Undirected Probabilistic Graphical Models

CS 760: Machine Learning

Mark Craven and David Page

Spring 2019

Goals for the lecture

you should understand the following concepts

- Markov networks
- Markov network syntax
- Markov network semantics
- Differences between Bayes nets and Markov nets
- Potential functions
- Partition function
- Loglinear formulation of MNs
- MN parameter learning by gradient ascent
- Markov chain Monte Carlo (MCMC) sampling
- Metropolis-Hastings
- Gibbs sampling
- Persistent contrastive divergence (PCD)
- Pseudo-likelihood
- Screening rules
- Graphical Lasso

Markov Networks

- Like Bayes Nets
  - Graphical model that describes joint probability distribution using tables (AKA potentials)
  - Nodes are random variables
  - Labels are outcomes over the variables

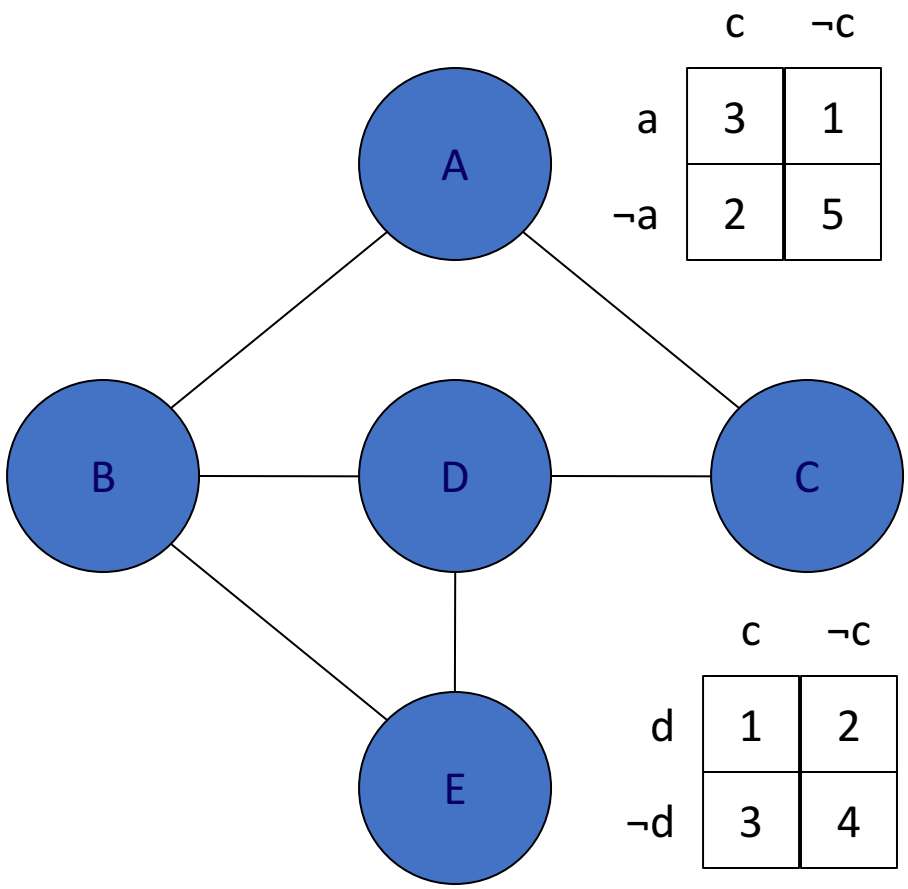
Markov Networks

- Like Bayes Nets
  - Graphical model that describes joint probability distribution using tables (AKA potentials)
  - Nodes are random variables
  - Labels are outcomes over the variables

- Unlike Bayes Nets
  - Undirected graph
  - No requirement that tables are (conditional) distributions
  - Therefore must normalize when computing probability of a setting

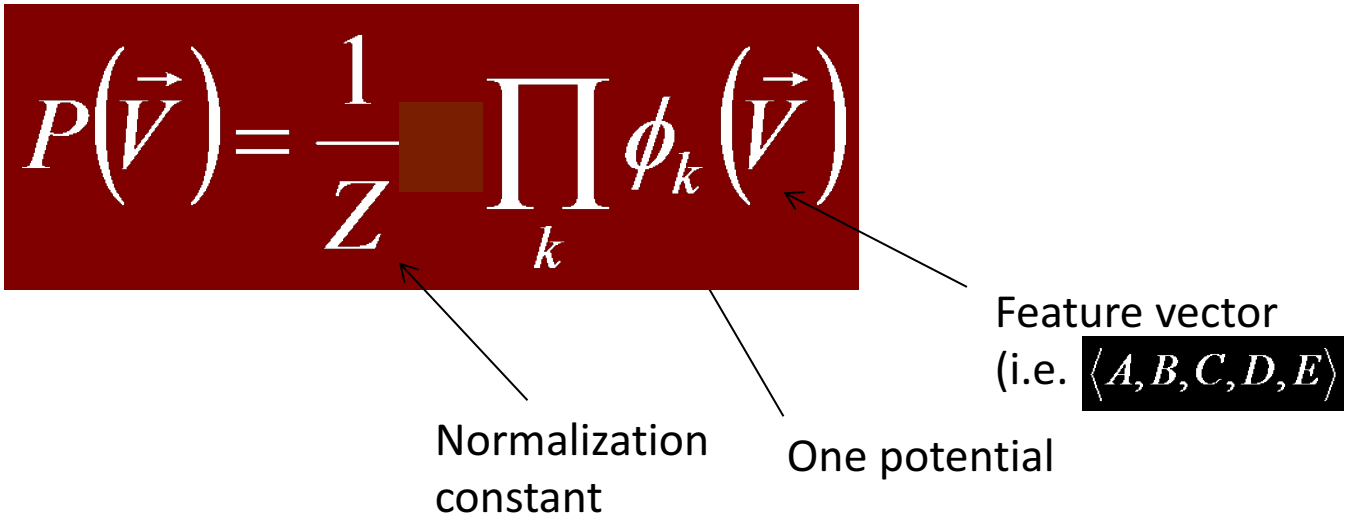
More on Potentials

- Values are typically non-negative
- Values need not be probabilities
- Generally, one table associated with each clique



Calculating the Full Joint Probability Density

- Full Joint Probability Density is the normalized product of the event probabilities



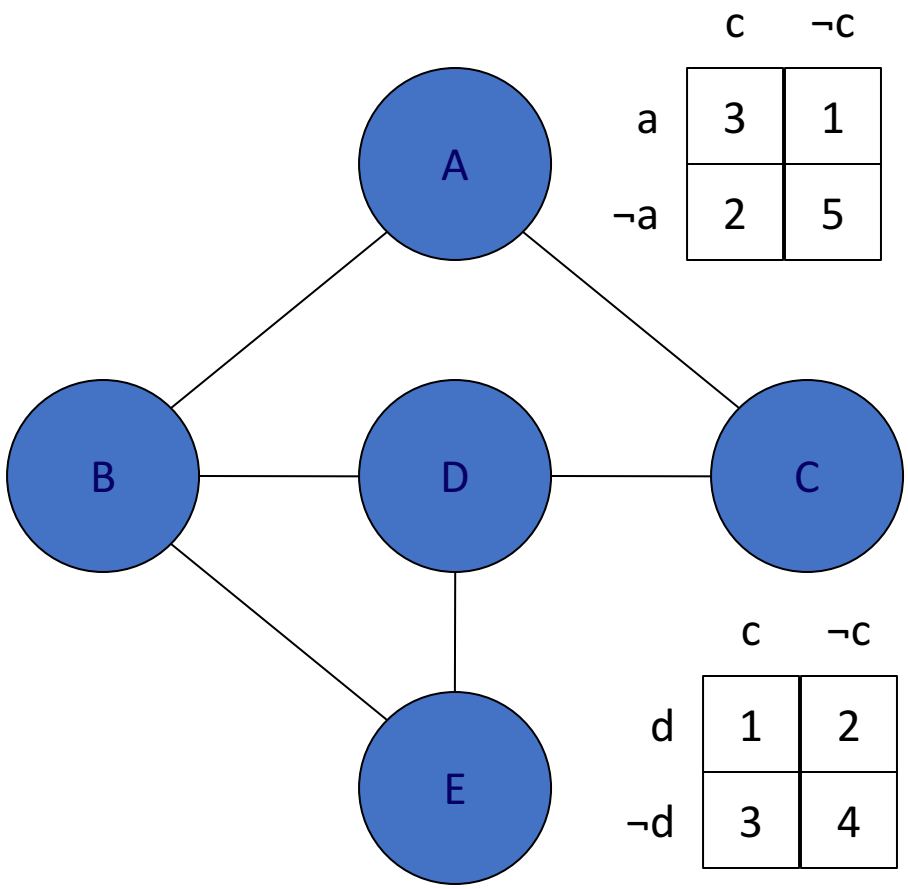
Calculating the Normalization Constant Z

$$Z = \sum_{\vec{v} \in \vec{V}} \prod_{k} \phi_k(\vec{v})$$

Using $$P(\vec{V}) = \frac{1}{Z} \prod_k \phi_k(\vec{V})$$

- Get probability of A=1, B=0, C=1, D=0, E=0
- Only need potentials
- Multiply entries consistent with this setting

  (3 x 3 = 9)



Hammersley-Clifford Theorem

- If Distribution is strictly positive ($P(x) > 0$)
- And Graph encodes conditional independences
- Then Distribution is product of potentials over cliques of graph

- Inverse is also true

- ("Markov network = Gibbs distribution")

Markov Nets versus Bayes Nets

- Advantages of Markov Nets
  - Easier to reason about conditional independence
  - Markov Blanket is just set of neighbors
  - d-separation: conditional independence achieved iff all paths cut off by evidence
  - No need to select an arbitrary, potentially misleading direction for a dependency in cases where the direction is unclear
  - Learn structure just by learning parameters

Markov Nets versus Bayes Nets

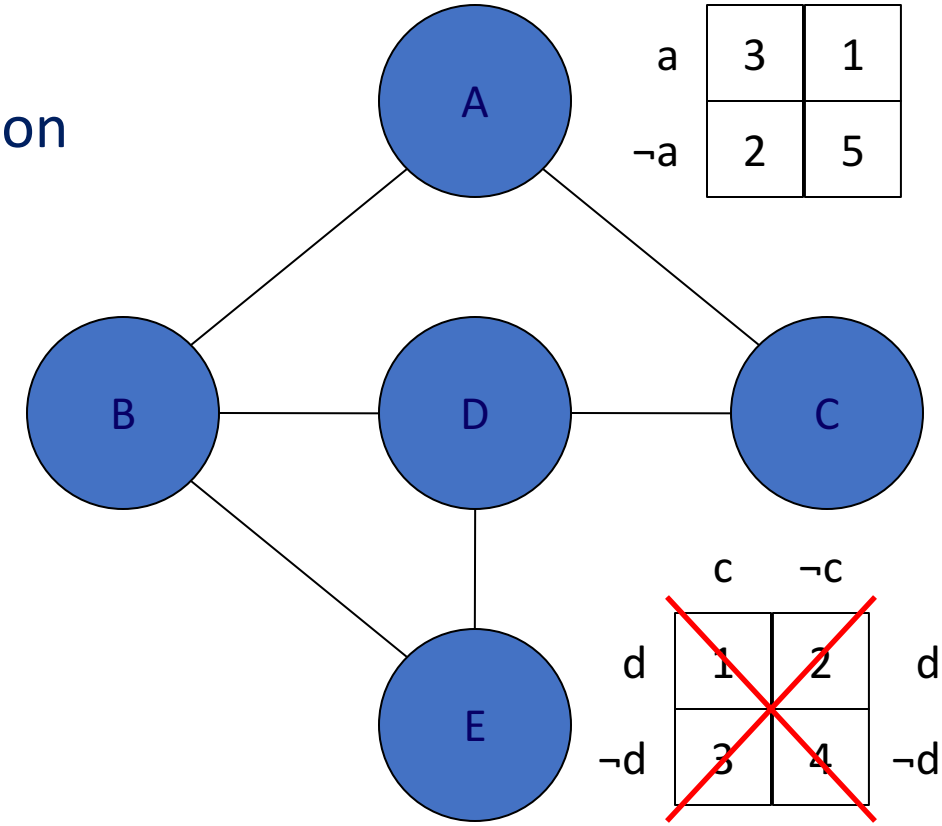
- Disadvantages of Markov Nets
  - Computationally intensive to compute probability of any complete setting of variables with Markov Net (NP-hard), easy for Bayes Net
  - Hard to learn Markov Net parameters in a straightforward way
  - Can’t just use marginal frequencies from data as for Bayes nets
  - Gradient ascent requires inference (hard)

Hand-Constructing Markov Nets

- Just as in Bayes Nets, the decision of which tables to represent is based on background knowledge
- Although the model can be built from the data, it is often easier for people to leverage domain knowledge
- Although the model is undirected, it can still be helpful to think of directionality when constructing the Markov Net

Scale Invariance

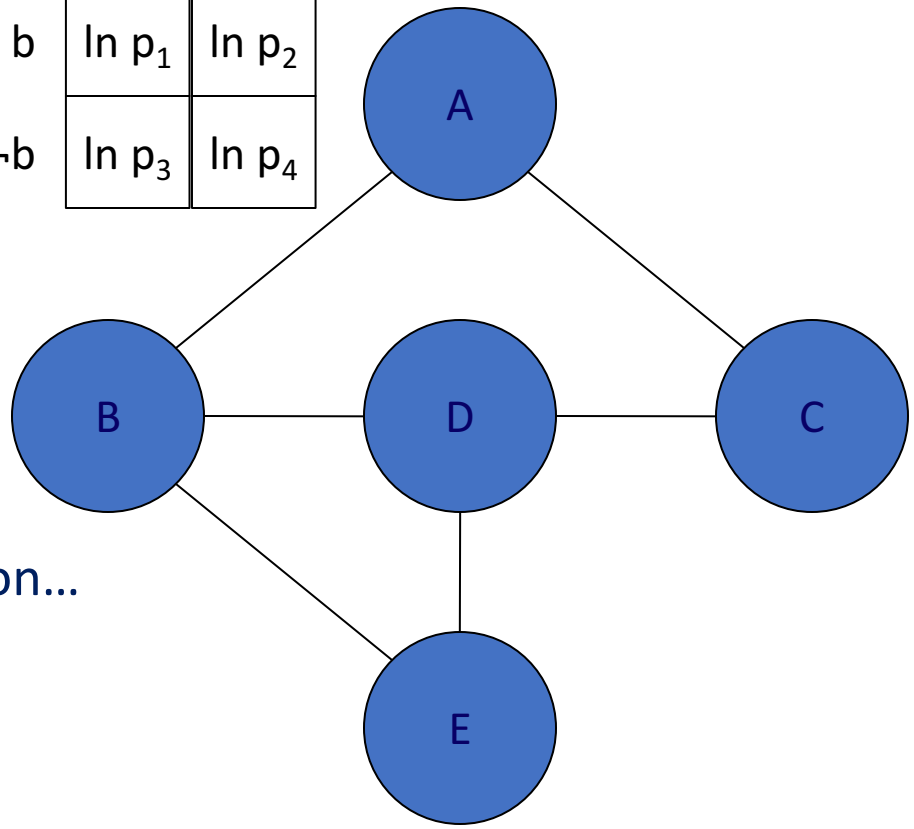
The change at the right will not effect the joint probability distribution



Log Linear Models

- Equivalent to Markov Nets (though they look very different)
- Take the natural log of each parameter:

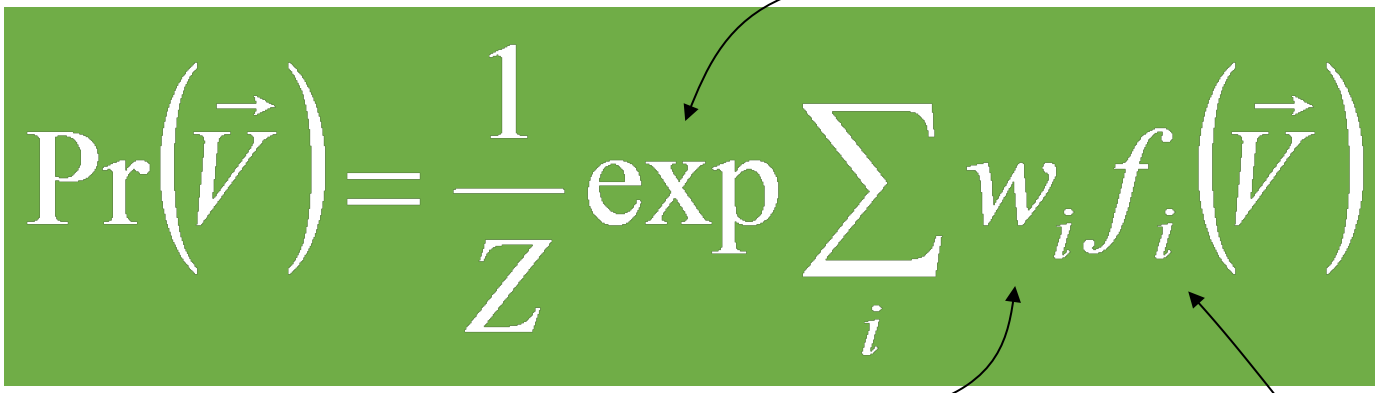
	a	¬a
b	$\ln p_1$	$\ln p_2$
¬b	$\ln p_3$	$\ln p_4$



- Not scale-invariant to this change
- So must change definition of distribution…

Log Linear Models

- This change allows us to write the joint probability distribution or density as:



Inference

- Almost the same as in Bayes Nets (this is somewhat surprising considering all the differences!)

- Possible approaches:
  - Gibbs sampling
  - Variable elimination
  - Belief propagation

Inference in Markov Networks

- Goal: Compute marginals & conditionals of

$$P(X) = \frac{1}{Z}\exp\left(\sum_i w_i f_i(X)\right) \qquad Z = \sum_X \exp\left(\sum_i w_i f_i(X)\right)$$

- Conditioning on Markov blanket of a proposition x is easy, because you only have to consider cliques (features) that involve x

- Gibbs sampling exploits this

$$P(x \mid MB(x)) = \frac{\exp\left(\sum_i w_i f_i(x)\right)}{\exp\left(\sum_i w_i f_i(x=0)\right) + \exp\left(\sum_i w_i f_i(x=1)\right)}$$

Markov Chain Monte Carlo (MCMC)

- General algorithm: Metropolis-Hastings
  - Sample next state given current one according to transition probability
  - Reject new state with some probability to maintain detailed balance

- Simplest (and most popular) algorithm: Gibbs sampling
  - Sample one variable at a time given the rest
  - Requires that no settings have probability 0

$$P(x \mid MB(x)) = \frac{\exp\left(\sum_i w_i f_i(x)\right)}{\exp\left(\sum_i w_i f_i(x=0)\right) + \exp\left(\sum_i w_i f_i(x=1)\right)}$$

Learning in MNs: Recall the Bayes Net approach

- In Bayes Nets, we go through each variable one at a time, row by row in the CPT adjusting weights

- One way to think of this approach is that we look at the prior setting and ask the likelihood of this setting based on what we see in the data, then adjust the CPT to be consistent with the data

Can we use this approach on Markov Nets?

- **No!** Consider changing a single table value
- This changes the partition function, Z
- Thus, a local change to one table effects other tables; local changes have global effects!
- Partition function induces a dependency among parameters that we didn't have in Bayes nets
- Must perform gradient descent instead

Markov Net Learning

- We want to get the derivative of the likelihood function. We can then incrementally move each weight in direction of the gradient based on a learning parameter η

- The above approach amounts to differencing the expectation of priors and observed occurrences

Analyzing

$$\Pr(\vec{V}) = \frac{1}{Z} \exp \sum_i w_i f_i(\vec{V})$$

- In this formulation, the w's are just weights and the f's are just features
- As such, we can throw the graph out if we want – we have everything we need in the w_is and f_is
- In this view, parameter learning is just weight learning

Assume: Data set D consists of $d_1, ..., d_m$
Features are $f_1, ..., f_n$
Weights on features are $W = \langle w_1, ..., w_n \rangle$
Let $n_i(D)$ denote the number of times (data points where)
f_i is true in D
With slight abuse of notation, let $E_W(n_i(m))$ denote the expected number of times f_i is true in m examples under the distribution defined by W
Let ll denote the log-likelihood of W given D: $\ln P(D|W)$

$$ll = \sum_j \ln \frac{1}{Z} e^{\sum_i w_i f_i(d_j)}$$

$$ll = \sum_{i,j} w_i f_i(d_j) - \ln Z$$

$$\frac{\partial ll}{\partial w_i} = \sum_j f_i(d_j) - \frac{1}{Z} \sum_V e^{\sum_i w_i f_i(V)} f_i(V)$$

$$\frac{\partial ll}{\partial w_i} = n_i(D) - E_W(n_i(m))$$

Assume: Data set D consists of $d_1, ..., d_m$
Features are $f_1, ..., f_n$
Weights on features are $W = \langle w_1, ..., w_n \rangle$
Let $n_i(D)$ denote the number of times (data points where)
f_i is true in D
With slight abuse of notation, let $E_W(n_i(m))$ denote the expected number of times f_i is true in m examples under the distribution defined by W
Let ll denote the log-likelihood of W given D: $\ln P(D|W)$

$$ll = \sum_j \ln \frac{1}{Z} e^{\sum_i w_i f_i(d_j)}$$

$$ll = \sum_{i,j} w_i f_i(d_j) - \ln Z$$

$$\frac{\partial ll}{\partial w_i} = \sum_j f_i(d_j) - \frac{1}{Z} \sum_V e^{\sum_i w_i f_i(V)} f_i(V)$$

$$\frac{\partial ll}{\partial w_i} = n_i(D) - E_W(n_i(m))$$

Target value

To be changed

Gradient Descent

- In practice, we minimize the negative log likelihood
- Just flip the difference and do gradient descent rather than ascent
- How do we know this will converge to the right answer?
- Negative log likelihood is convex (no local minima), so when we land at a local minimum it's a global minimum
- We won't prove convexity now but here's the definition…

Convexity (from Bubeck, 2015)

Definition 1.1 (Convex sets and convex functions)**.** A set $\mathcal{X} \subset \mathbb{R}^n$ is said to be convex if it contains all of its segments, that is

$$\forall (x, y, \gamma) \in \mathcal{X} \times \mathcal{X} \times [0, 1],\ (1 - \gamma)x + \gamma y \in \mathcal{X}.$$

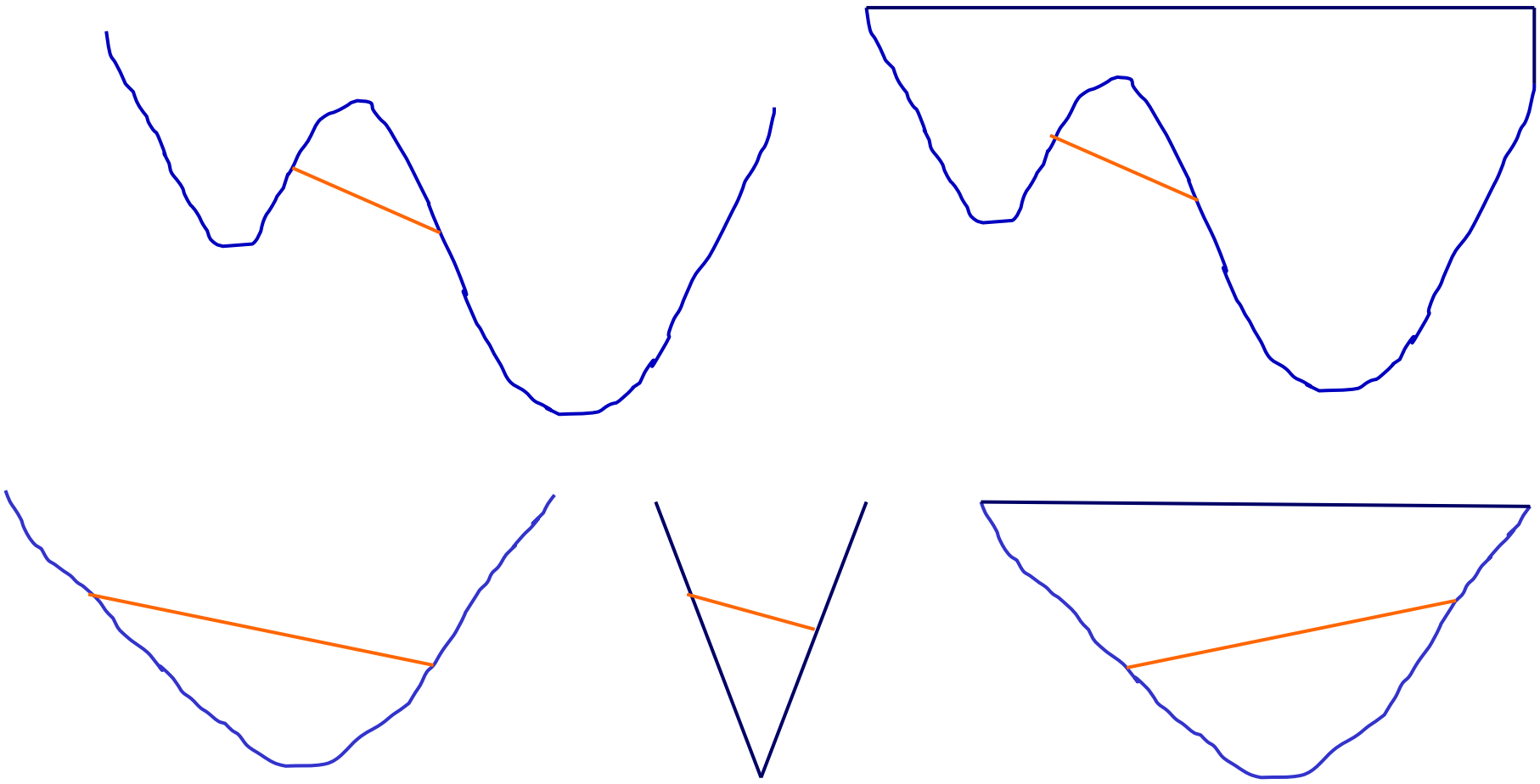
A function $f : \mathcal{X} \to \mathbb{R}$ is said to be convex if it always lies below its chords, that is

$$\forall (x, y, \gamma) \in \mathcal{X} \times \mathcal{X} \times [0, 1],\ f((1 - \gamma)x + \gamma y) \leq (1 - \gamma)f(x) + \gamma f(y).$$

We are interested in algorithms that take as input a convex set $\mathcal{X}$ and a convex function f and output an approximate minimum of f over $\mathcal{X}$. We write compactly the problem of finding the minimum of f over $\mathcal{X}$ as

$$\begin{aligned} &\text{min. } f(x) \\ &\text{s.t. } x \in \mathcal{X}. \end{aligned}$$

(Non-)Convex Function or Set



- Show for all real $a < b$ and $0 \leq c \leq 1$,
  $f(ca + (1-c)b) \leq c\, f(a) + (1-c)\, f(b)$ for

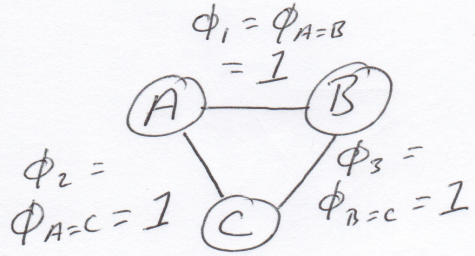
following:

  - $f(x)=|x|$
  - $f(x)=x^2$
  - Not so for $f(x)=x^3$

- In general x could be a vector **x**

- For gradient descent, also want f(x) to be continuous differentiable

Example: Ising Model



$$Z = \sum_{\vec{v}} e^{\sum_i \phi_i(\vec{v})} \approx 56.2$$

Data

A	B	C
1	1	0
0	1	1
1	1	1
0	0	0
1	0	1

$\#_{A=B} = 3$

$\#_{B=C} = 3$

$\#_{A=C} = 3$

A	B	C	$e^{\sum_i \phi_i(\vec{v})}$	$P_{\vec{\phi}}(\vec{v})$
0	0	0	$e^3 \approx 20$	.35
0	0	1	$e \approx 2.7$	.05
0	1	0	.	.
0	1	1	.	.
1	0	0	.	.
1	0	1	.	.
1	1	0	.	.
1	1	1	$e^3 \approx 20$	.35

$E_{\vec{\phi}, 101=5}[\#_{A=B}] = .8(5) = 4$

same for $\#_{A=C}$, $\#_{B=C}$

Gradient: $\langle -1, -1, -1 \rangle$

New $\vec{\phi}$: $\langle 1, 1, 1 \rangle + \eta \langle -1, -1, -1 \rangle$

(e.g., .05) $= \langle .95, .95, .95 \rangle$

For real-world problems, *Z* takes too long to compute

- Combine gradient descent and MCMC inference: *Contrastive Divergence (CD)* and don't wait for MCMC to converge (CD-k)
- Use *pseudo-likelihood* approximation to likelihood: do many logistic regressions (next lecture) to get neighbors of each node; in some situations this gives provably correct structure, e.g. Drugs & Events
- Often include Lasso penalty (later)… *screening rules* can eliminate many features by guaranteeing their weights will be 0 for a given λ
- Can avoid problem altogether if use continuous variables instead of discrete, and use Gaussian graphical model: *Graphical Lasso*

CD-*k*

- Initialize *m*=100 MCMC (Gibbs) chains
- After every *k* steps of Gibbs sampling, take a gradient step in weight space, based on expected feature counts from the last state of each of the *m* chains
- Don't restart Gibbs chains after weight update: continue same chains
- Common to use *k*=1
- Justification:
  - Weight updates are small so…
  - After any weight update the model has changed very little
  - Weight update only needs right gradient direction, not exact expectations
  - Geng, Kuang, et al., UAI 2018, show how to *adaptively* choose k

Comments on Probabilistic Graphical Models

- Much current work on structure learning focuses on binary MRFs, not BNs
  - Start with a clique and learn the weights; 0 weight implies no edge
  - Incorporate a penalty (e.g., lasso) as in BNs to avoid overfitting

- Much current BN work is focused on learning provable *causal* networks
  - E.g., Pearl; Robbins; Cooper, Glymour, Spirtes & Scheines
  - One example: PC Algorithm
  - Causal inference is a big general research area in AI, ML, Statistics

- In practice, may get better causal inferences from temporal data and models
  - Dynamic Bayes nets
  - Continuous-time Bayes nets
  - Point Process models such as Hawkes and others that assume an event's occurrence probability is governed by an intensity function that is influenced by previous events

- Above areas tie into incorporating actions and decision theory
  - Influence Diagrams, MDPs, POMDPs
  - Reinforcement Learning (later)
  - Bandits