()
g ←"" # accumulator for the result
while True:
    r ←random number in [0, 1]
    # find i through binary search
    i ←rightmost index s.t. C_s[i] > r
    if t_{s,i} = ⊥: return g
    append t_{s,i} to g
s ←update_state(s, t_{s,i})
```

Monte Carlo Strength Evaluation - Attack Model 2

ML@B

- Probabilistic Context-Free Grammars (PCFGs): passwords grouped by templates.
- Ex: Make a probabilistic model out of template: L_3D_3 (3 letters followed by 3 digits).
  The probability of "abc123" is:
  $P_{PCFG}(\text{"}abc123\text{"}) = P(L_3D_3)P(\text{"}abc\text{"}|L_3)P(\text{"}123\text{"}|D_3)$

Monte Carlo Strength Evaluation - Attack Model 3

- Backoff Model: By Katz. Addresses sparsity by considering n-grams of all lengths, and discarding those with less occurrences than a threshold τ .

Algorithm 2 Sample creation for the backoff model.

```
def starting_state():
    if using the start symbol:  return "⊥"
    else: return ""
def update_state(s, t):
    append t to s
    while o(s) < τ:
        drop the first character in s
Run Algorithm 1 using these functions.
```

Monte Carlo Strength Evaluation - Attack Model 3

- Given a threshold τ, the probability of a single character $\hat{c}$ is its frequency in the training set:

$$p_{b0}(\hat{c}) = \frac{o(\hat{c})}{\sum_c o(c)}$$

where $o(c_1 \ldots c_n)$ is the number of occurrences of the $c_1 \ldots c_n$ string in the training set.

$$p_{bo}(c_1 \ldots c_{n+1}) = p_{b0}(c_1 \ldots c_n) P(c_{n+1} | c_1 \ldots c_n)$$

where

$$P(c|c_1 \ldots c_n) = \begin{cases} \frac{o(c_1 \ldots c_n c)}{o(c_1 \ldots c_n)} if\, o(c_1 \ldots c_n c) \geqslant \tau \\ P(c|c_2 \ldots c_n)\tau(c_1 \ldots c_n) otherwise \end{cases}$$

and

$$r(c_1 \ldots c_n) = \sum_{c: o(c_1 \ldots c_n c) \geqslant \tau} \frac{o(c_1 \ldots c_n c)}{o(c_1 \ldots c_n)}$$

Supplemental Reading: Latent Dirichlet Allocation

- Probabilistic Topic Models by David M. Blei
  http://www.cs.columbia.edu/∼blei/papers/Blei2012.pdf

Sources

- Topic Reference:
  https://github.com/HFTrader/DeepLearningBook/blob/master/Dee
- Paper 1:
  http://www.phys.ubbcluj.ro/ zneda/edu/mc/mc4.pdf
- Paper 2:
  http://www.eurecom.fr/en/publication/4711/download/rs-publi-4711.pdf
- Paper 3:
  http://www.cs.columbia.edu/∼blei/papers/Blei2012.pdf

Questions

Questions?!