Bayes Net Learning

EECS 474 Fall 2016

Homework Remaining

- Homework #3 assigned
- Homework #4 will be about semi-supervised learning and expectation-maximization
- …Homeworks #3-#4: the "how" of Graphical Models
- Then project (more on this soon)

Road Map

- Basics of Probability and Statistical Estimation
- Bayesian Networks
- Markov Networks
- Inference
- Learning
  - Parameters, Structure, EM
- Semi-supervised Learning, HMMs

Today: Learning

- General Rules of Thumb in Learning
- Learning in Graphical Models
  - Parameters in Bayes Nets

What is Learning?

- Given:
  - target domain (set of random variables)
    - E.g., *disease diagnosis: symptoms, test results, diseases*
  - Expert knowledge
    - *MD's opinion on which diseases cause which symptoms*
  - **Training examples** from the domain
    - *Existing patient records*
- Build a **model** that predicts future examples
  - Use expert knowledge & data to learn PGM **structure** and **parameters**

General Rules of Thumb in Learning

- The more training examples, the better
- The more (~correct) assumptions, the better
  - Model structure (e.g., edges in Bayes Net)
  - Feature selection
    - Fewer irrelevant params => better

Optimizing on Training Set

- Cross-validation
  - Partition data into *k* pieces (a.k.a. "folds")
  - For each piece *p*
    - train on all pieces but *p*, test on *p*
    - Average the results
- Homework 3: 10-fold CV on training set
  - How well will this predict test set performance?

Today: Learning

- General Rules of Thumb in Learning
- **Learning in Graphical Models**
  - **Parameters in Bayes Nets**
  - Briefly: Continuous conditional distributions in Bayes Nets
  - Bias vs. Variance
  - Discriminative vs. Generative training
  - Parameters in Markov Nets

Learning in Graphical Models

- Problem Dimensions
  - Model
    - Bayes Nets
    - Markov Nets
  - Structure
    - Known
    - Unknown (structure learning)
  - Data
    - Complete
    - Incomplete (missing values or hidden variables)

Learning in Graphical Models

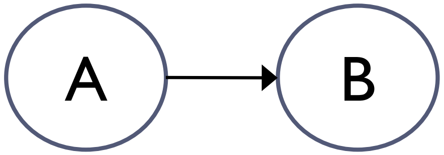
- Problem Dimensions (**today**)
  - Model
    - **Bayes Nets**
    - Markov Nets
  - Structure
    - **Known**
    - Unknown (structure learning)
  - Data
    - **Complete**
    - Incomplete (missing values or hidden variables)

Learning in Bayes Nets – the upshot

- Just statistical estimation for each CPT

Training Data

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



$P_{ML}(A) = 0.714$

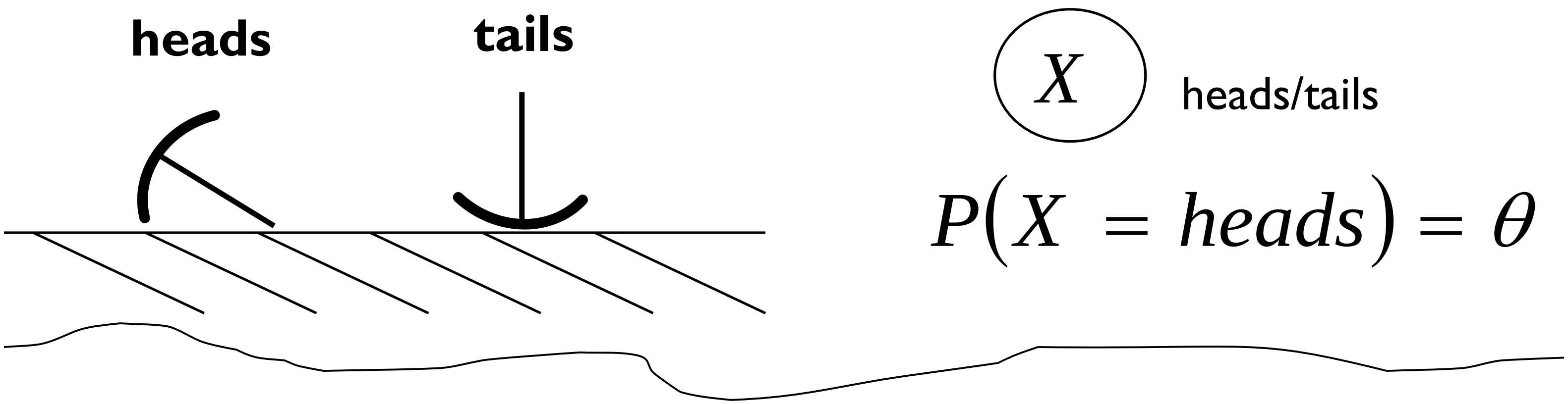
$P_{ML}(B \mid A=1) = 0.6$

Learning in Bayes Nets – details

- Problem statement (for today):
  - Given a Bayes Network structure *G*, and a set of complete training examples $\{\boldsymbol{X}_i\}$
  - Learn the CPTs for *G*.
- Assumption (as before in stat. estimation):
  Training examples are independent and identically distributed (i.i.d.) from an underlying distribution P^*
- *Why* just statistical estimation for each CPT?

Learning in Bayes Nets

- Thumbtack problem can be viewed as learning the CPT for a very simple Bayes Net:

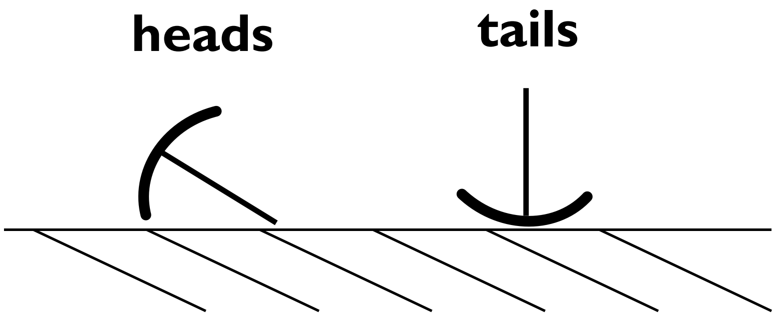


$$P(X = heads) = \theta$$

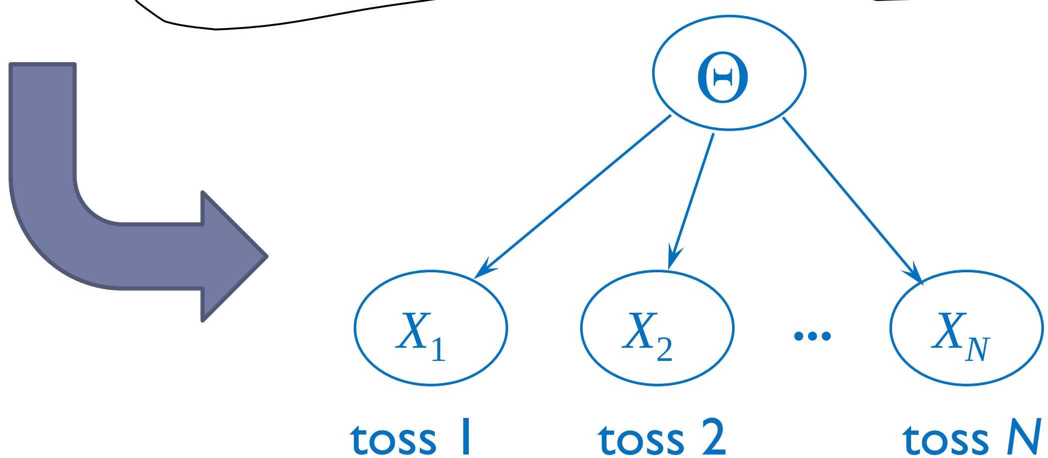
Thumbtack problem examples from Chris Meek, Microsoft Research

Learning as Inference

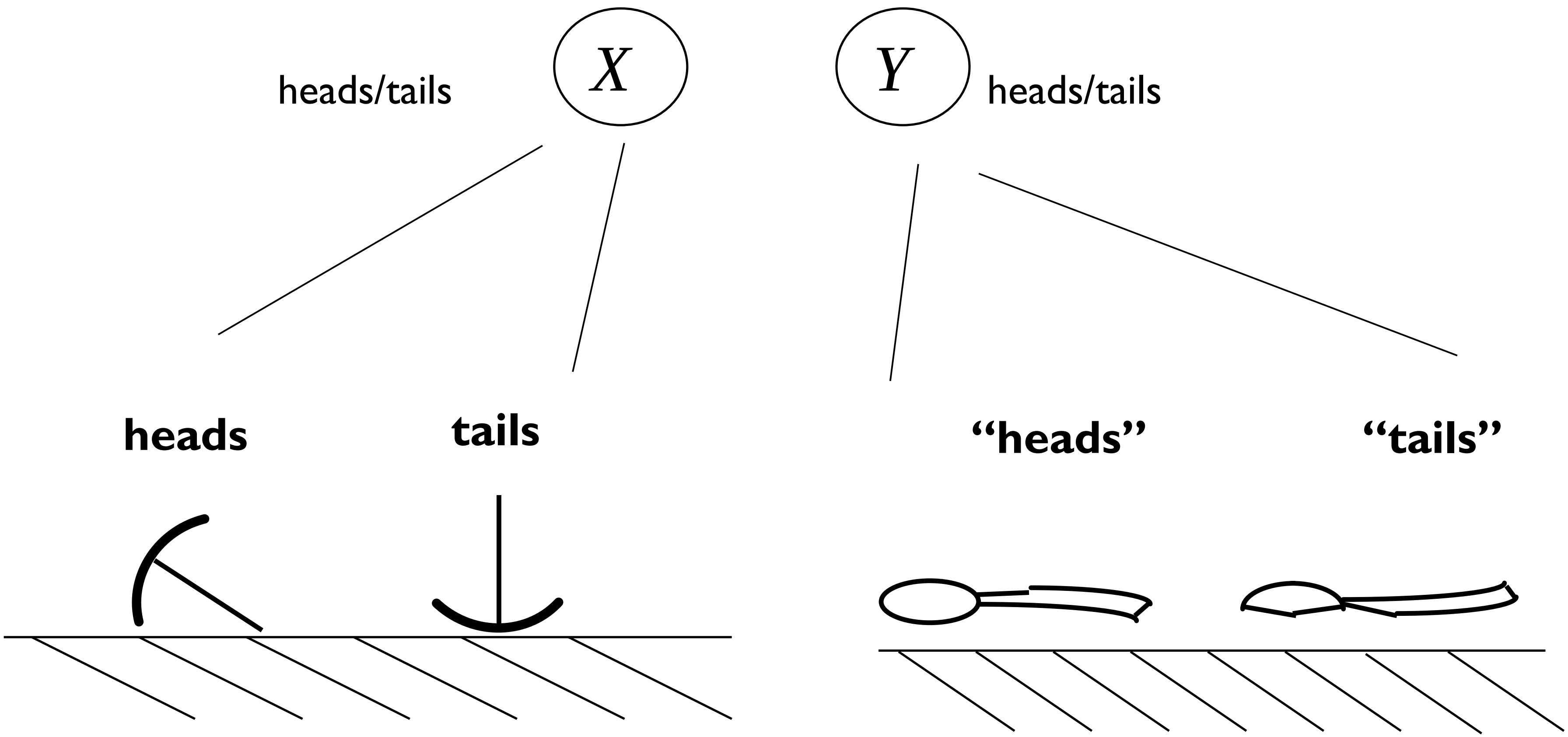
- Think of learning P(Θ = θ | {X_i}) as *inference*



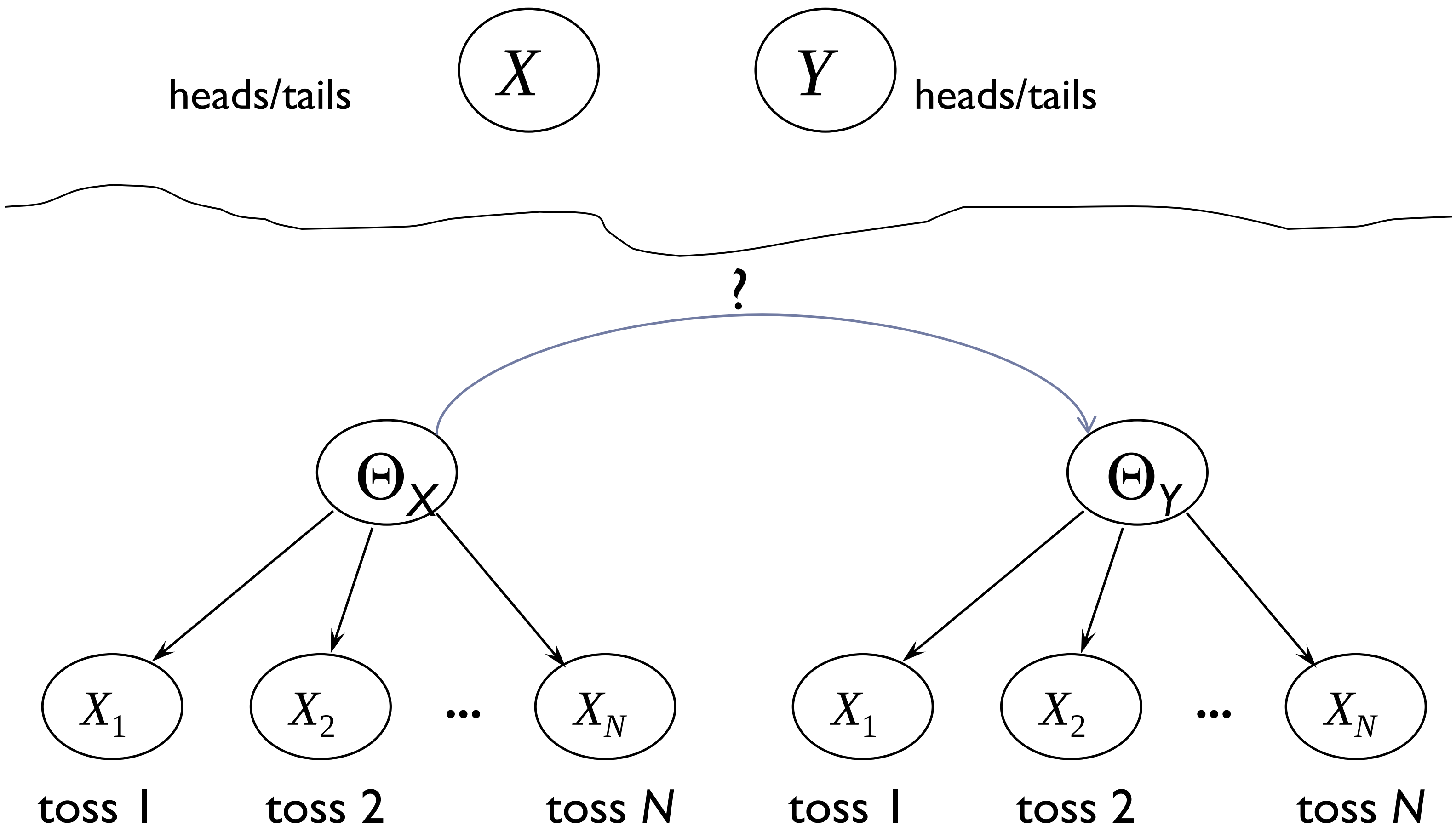
$$P\left(X_i = heads\right) = \theta$$



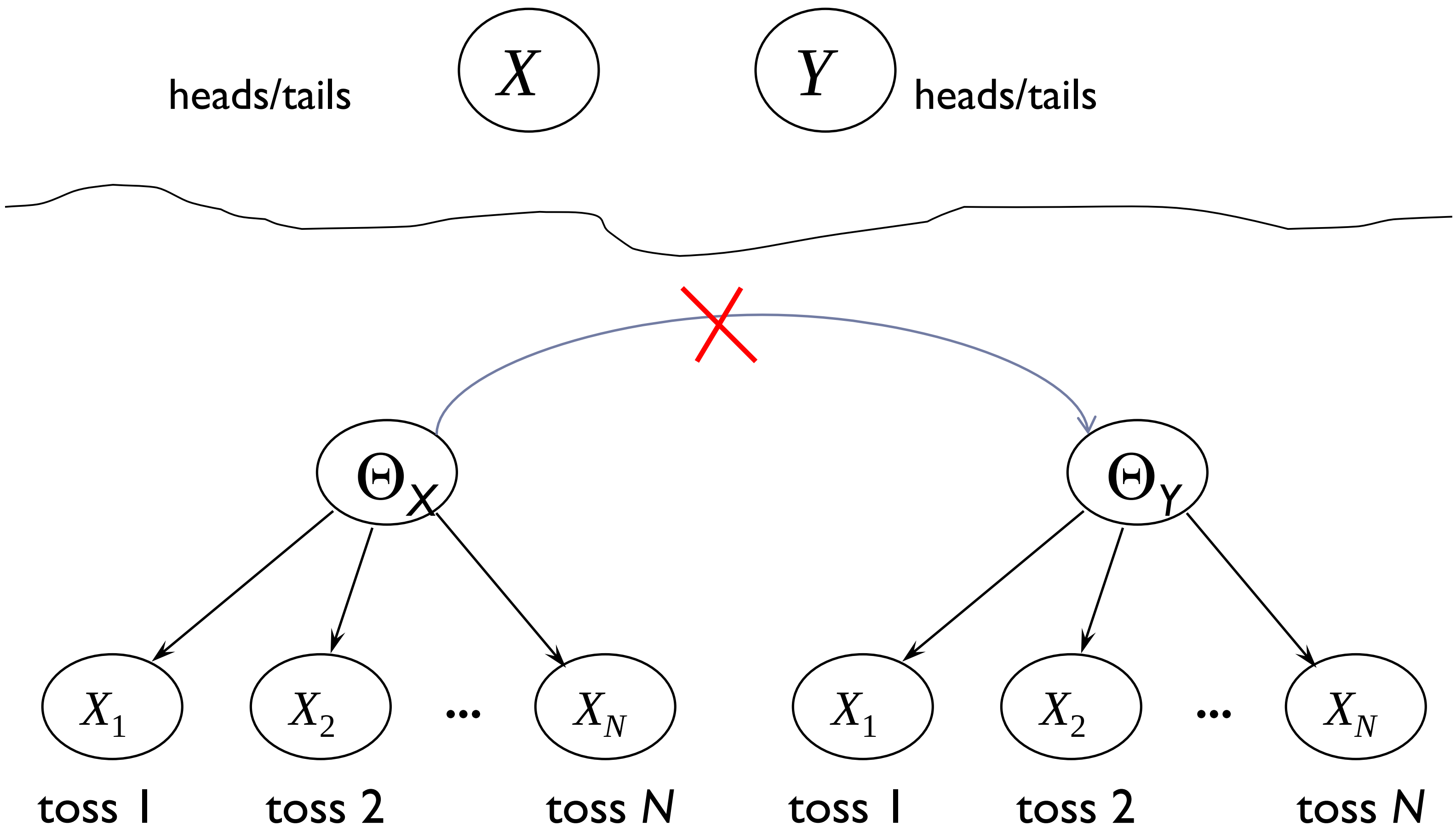
Next Simplest Bayes Net



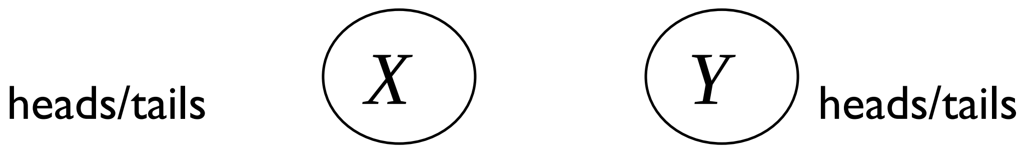
Next Simplest Bayes Net



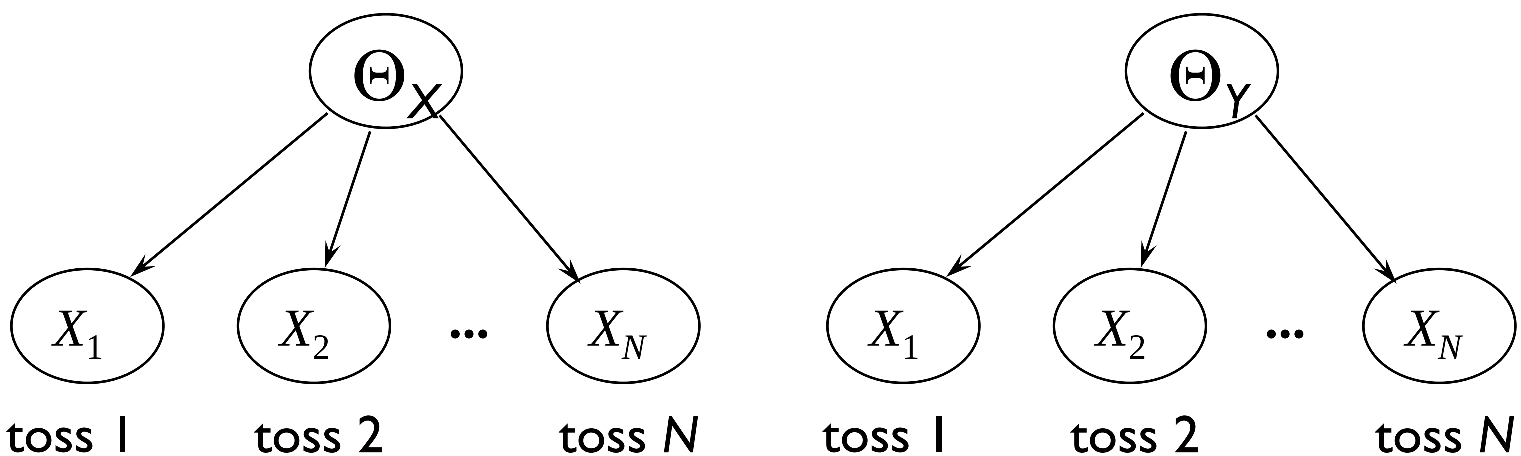
Next Simplest Bayes Net



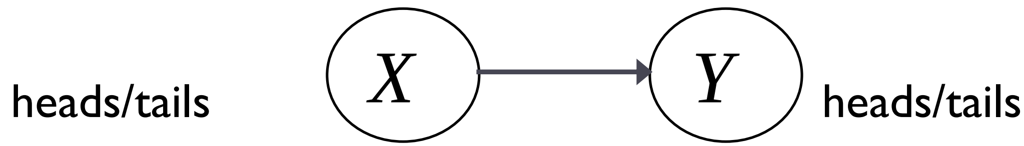
Next Simplest Bayes Net



"Parameter Independence"



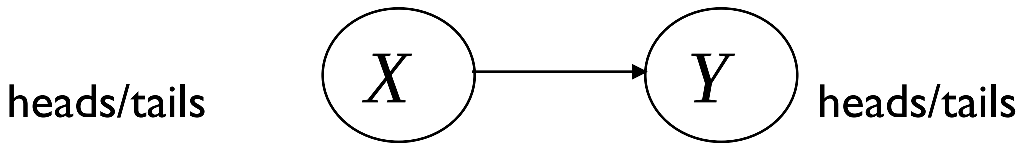
Getting Tougher

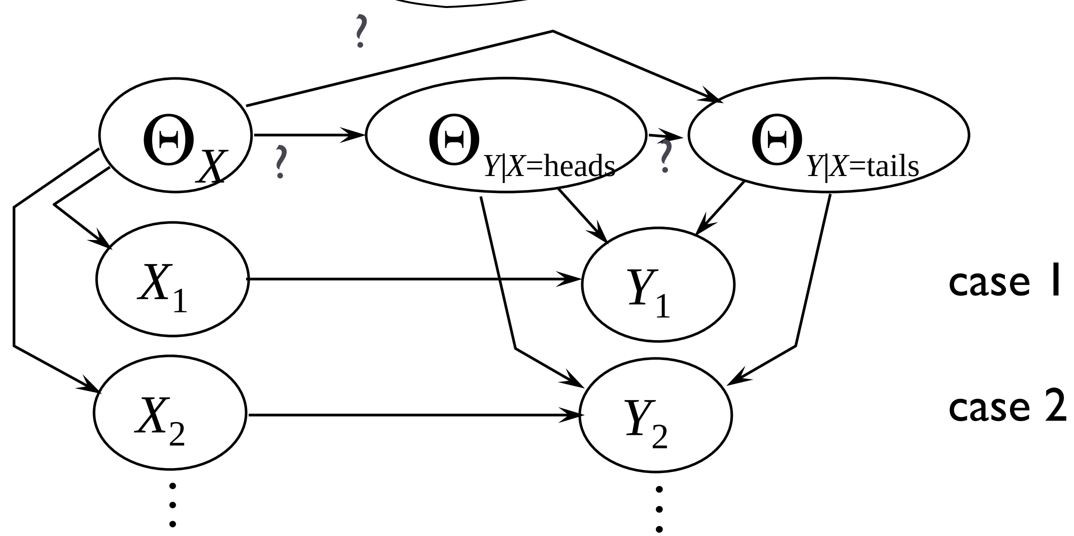


<u>Three probabilities to learn:</u>

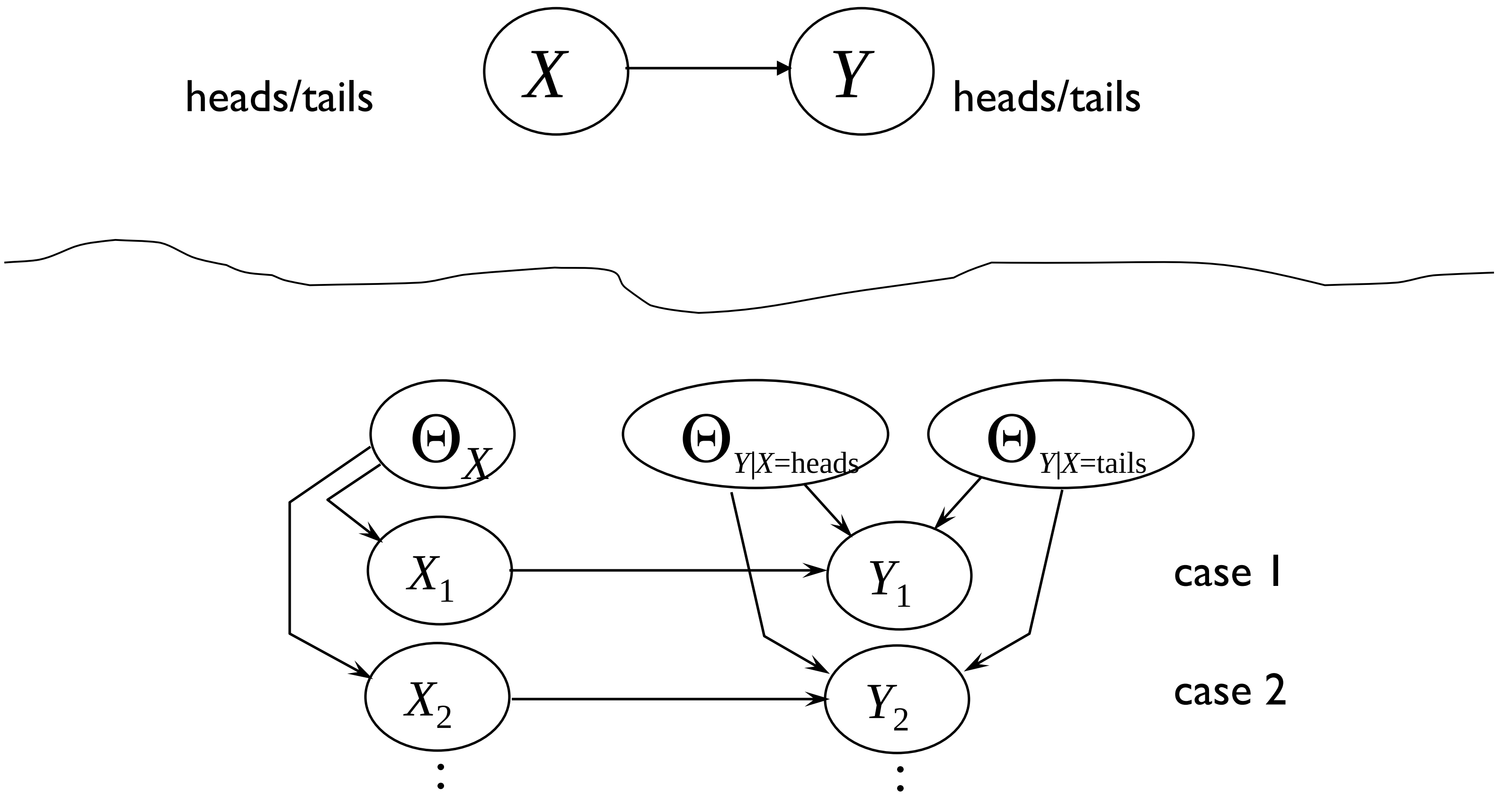
- $\theta_{X=\text{heads}}$
- $\theta_{Y=\text{heads}|X=\text{heads}}$
- $\theta_{Y=\text{heads}|X=\text{tails}}$

Learning as Inference

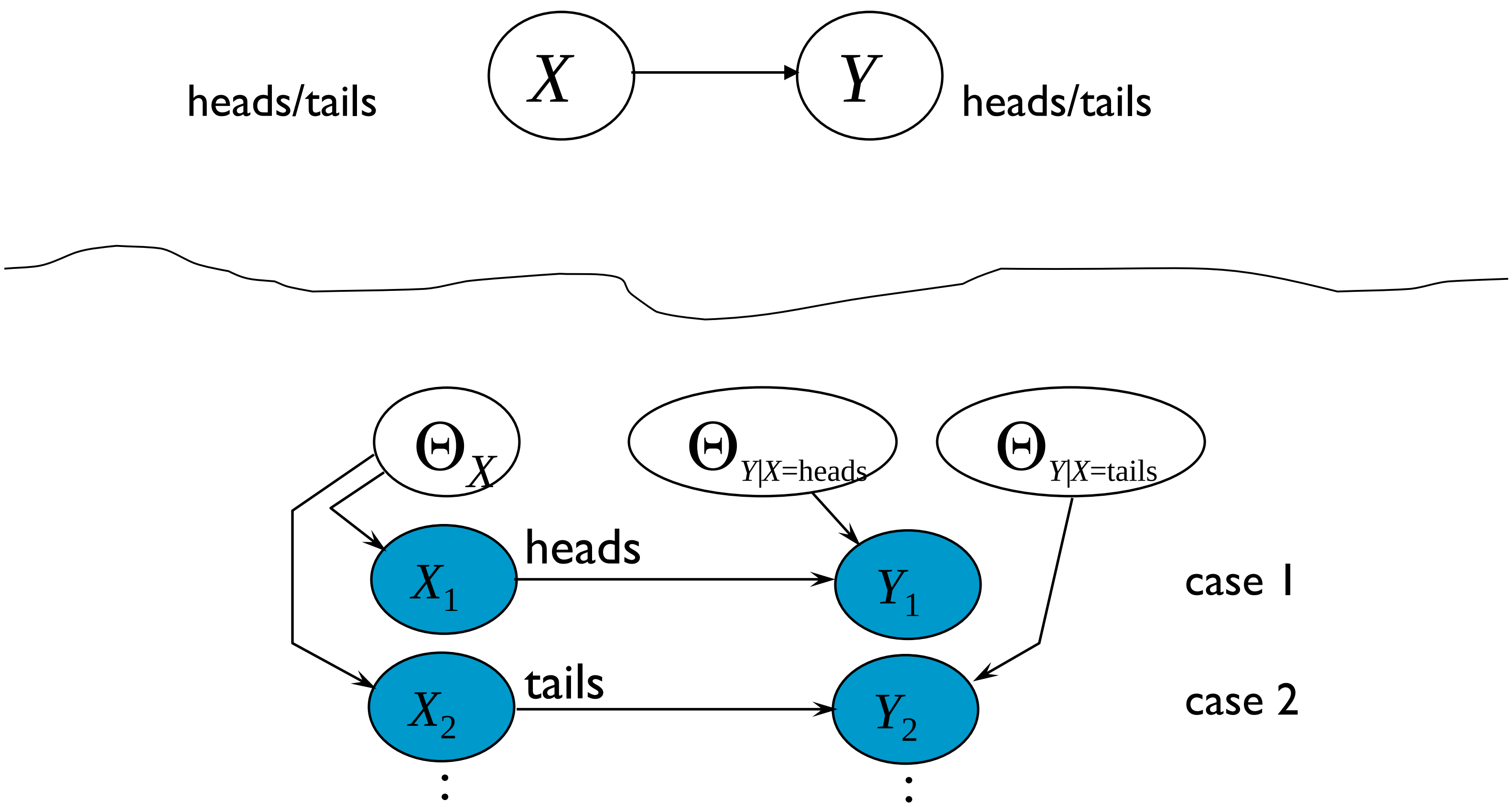




Parameter Independence



Three **Separate** Thumbtack Problems



Parameter Estimation in Bayes Nets

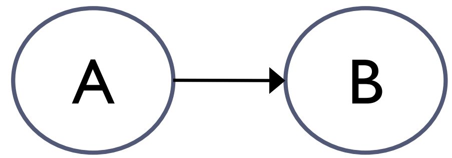
- Each CPT learned **independently**
- Easy when CPTs have convenient form
  - Multinomials
    - Maximum Likelihood = counting
  - Gaussian, Poisson, etc.
- And priors are conjugate ⟵
  - E.g. Beta for Binomials, etc.
- And data is complete

Parameter Priors

- MAP estimation

Training Data

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



$P_{ML}(B \mid A=0) = 2/2 = 1.0$

$P_{MAP}(B \mid A=0)$

$= (2+1)/(2+2) = 0.75$

"Laplace smoothing"

…same as $P(\Theta_{B \mid A=0}) = \text{Beta}(2, 2)$

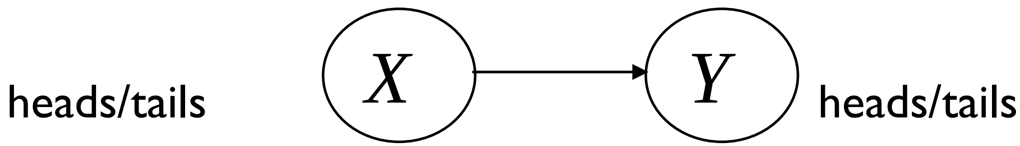
Parameter Estimation in Bayes Nets

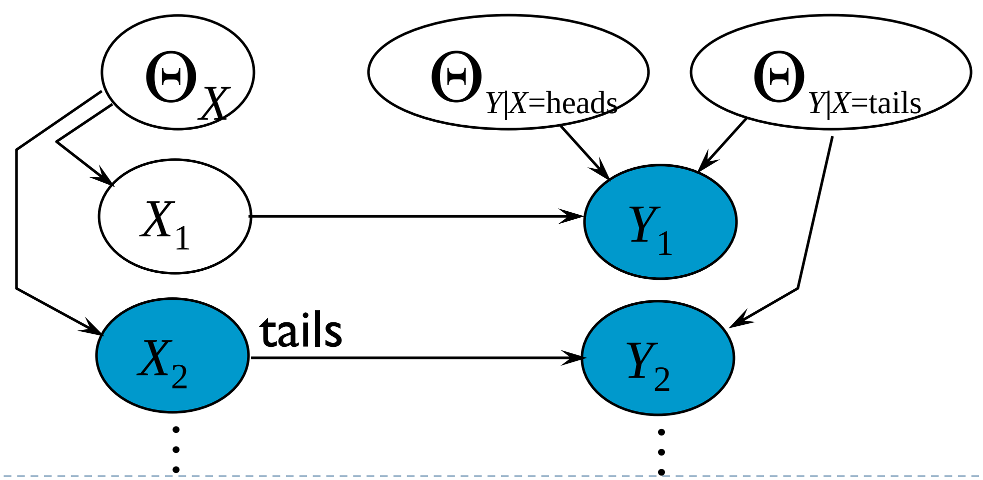
- Each CPT learned **independently**
- Easy when CPTs have convenient form
  - Multinomials
    - Maximum Likelihood = counting
  - Gaussian, Poisson, etc.
- And priors are conjugate
  - E.g. Beta for Binomials, etc.

- And data is complete ⇐

Incomplete Data

- Say we don't know X_1





Parameters are now dependent!

Incomplete Data in Practice

- Options:
  - Just ignore it (for all examples)
  - Replace missing X_i with most typical value in training set
  - Sample X_i from $P(X_i)$ in training set
  - Let "unknown" be a value for X_i
  - Try to *infer* missing values (special case: semi-supervised learning)

Today: Learning

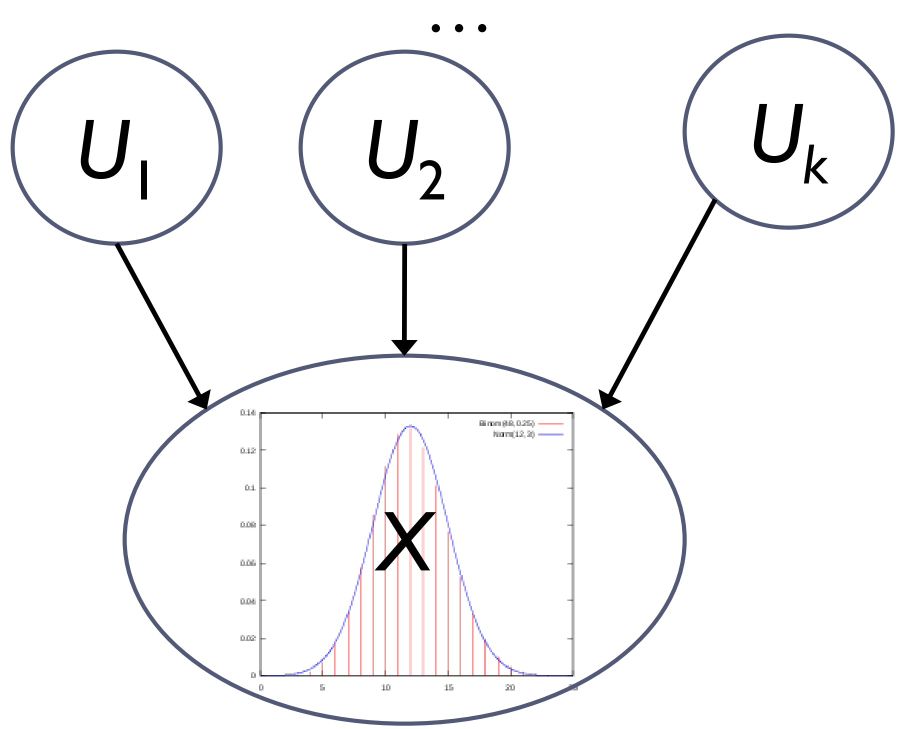
- General Rules of Thumb in Learning
- Learning in Graphical Models
  - Parameters in Bayes Nets
  - **Briefly: Continuous conditional distributions in Bayes Nets**
  - Bias vs. Variance
  - Discriminative vs. Generative training
  - Parameters in Markov Nets

Learning Continuous CPTs

- Options:
  - Discretize
    - Weka does this
    - Not a bad option
  - Use canonical functions
    - Gaussians most popular
    - see Matlab's package or WinMine, etc.

Continuous CPT Example

E.g., Linear Gaussian

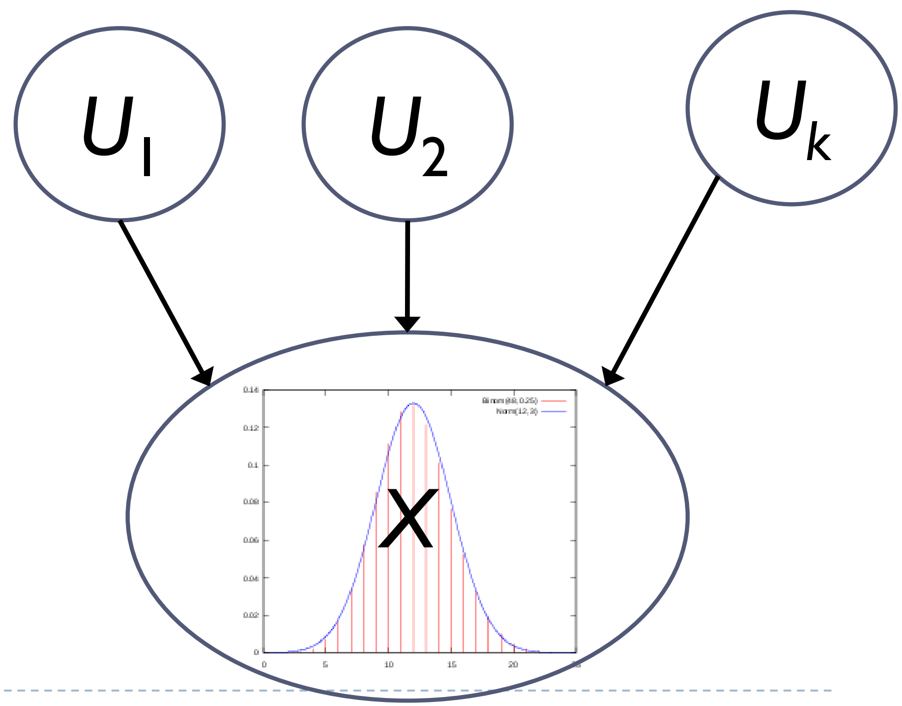


$P(X \mid \boldsymbol{u}) = N(\beta_0 + \beta_1 u_1 + \ldots\ \beta_k u_k; \sigma^2)$

Linear Gaussian

ML solution from system of equations, e.g.:

$$\boldsymbol{E}[X] = \beta_0 + \beta_1 \boldsymbol{E}[u_1] + \ldots \beta_k \boldsymbol{E}[u_k]$$



Today: Learning

- General Rules of Thumb in Learning
- Learning in Graphical Models
  - Parameters in Bayes Nets
  - Briefly: Continuous conditional distributions in Bayes Nets
  - **Bias vs. Variance**
  - Discriminative vs. Generative training
  - Parameters in Markov Nets

Bias vs. Variance

- Efficacy of learning varies with Bayes Net structure and amount of training data

Bayes Net design impacts learning

- Data required to learn a CPT **grows** roughly linearly with number of parameters
  - Fewer variables & edges is better
- Including **more** informative variables and relationships **improves** accuracy
  - *More* variables & edges is better (?)

- => selection of variables and edges is the art of Bayes Net design

Overfitting in Bayes Nets

- P(C | B) =

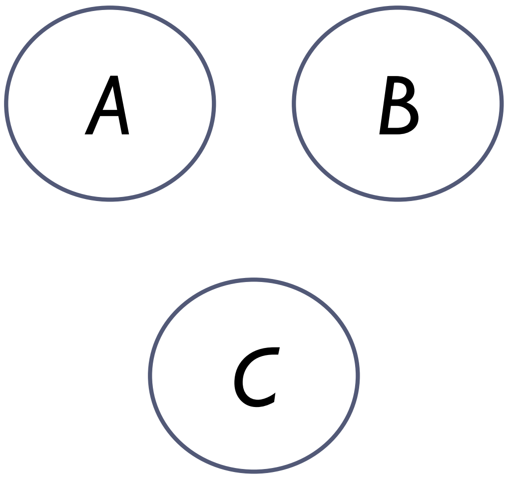
	P(C)
B=0	4/12
B=1	16/16

Training data is the following, repeated **4** times:

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

- Using P(C | A, B) => zero training error (vs. 17% error for P(C | B)), but cells have 12, 8, **4**, **4** total samples
- => Very susceptible to random noise

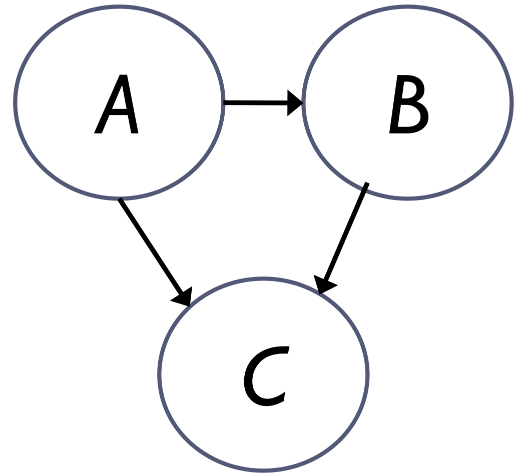
Bias vs. Variance (1 of 3)



High Bias

Low Variance

Underfitting

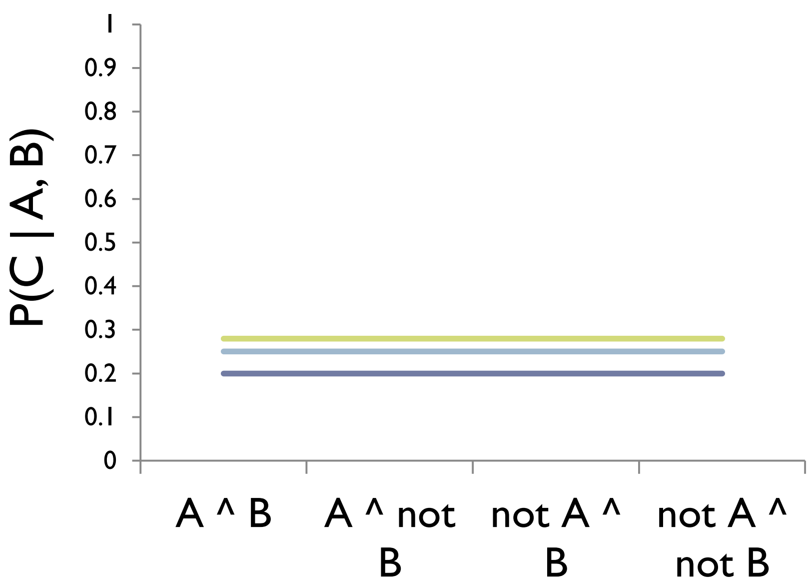


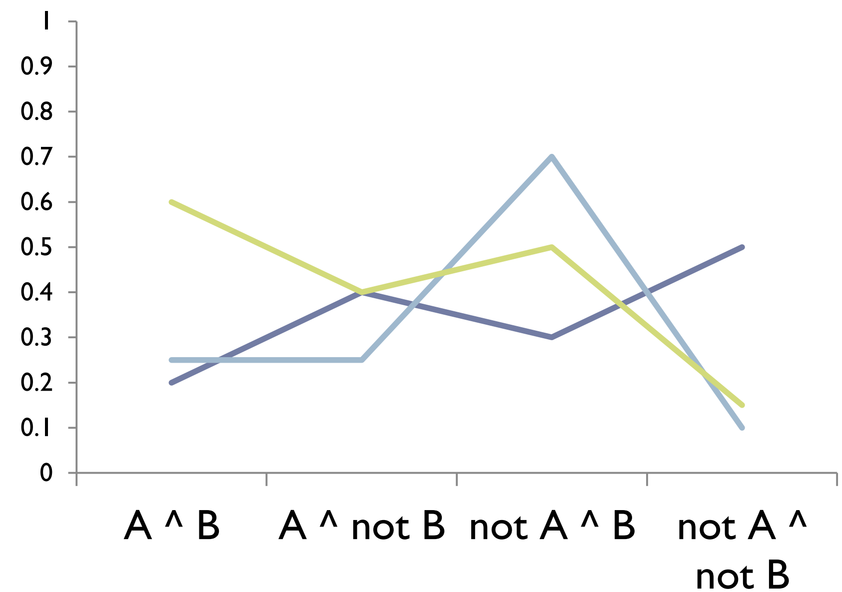
Low Bias

High Variance

Overfitting

Bias vs. Variance (2 of 3)





High Bias

Low Variance

Underfitting

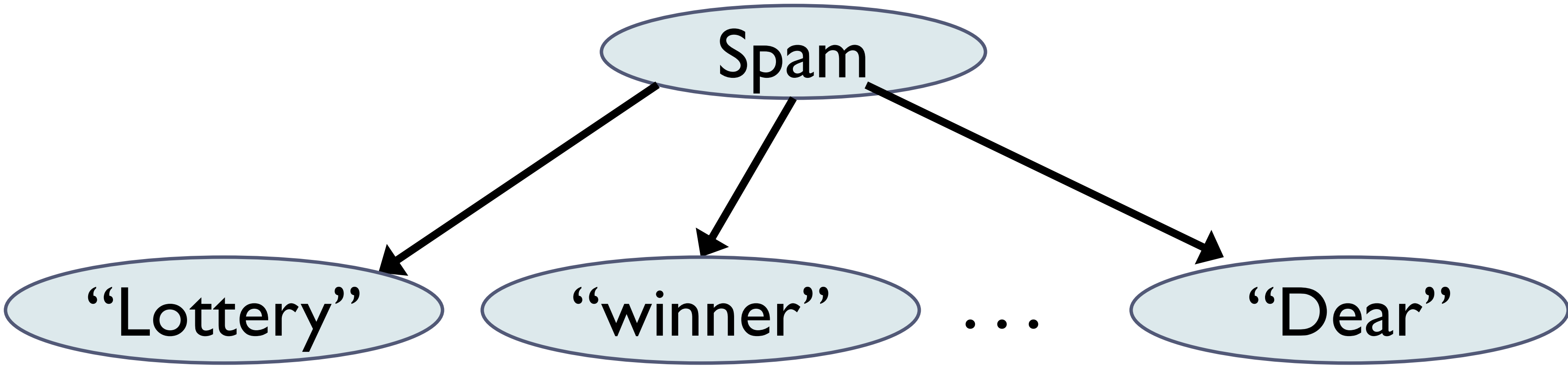
Low Bias

High Variance

Overfitting

Bias vs. Variance (3 of 3)

- High bias sometimes okay
  - E.g. Naïve Bayes effective in practice



How do you choose?

- Cross-validation

- And/or use heuristics for trading training accuracy for model complexity
  - Useful in automated structure learning
  - E.g., pick a structure and algorithmically refine
  - Later

Learning

- General Rules of Thumb in Learning

- Learning in Graphical Models
  - Parameters in Bayes Nets
  - Briefly: Continuous conditional distributions in Bayes Nets
  - Bias vs. Variance
  - **Discriminative vs. Generative training**
  - Parameters in Markov Nets

Discriminative vs. Generative training

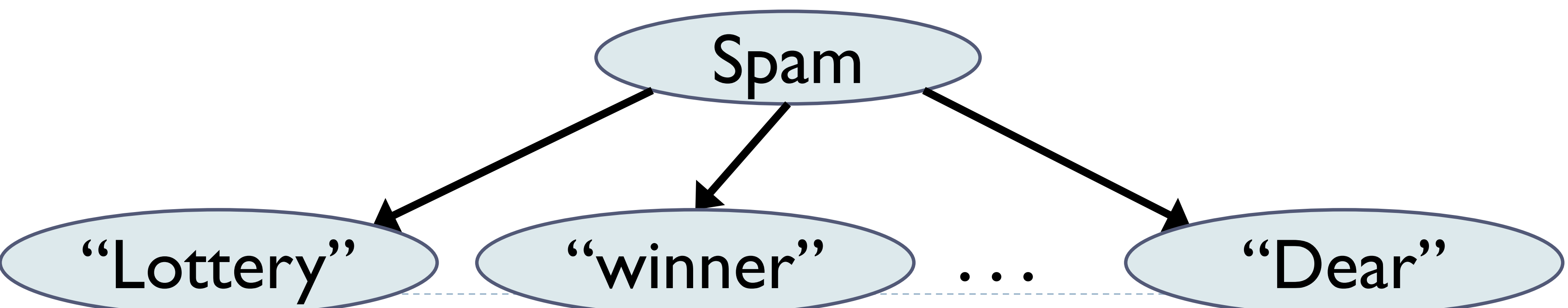
- Say our graph *G* has variables ***X*** , ***Y***
- Previous method learns P(***X*** , ***Y***)
- But often, the only inferences we care about are of form P(***Y*** | ***X***)
  - P(*Disease* | *Symptoms* = **e**)
  - P(*StockMarketCrash* | *RecentPriceActivity* = **e**)

Discriminative vs. Generative training

- Learning P(***X*** , ***Y***): **generative** training
  - Learned model can "generate" the full data ***X, Y***
- Learning only P(***Y*** | ***X***): **discriminative** training
  - Model can't assign probs. to ***X*** – only ***Y*** given ***X***
- Idea: Only model what we care about
  - Don't "waste data" on params irrelevant to task
  - Side-step false independence assumptions in training (example to follow)

Generative Model Example

- Naïve Bayes model
  - *Y* binary {1=spam, 0=not spam}
    X an *n*-vector: message has word (1) or not (0)
  - Re-write P(Y | ***X***) using Bayes Rule, apply Naïve Bayes assumption
  - 2*n* + 1 parameters, for *n* observed variables

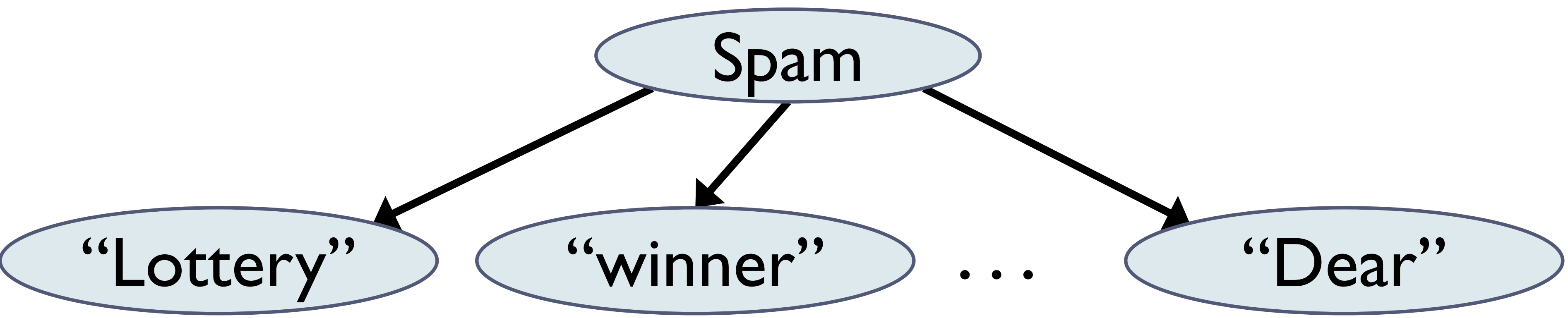


Generative => Discriminative (1 of 3)

- But P(*Y* | ***X***) can be written more compactly

$$P(Y \mid \mathbf{X}) = \frac{1}{1 + \exp(w_0 + w_1 x_1 + \ldots + w_n x_n)}$$

- Total of n + 1 parameters w_i



Generative => Discriminative (2 of 3)

- One way to do conversion (vars binary):

$$\exp(w_0) = \frac{P(Y=0)\, P(X_1=0|Y=0)\, P(X_2=0|Y=0)\ldots}{P(Y=1)\, P(X_1=0|Y=1)\, P(X_2=0|Y=1)\ldots}$$

for $i > 0$:

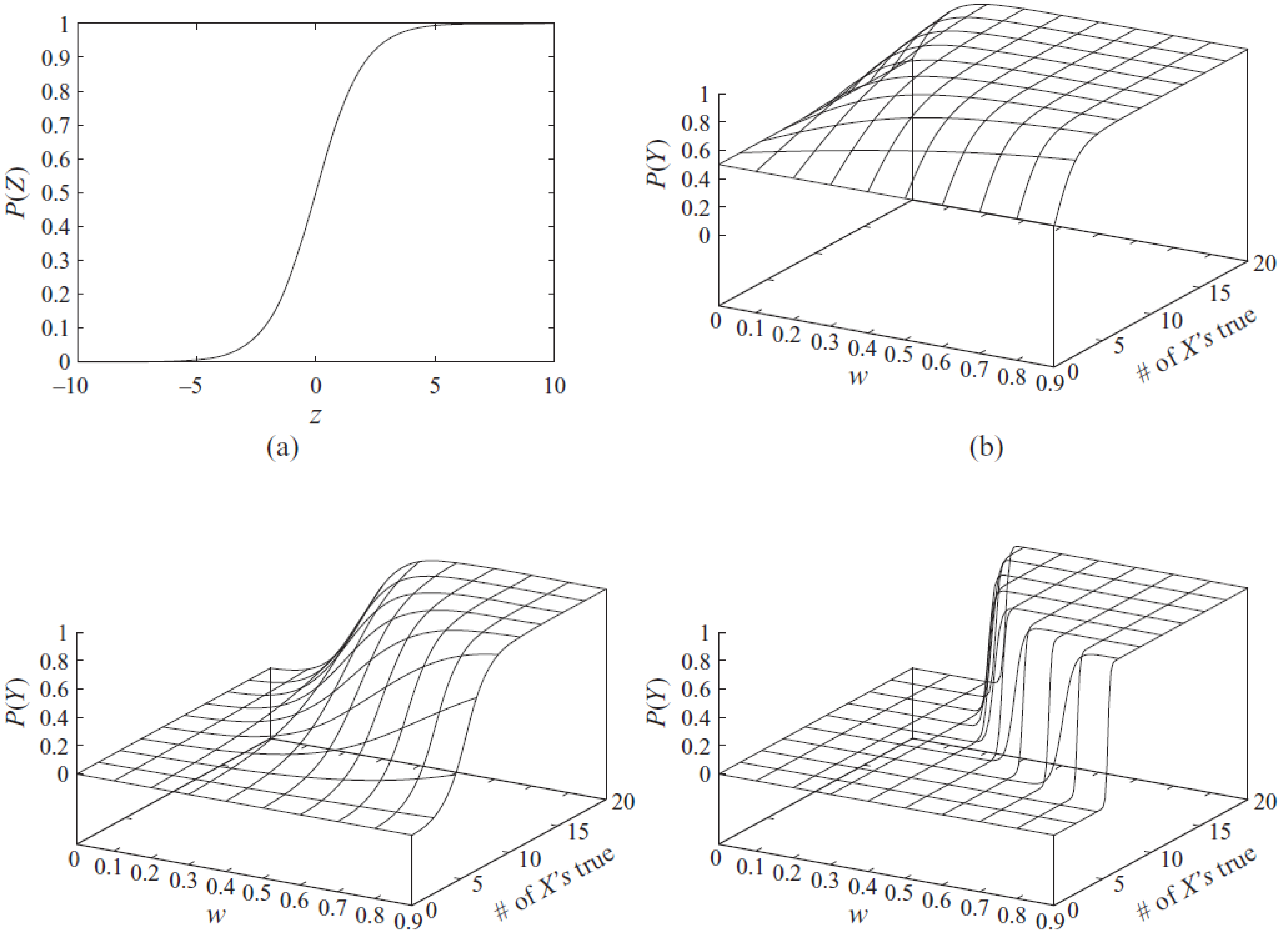
$$\exp(w_i) = \frac{P(X_i=0|Y=1)\, P(X_i=1|Y=0)}{P(X_i=0|Y=0)\, P(X_i=1|Y=1)}$$

Generative => Discriminative (3 of 3)

- We reduced $2n + 1$ parameters to $n + 1$
  - Bias vs. Variance arguments says this must be better, right?
- Not exactly. If we construct $P(Y \mid \boldsymbol{X})$ to be equivalent to Naïve Bayes (as before)
  - then it's…equivalent to Naïve Bayes
- Idea: optimize the $n + 1$ parameters directly, using training data

Discriminative Training

- In our example:
  $$P(Y \mid \boldsymbol{X}) = \frac{1}{1 + \exp(w_0 + w_1 x_1 + \ldots + w_n x_n)}$$
- Goal: find w_i that maximize likelihood of training data *Y*s given training data ***X***s
  - Known as "logistic regression"
  - Solved with gradient ascent techniques
  - A convex (actually concave) optimization problem


P(Z)
z
(a)
P(Y)
w
# of X's true
(b)
P(Y)
w
# of X's true
P(Y)
w
# of X's true

Naïve Bayes vs. LR

- Naïve Bayes "trusts its assumptions" in training
- Logistic Regression doesn't – recovers better when assumptions violated

NB vs. LR: Example

Training Data

SPAM	Lottery	Winner	Lunch	Noon
1	1	1	0	0
1	1	1	1	1
0	0	0	1	1
0	1	1	0	1

- Naïve Bayes will classify the last example incorrectly, even after training on it!
- Whereas Logistic Regression is perfect with e.g.,
  $w_0 = 0.1 \quad w_{\text{lottery}} = w_{\text{winner}} = w_{\text{lunch}} = -0.2 \quad w_{\text{noon}} = 0.4$

Logistic Regression in practice

- Can be employed for any numeric variables X_i
  - or for other variable types, by converting to numeric (e.g. indicator) functions
- "Regularization" plays the role of priors in Naïve Bayes
- Optimization tractable, but (way) more expensive than counting (as in Naïve Bayes)

Discriminative Training

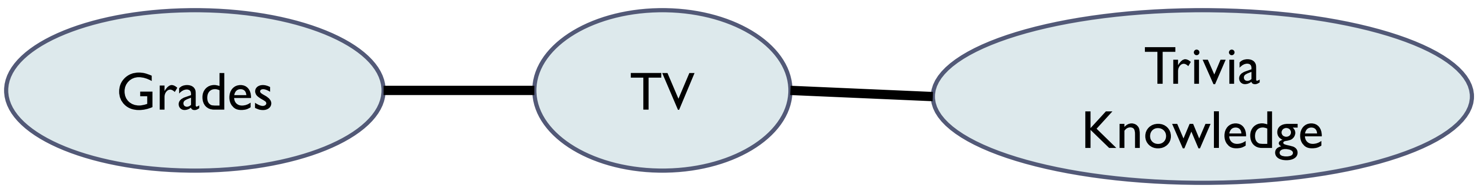
- Naïve Bayes vs. Logistic Regression one illustrative case
- Applicable more broadly, whenever queries P(***Y*** | ***X***) known *a priori*

Learning

- General Rules of Thumb in Learning
- Learning in Graphical Models
  - Parameters in Bayes Nets
  - Briefly: Continuous conditional distributions in Bayes Nets
  - Bias vs. Variance
  - Discriminative vs. Generative training
  - **Parameters in Markov Nets**

Recall: Markov Networks

- Undirected Graphical Model
  - **Potential functions** ϕ_c defined over cliques
- $P(\mathbf{x}) = \dfrac{\prod_c \phi_c(\mathbf{x}_c)}{Z}$ $\quad\quad Z = \sum_{\mathbf{x}} \prod_c \phi_c(\mathbf{x}_c)$



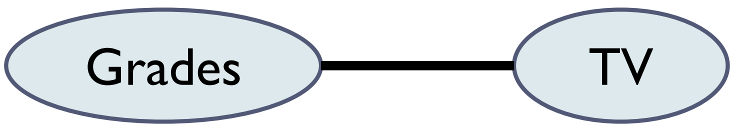
Grades	TV	ϕ_1(G,TV)
bad	none	2.0
good	none	3.0
bad	lots	3.0
good	lots	1.0

TV	Trivia Knowledge	ϕ_2(TV, K)
none	weak	2.0
lots	weak	1.0
none	strong	1.5
lots	strong	3.0

Log-linear Formulation (1 of 2)

- $P(\boldsymbol{x}) = \dfrac{\exp(\Sigma_i \, w_i \, f_i \, (\boldsymbol{D}_i) \,)}{Z}$
- E.g.: write $\phi_1(G, TV)$ as $\exp(w_1 \, f_1(G, TV) + \ldots + w_4 \, f_4(G, TV))$

$w_1 = \ln 2.0 \quad w_2 = \ln 3.0 \quad w_3 = \ln 3.0 \quad w_4 = \ln 1.0$



Grades	TV	$\phi_1(G,TV)$	$f_1(G,TV)$	$f_2(G,TV)$	$f_3(G,TV)$	$f_4(G,TV)$
bad	none	2.0	1	0	0	0
good	none	3.0	0	1	0	0
bad	lots	3.0	0	0	1	0
good	lots	1.0	0	0	0	1

Log-linear Formulation (2 of 2)

- $P(\boldsymbol{x}) = \dfrac{\exp(\Sigma_i \, w_i \, f_i \, (\boldsymbol{D}_i) \,)}{Z}$

- Why?
  - "Feature" f_i can be simpler than full potentials
  - Learning easy to express

Learning in Markov Networks

- Harder than in Bayes Nets
- Why? In Bayes Nets, likelihood is:
  - $P(\text{Data} \mid \theta) = \Pi_{m \in \text{Data}} \Pi_{\text{i}}\, P(X_i[m] | \text{Parents}(X_i)[m] : \theta_{\text{i}})$

    where X_i[m] is the assignment to X_i in example *m*

    $= \Pi_i\, \Pi_{m \in \text{Data}}\, P(X_i[m] | \text{Parents}(X_i)[m] : \theta_{\text{i}})$
  - Assuming param independence, maximize global likelihood by maximizing each CPT likelihood $\Pi_{m \in \text{Data}}\, P\,(X_i[m] \mid \text{Parents}(X_i)[m] : \theta_{\text{i}})$ **independently**

Learning in Markov Networks

- Harder than in Bayes Nets
- In Markov Net,

  Likelihood =

$$P(\text{Data} \mid \mathbf{w}) = \prod_{m \in \text{Data}} \frac{\exp(\sum_i w_i f_i(\boldsymbol{D}_i[m]))}{Z_{\mathbf{w}}}$$

- But $Z_{\mathbf{w}} = \sum_{\mathbf{x} \in Val(\boldsymbol{X})} \exp(\sum_i w_i f_i(\boldsymbol{x}))$
  - Sum over exps involving all w_i
- **Can't** decompose as we did in Bayes Net case

So what do we do?

- Maximize likelihood using Gradient Ascent
  - Or 2nd order optimization

- $\partial / \partial w_i \ln P(\text{Data} \mid \mathbf{w}) = \boldsymbol{E}_{\text{Data}}[f_i(\boldsymbol{D}_i)] - \boldsymbol{E}_{\mathbf{w}}[f_i]$

- Concave (no local maxima)
- Requires inference at each step
  - Slow

Approximation: Pseudo-likelihood

- Pseudo-likelihood PL(Data | θ) =

  $\Pi_{m \in \mathrm{Data}} \Pi_{\mathrm{i}}$ P($X_i[m]$| Neighbors(X_i)[m] : θ_i)

  - Assume variables depend only on values of neighbors in data

- No more Z!
  - Easier to compute/optimize (decomposes)

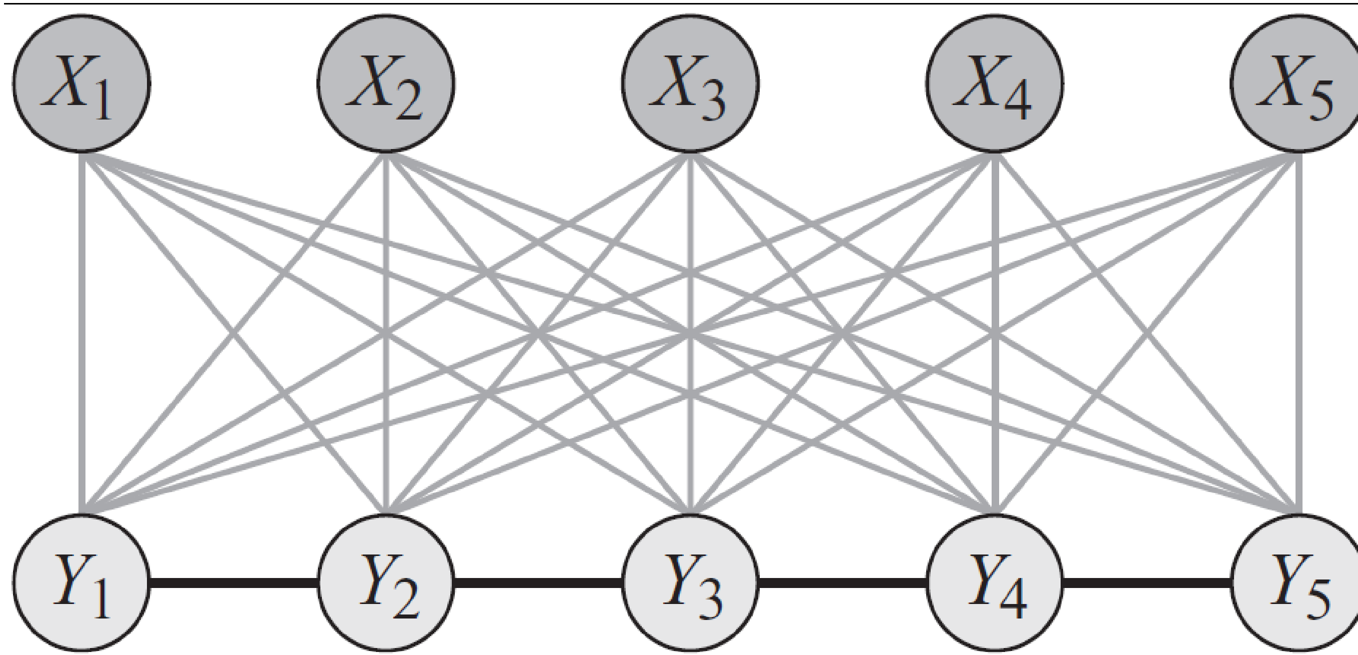
- But not necessarily a great approximation
  - Equal to likelihood in limit of infinite training data

Discriminative Training

- Learn P(***Y*** | ***X***)
- $\partial / \partial w_i \ln P(\boldsymbol{Y}_{\text{Data}} | \boldsymbol{X}_{\text{Data}}, \mathbf{w}) =$

$$\sum_m (f_i(\boldsymbol{y}[m], \boldsymbol{x}[m]) - \boldsymbol{E}_{\boldsymbol{w}}[f_i \mid x[m]])$$

- Rightmost term: run inference for each value ***x***[m] in data



What have we learned?

- General Rules of Thumb in Learning

- Learning in Graphical Models
  - Parameters in Bayes Nets
  - Briefly: Continuous conditional distributions in Bayes Nets
  - Bias vs. Variance
  - Discriminative vs. Generative training
  - Parameters in Markov Nets

Rest of course

- Next:
  - Structure Learning
- After that:
  - learning with missing data (semi-supervised learning), HMMs