Introduction to Artificial Intelligence

Lecture 18 – Learning Bayes Nets

CS/CNS/EE 154

Andreas Krause

Announcements

- Exam:
  - Take home
  - December 8 10am till December 9 10am
  - Open book and lecture notes
  - No other materials, no collaboration
  - Recitation next Thursday
- Final project due December 7

(Supervised) Learning

- Want to learn $f : \mathcal{X} \rightarrow \mathcal{Y}$

  X: Set up inputs (discrete, continuous)

  Y: Set of outputs

  from i.i.d. training examples

$$(x_1, y_1), ..., (x_N, y_N) \sim P(X, Y)$$

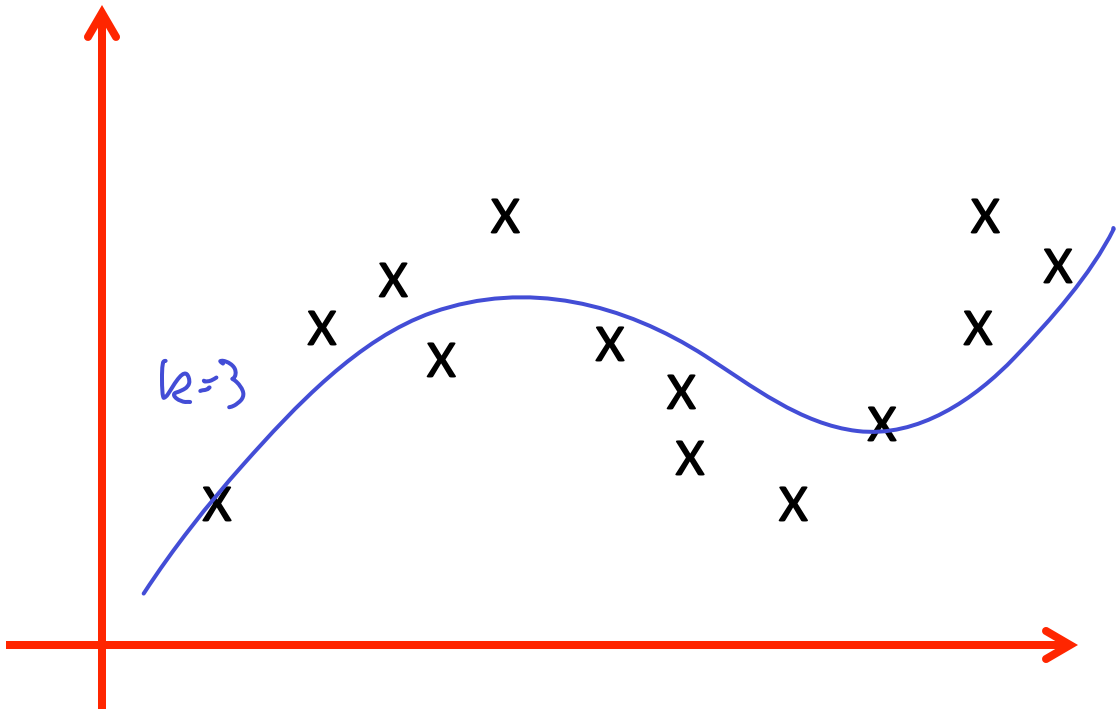
- Goal: minimize generalization error

$$\mathbb{E}_{X,Y}[\ell(Y, f(X))]$$

with loss function ℓ, for example:

$$\ell(y, f(x)) = (y - f(x))^2$$

Linear regression: squared loss



Assume

$$f(x) = \sum_{i=0}^{k} w_i x^i$$

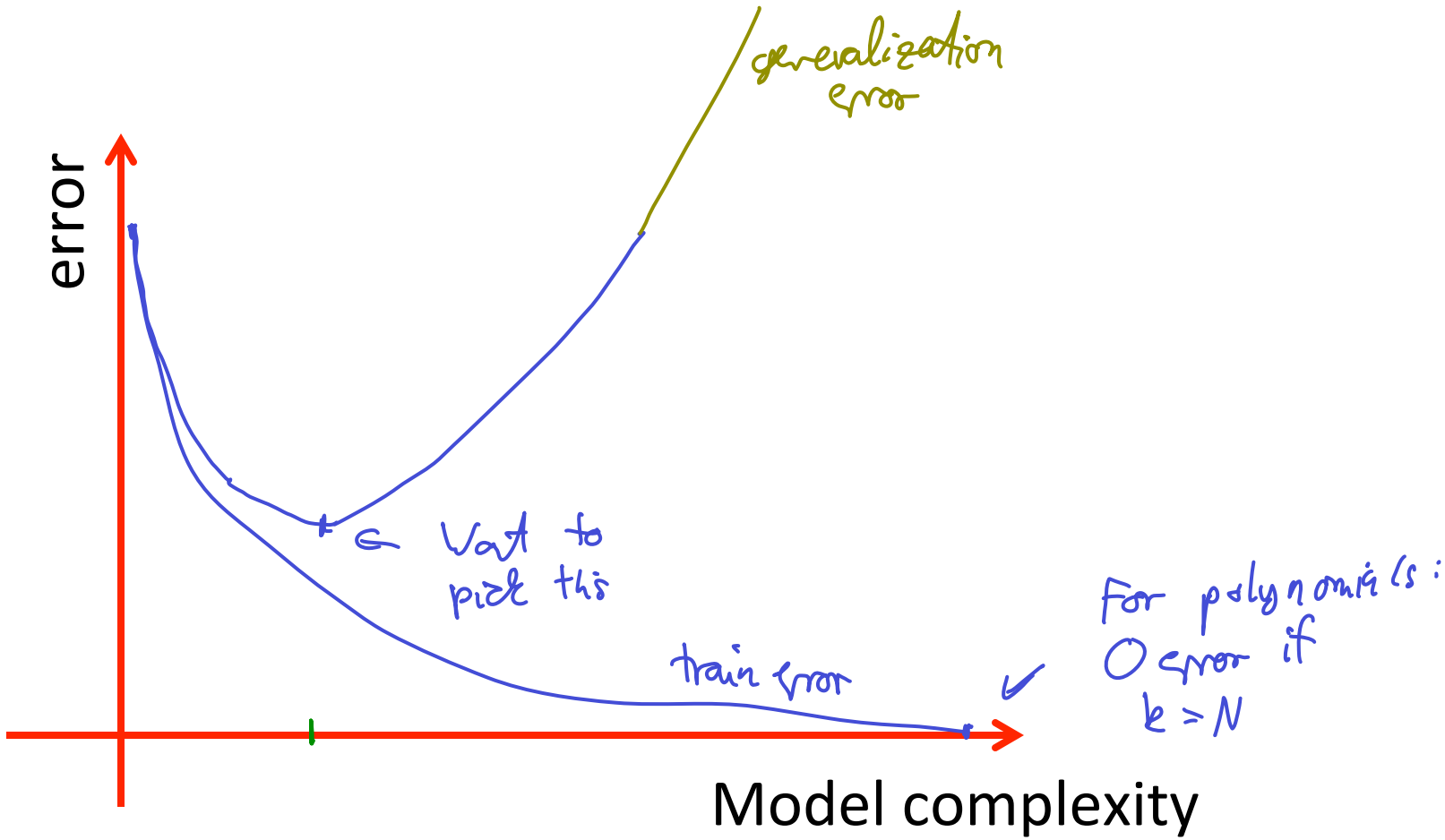
$$\tilde{x} = [1, x, x^2, x^3, \dots x^k]$$

$$f(\tilde{x}) = w^T \tilde{x}$$

linear regression!

Overfitting

- Min. training error ≠ min. generalization error!



Regularization ≈ Posterior inference

- A priori, assume weights should be small
  ➔ need fewer bits to describe, simpler model

E.g.: $P(w) = \mathcal{N}(0; \lambda^2 \cdot I)$

$$\arg\max_w P(w \mid D) = \arg\max_w P(w) \cdot P(D \mid w)$$

$$= \arg\max_w \ln P(w) + \ln P(D \mid w)$$

$$= \arg\min_w \underbrace{\frac{1}{\sigma^2} \sum_{i=1}^{N} (y_i - w^T x_i)^2}_{\text{Log likelihood}} + \underbrace{\boxed{\lambda^2} \sum_{j=1}^{k} w_i^2}_{\text{log prior}}$$

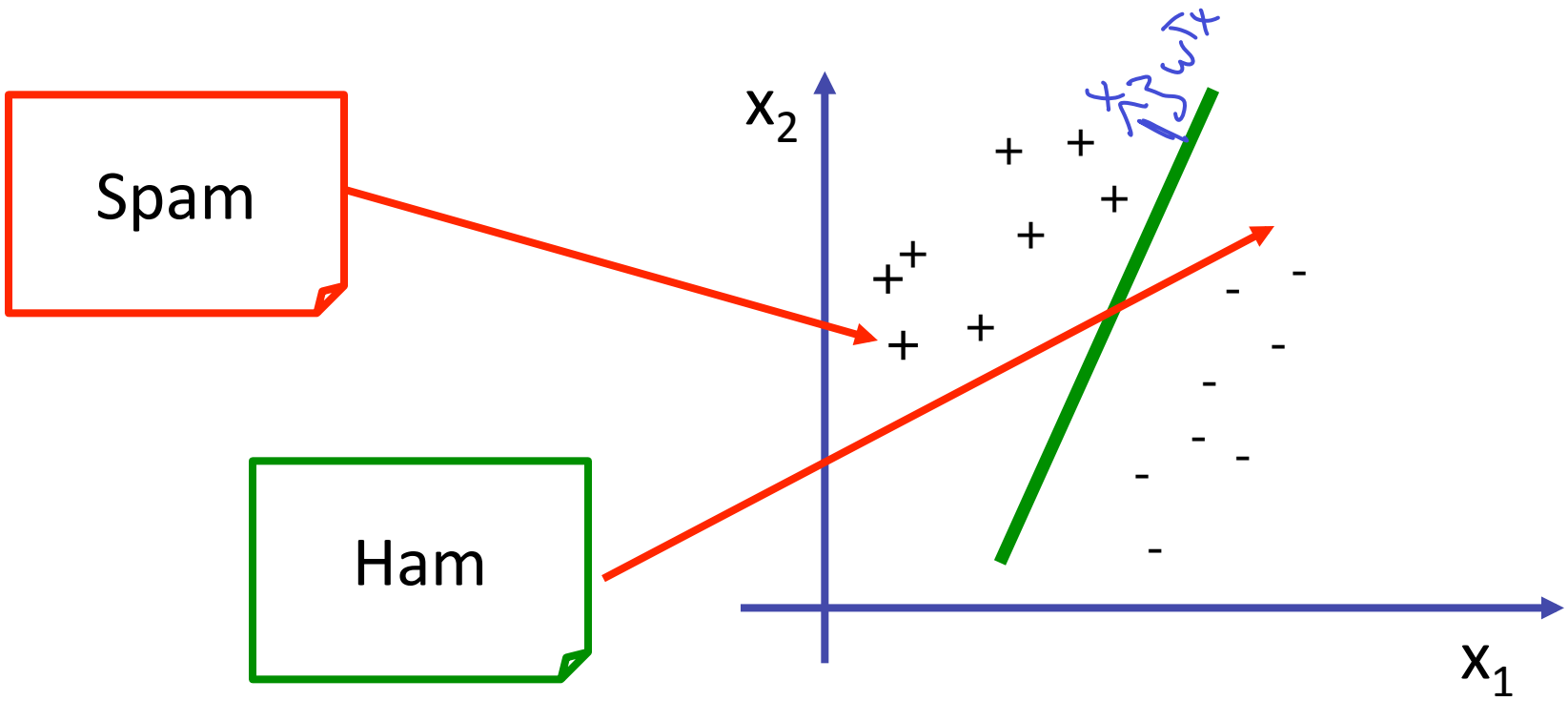
How should we choose λ?

Log likelihood = Fit to data

log prior = Complexity of model

Classification

- In classification, want to predict discrete label
- For example: binary linear classification $f: X \to \{+1, -1\}$



$$f(x;w) = \text{sign}(w_0 + w_1 x_1 + w_2 x_2)$$

linear classifier

0-1 loss

- Predict according to $f(x; w) = \text{sign}(w^T x)$

0-1 loss: $\ell(y, f(x; w)) = \begin{cases} 1 & \text{if } y \neq \text{sign}(w^T x) \\ 0 & \text{otherwise} \end{cases}$

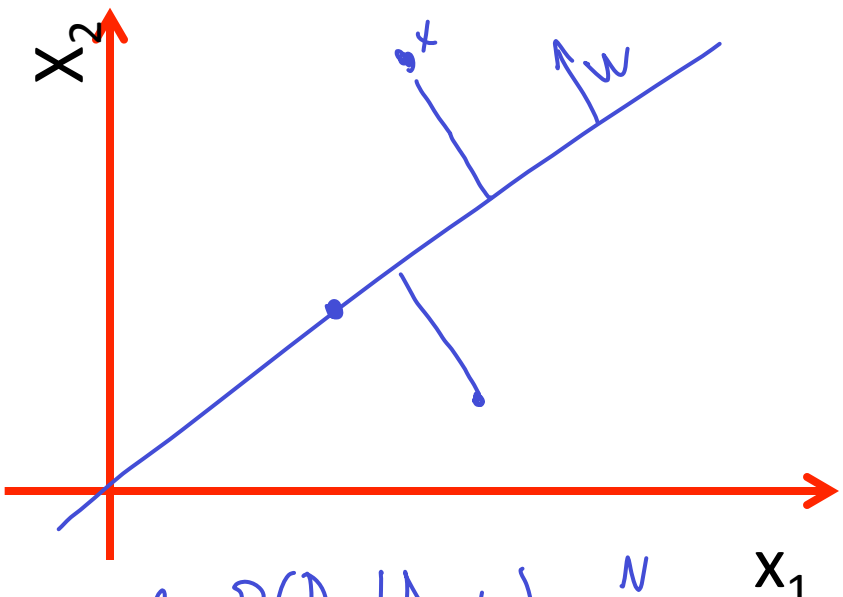
$w^* \in \underset{w}{\text{argmin}} \sum_i \ell(y_i; f(x_i, w))$

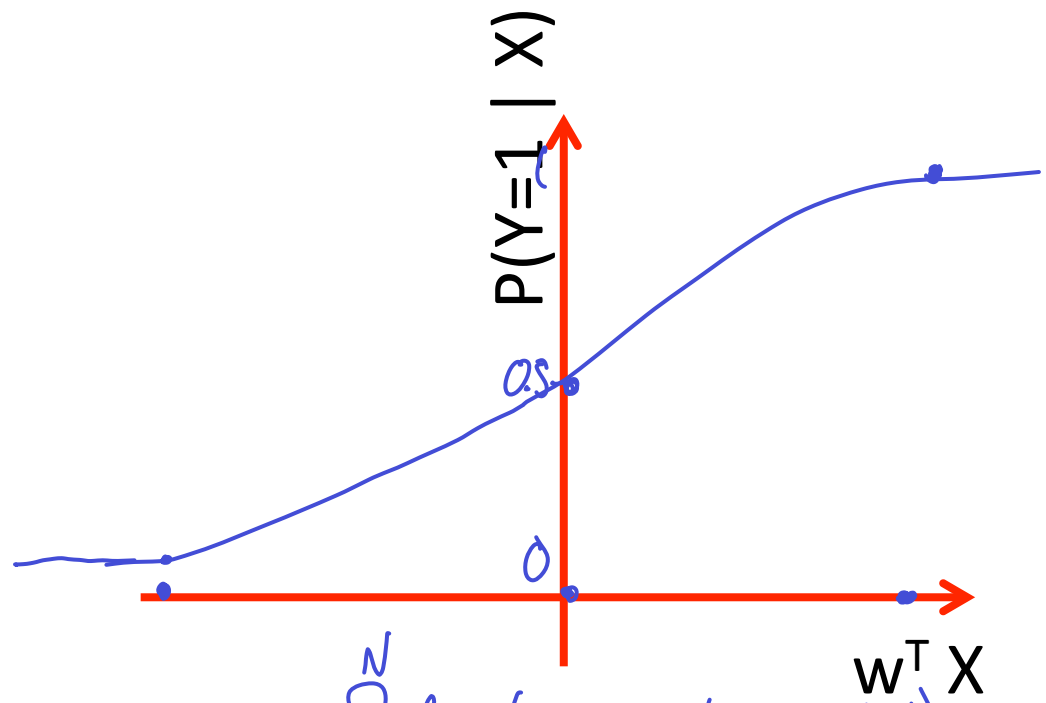
Non-differentiable, non-convex !!

Logistic regression

- Key idea: Predict the probability of a label

$$P(y \mid x, w) = \frac{1}{1+\exp(-y \cdot w^T x)}$$





$$\ln P(D_y \mid D_x, w) = \sum_{i=1}^{N} \ln P(y_i \mid x_i, w) = -\sum_{i=1}^{N} \ln\left(1+\exp(-y_i \cdot w^T x_i)\right)$$

Logistic regression

- Maximize (conditional) likelihood

$$\ln P(D_Y \mid D_X, w) = \sum_{i=1}^{N} \ln P(y_i \mid x_i, w) = -\sum_{i=1}^{N} \log\Big(1 + \exp(-y_i w^T x_i)\Big)$$

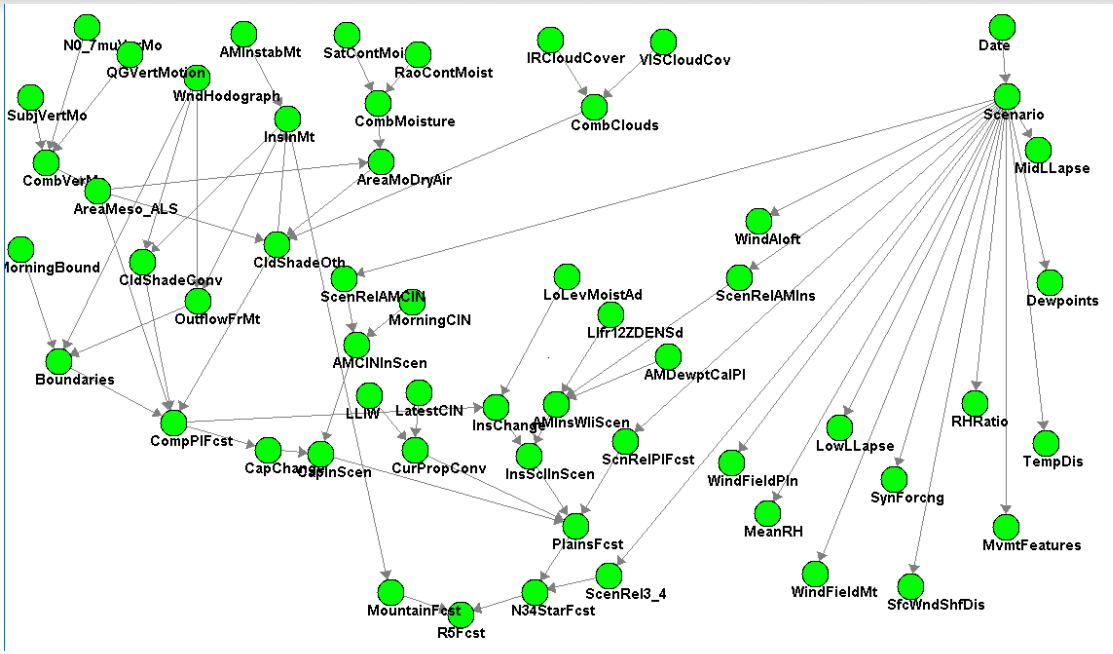
- Convex, differentiable!
- Can find optimal weights w efficiently!

- Can regularize by putting prior on weights w (exactly as in linear regression)

Summary supervised learning

- Want to learn function f from X to Y
- Typically f has parameters w (e.g., slope)
- Want to choose parameters to simultaneously minimize training error and model complexity
- Quantify training error using loss function
  - E.g., squared loss for regression
  - E.g., log-loss for classification
  - Many other loss functions useful (e.g., hinge loss for support vector machines)
- Quantify model complexity using regularizer
- Choose regularization parameter using cross validation

Reminder: Bayesian networks



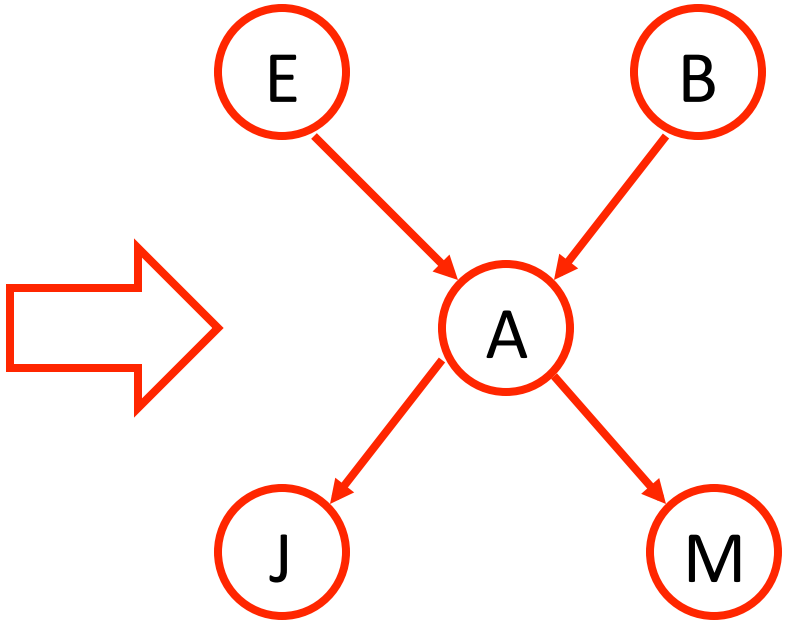
- A **Bayesian network** (G,P) consists of
  - A BN structure G and ..
  - ..a set of conditional probability distributions (CPTs) $P(X_s \mid \mathbf{Pa}_{X_s})$, where $\mathbf{Pa}_{X_s}$ are the parents of node X_s such that
  - (G,P) defines ioint distribution

$$P(X_1, ..., X_n) = \prod_i P(X_i \mid \mathbf{Pa}_{X_i})$$

Next topic: Learning BN from Data

- Two main parts:
  - Learning structure (conditional independencies)
  - Learning parameters (CPDs)

E	B	A	M	J
0	0	0	0	0
1	0	1	1	0
1	0	1	1	1
0	1	1	0	1
0	1	0	0	1
…	…	…	…	…



Parameter learning

- Suppose F is Bernoulli distribution with unknown parameter P(F=cherry) = θ.
- Given training data D = $\{f^{(1)},\ldots,f^{(N)}\}$ (e.g., c c l c c c l l c l c c c..) how do we estimate θ?

$P(F=cherry)$
θ

Want $\theta^* \in \arg\max P(D \mid \theta)$

$$= \arg\max_{\theta} \underbrace{\prod_{i=1}^{N} P(f^{(i)} \mid \theta)}_{= \theta^{n_c} (1-\theta)^{n_\ell}}$$

Maximum likelihood estimation

$$\theta^* \in \underset{\theta}{\operatorname{argmax}} \; \theta^{n_c} (1-\theta)^{n_\ell}$$

$$= \underset{\theta}{\operatorname{argmax}} \; \underbrace{n_c \log \theta + n_\ell \log(1-\theta)}_{\ell(\theta)}$$

$$\frac{d}{d\theta} \ell(\theta) = \frac{n_c}{\theta} - \frac{n_\ell}{1-\theta} \stackrel{!}{=} 0$$

$$\Rightarrow \theta = \frac{n_c}{n_c + n_\ell} = \frac{\text{Count}(F=c)}{N}$$

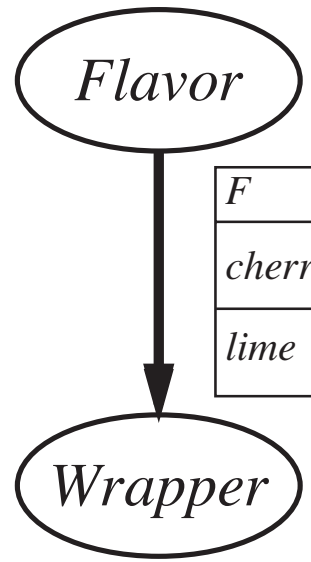
Estimating CPDs

$$P(W=r \mid F=c) = \theta_1$$

$$\theta_1 = \arg\max P(w^{(1)}=r \ldots w^{(N)}=r \mid f^{(1)} \ldots f^{(N)}, \theta_1)$$

$$\theta_1 = \frac{\text{Count}(W=r, F=c)}{\text{Count}(F=c)}$$

$$\theta_{F \mid Pa_F} = [\theta]$$



F	$P(W=red \mid F)$
cherry	θ_1
lime	θ_2

$$\theta_{W \mid Pa_W} = [\theta_1, \theta_2]$$

MLE for general Bayes nets

- Given complete data D = $\{x^{(1)},\ldots,x^{(N)}\}$, want maximum likelihood estimate

$$P(X_1,\ldots,X_n) = \prod_i P(X_i \mid \mathbf{Pa}_{X_i})$$

$$X^{(i)} = [X_1^{(i)} \ldots X_n^{(i)}]$$

$$\theta^* = \arg\max_\theta \underbrace{\log P(D \mid \theta)}_{\ell(\theta)} = \arg\max_\theta \sum_{i=1}^{N} \log P(X^{(i)} \mid \theta)$$

$$= \arg\max_\theta \sum_{i=1}^{N} \sum_{j=1}^{n} \log P(X_j^{(i)} \mid Pa_j^{(i)}, \theta_{j|Pa_j})$$

$$\theta = [\theta_{1|Pa_1}, \theta_{2|Pa_2}, \ldots \theta_{n|Pa_n}]$$

$$\frac{\partial}{\partial \theta_{j|Pa_j}} \ell(\theta) = \frac{\partial}{\partial \theta_{j|Pa_j}} \sum_{i=1}^{N} \log P(X_j^{(i)} \mid Pa_j^{(i)}, \theta_{j|Pa_j})$$

$$\Rightarrow \theta_{j|Pa_j} = \frac{Count(X_j, Pa_j)}{Count(Pa_j)}$$

Algorithm for Bayes net MLE

- Given:
  - Bayes Net structure G
  - Data set D of complete observations
- For each variable X_i estimate

$$\theta_{X_i|\mathbf{Pa}_i} = \frac{Count(X_i, \mathbf{Pa})}{Count(\mathbf{Pa}_i)}$$

- Results in globally optimal maximum likelihood estimate!

Regularizing Bayesian Networks

- With few samples, counts are "imprecise:

$$\theta_{X_i|\mathbf{Pa}_i} = \frac{Count(X_i, \mathbf{Pa})}{Count(\mathbf{Pa}_i)}$$

- Suppose we test only a single candy, and it has cherry flavor.
- What's the MLE?

$$\theta_c = \frac{Count(F=c)}{N} = \frac{1}{1} = 100\%$$

Pseudo-counts

- Make prior assumptions about parameters
- E.g.,: A priori, assume coin to be fair
- Practical approach: Assume we've seen a certain number of heads / tails:

$$\theta_{F=c} = \frac{Count(F=c) + \alpha_c}{N + \alpha_c + \alpha_l}$$

"Pseudocounts"

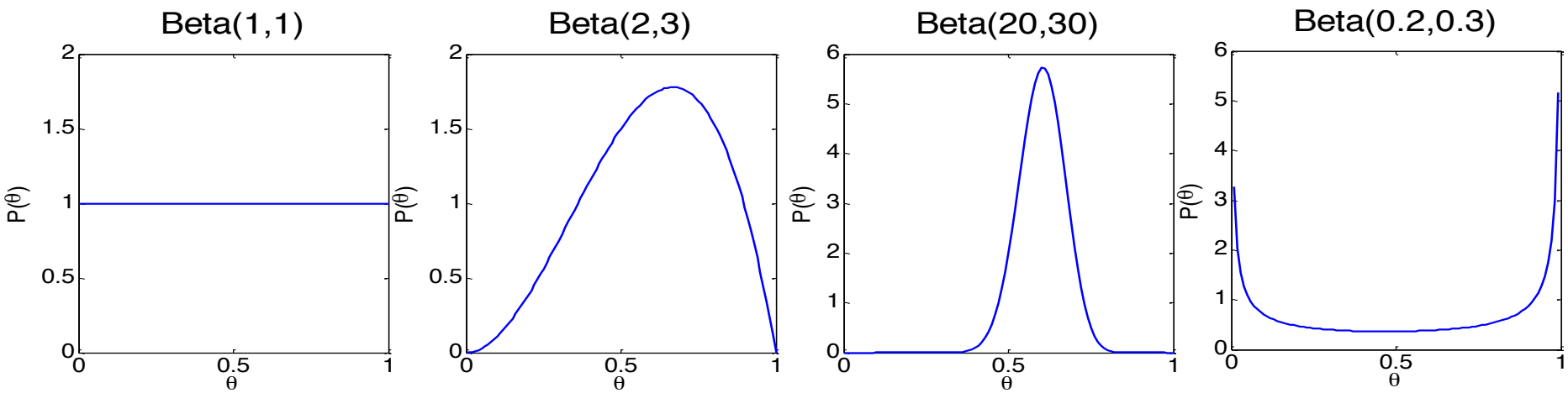
- Looks like a hack.. In fact, this is equivalent to assuming a Beta prior

(Similar to the Gaussian prior for weights in regression)

Beta prior over parameters

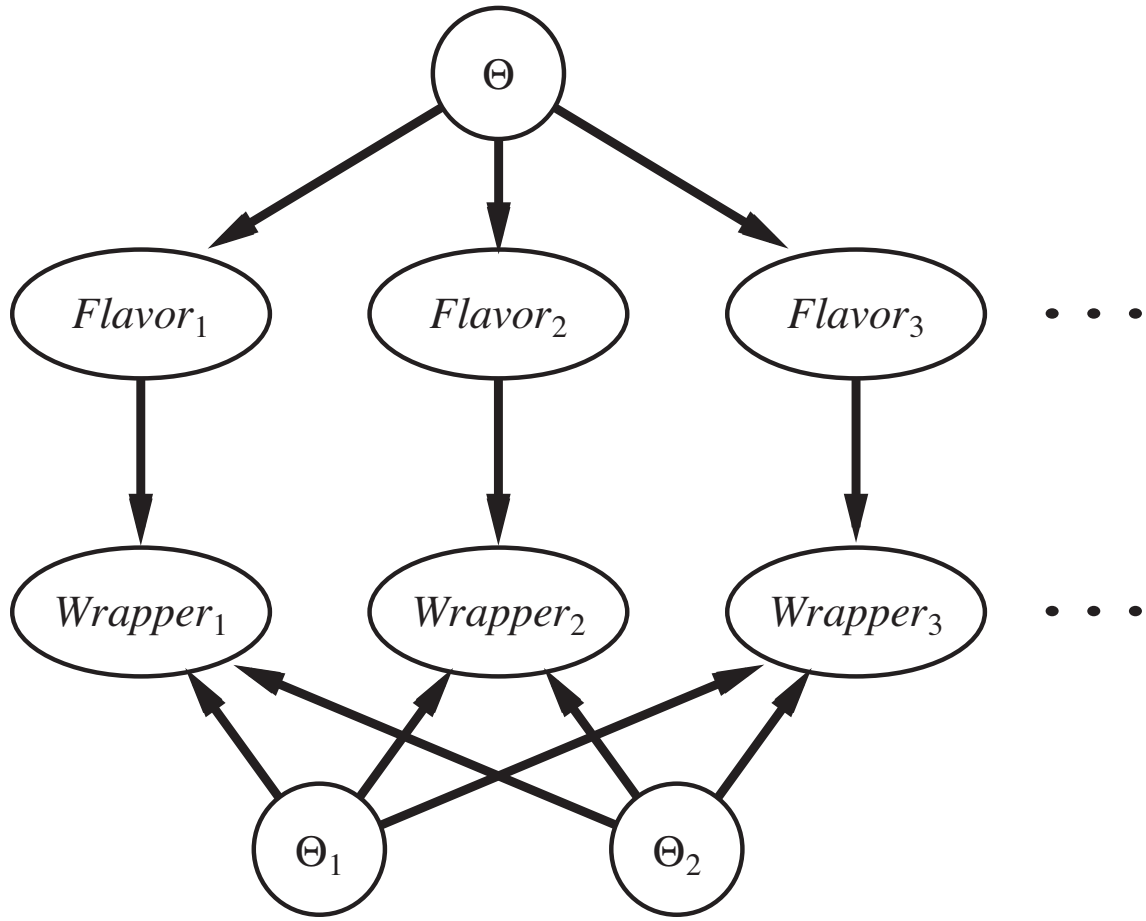
- Beta distribution

$$Beta(\theta; \alpha_c, \alpha_l) = \frac{1}{B(\alpha_c, \alpha_l)} \theta^{\alpha_c - 1} (1-\theta)^{\alpha_l - 1}$$

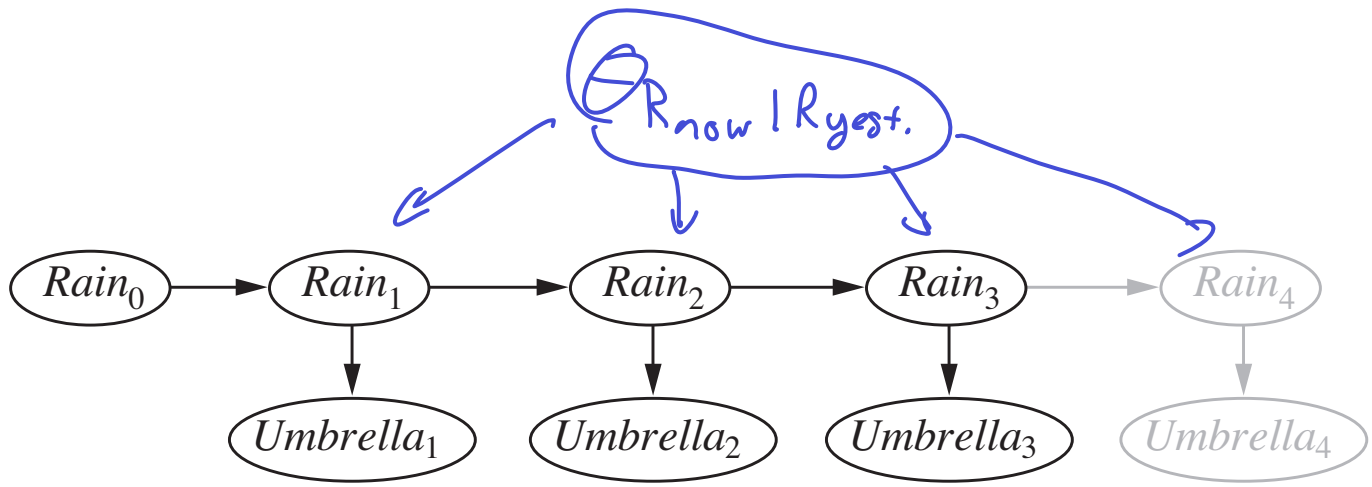


$$\underset{\theta_c}{\arg\max}\ P(\theta_c \mid D) = \frac{n_c + \alpha_c}{n_c + n_\ell + \alpha_c + \alpha_\ell}$$

Bayesian parameter estimation



Learning parameters for dynamical models



$$\Theta_{R_{now}=T \mid R_{yest}=T} = \frac{\sum_{t=1}^{T} I(R_t = T, R_{t-1} = T)}{\sum_{t=1}^{T} I(R_{t-1} = T)}$$

Structure learning

- So far: Discussed how to learn parameters for given Bayes Net structure

- Where does the structure come from?

Score based structure learning

- Define scoring function S(G;D)
  - Quantifies, for each structure G, the fit to the data D
- Search over Bayes net structures G

$$G^* \in \arg\min_G S(G; D)$$

- *Key question*: how should we score a Bayes Net?

MLE score

- *Idea*: For fixed structure G, already know how to calculate MLE for parameters

$$\theta_G \in \arg\max_{\theta} \log P(D \mid \theta, G)$$

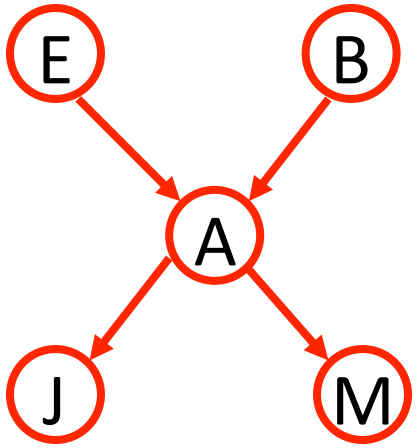
- Can use maximum likelihood of data to score a Bayes Net

$$S(G; D) = \max_{\theta} \log P(D \mid \theta, G)$$

- Turns out, this scoring function has a simple form:

$$\log P(D \mid \theta_G, G) = \cancel{m}\,N \sum_{i=1}^{n} \hat{I}(X_i, \mathbf{Pa}_i) - \underbrace{\cancel{m}\,N \sum_{i=1}^{n} \hat{H}(X_i)}_{\text{Indep. of } G}$$

Example: Likelihood score



$$S(G;D) = \sum_i \hat{I}(X_i ; \mathbf{Pa}_i)$$

E	B	A	M	J
0	0	0	0	0
1	0	1	1	0
1	0	1	1	1
0	1	1	0	1
0	1	0	0	1
…	…	…	…	…

$$= \underbrace{\hat{I}(E;\{\})}_{\hat{H}(E)-\hat{H}(E|\{\})=0} + \hat{I}(B;\{\}) + \hat{I}(A;E,B) + \hat{I}(J;A) + \hat{I}(M;A)$$

Eg. $\hat{I}(J;A) = \hat{H}(J) - \hat{H}(J|A)$; $\hat{H}(J) = -\frac{\text{Count}(J=T)}{N}\log_2\frac{\text{Count}(J=T)}{N} - \frac{\text{Count}(J=F)}{N}\log_2\frac{\text{Count}(J=F)}{N}$

Maximizing the MLE score

- Score of a BN:

$$\arg\max_G S(G;D) = \arg\max_G \sum_{i=1}^{n} \hat{I}(X_i, \mathbf{Pa}_i)$$

- What's the optimal solution to this problem??

$I(X;A) \leq I(X;A,B)$

$\Rightarrow$ adding parents always helps!

$\Rightarrow$ Opt. solution: Complete Network

Finding the optimal MLE structure

- Optimal solution for MLE is always the fully connected graph!!! ☹
  - ➔ Non-compact representation; Overfitting!!

- Solutions:
  - Priors over parameters / structures
  - Constraint optimization (e.g., bound #parents)

Regularizing a BN

- Prefer "simpler" models
- Bayesian Information Criterion (BIC) score:

$$S_{BIC}(G) = \underbrace{\sum_{i=1}^{n} \hat{I}(X_i, \mathbf{Pa}_i)}_{\text{ML-Score}} - \underbrace{\frac{\log N}{2N}|G|}_{\text{Penalty term}}$$

- Hereby
  - $|G|$ is the number of parameters of G
  - n is number of variables
  - N is number of training examples
- **Theorem**: BIC is consistent (will identify correct structure as N → ∞)

Finding the best structure

- Finding the BN that maximizes the BIC score is NP hard ☹
- Instead: use search techniques (see first part of this course)
  - Start with no edges
  - Add or remove edges, while preserving acyclicity

- However: Can solve one interesting special case optimally:

Finding the optimal tree BN

- Scoring function

$$\mathrm{Score}(\mathcal{G};\mathcal{D}) = \sum_i \widehat{I}(X_i, \mathbf{Pa}_i)$$

- Scoring a tree

Finding the optimal tree

- Given graph G = (V,E), and nonnegative weights w_e for each edge $e=(X_i,X_j)$
  - In our case: $w_e = I(X_i,X_j)$

- Want to find tree T that maximizes $\sum_e w_e$
- Maximum spanning tree problem!
- Can solve in time O(|E| log |E|)!

Chow-Liu algorithm

- For each pair X_i, X_j of variables compute

$$\widehat{P}(x_i, x_j) = \frac{\mathrm{Count}(x_i, x_j)}{m}$$

- Compute mutual information

$$\widehat{I}(X_i, X_j) = \sum_{x_i, x_j} \widehat{P}(x_i, x_j) \log \frac{\widehat{P}(x_i, x_j)}{\widehat{P}(x_i)\widehat{P}(x_j)}$$

- Define complete graph with weight of edge (X_i,X_i) given by the mutual information
- Find maximum spanning tree ➔ undirected tree
- Orient the edges using breadth-first search

Example

E	B	A	M	J
0	0	0	0	0
1	0	1	1	0
1	0	1	1	1
0	1	1	0	1
0	1	0	0	1
…	…	…	…	…

Summary

- To learn a Bayes net, need to
  - Learn structure
  - Learn parameters
- If all variables are observed
  - Get maximum likelihood parameter estimate by counting
  - Use pseudo-counts (Beta prior) to avoid overfitting
- Search for structure by maximizing score function
  - Score = Likelihood for best choice of parameters
- Can find optimal trees efficiently!