Artificial Intelligence

Lecture 9：Learning for PGM

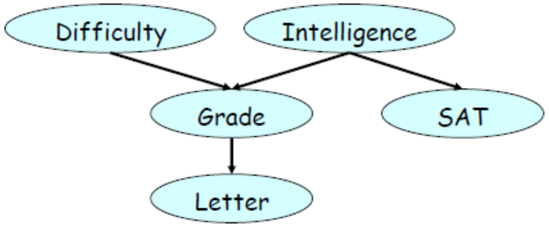
Xiaojin Gong

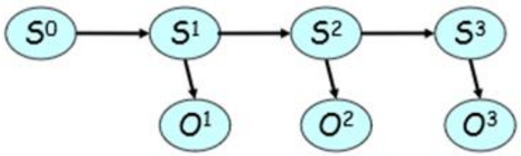
2022-04-18

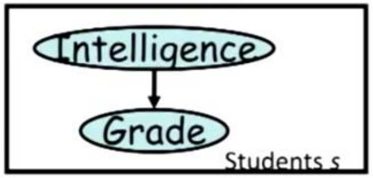
Credits: PGM course in Stanford, by D. Koller

Review

- Bayesian network
  - Representation
    - Bayesian network
    - Template models
  - Inference
    - Exact inference
      - Enumeration
      - Variable elimination
    - Approximate inference
      - Sampling
  - Applications
    - Kalman filtering
    - Particle filtering
    - Object tracking
    - Scene parsing

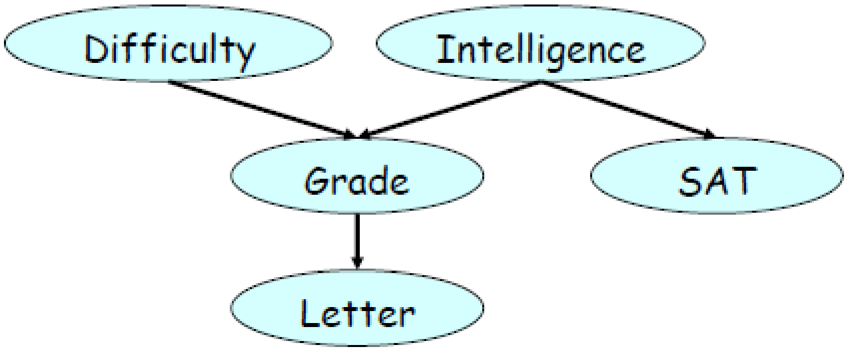






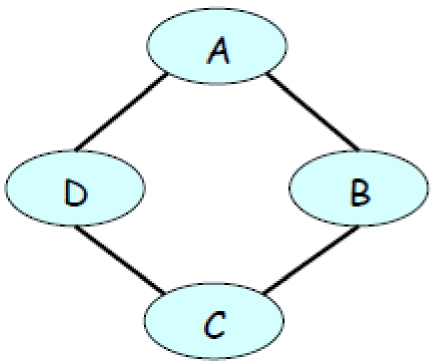
Probabilistic Graphical Models

Bayesian network



Causality relationship

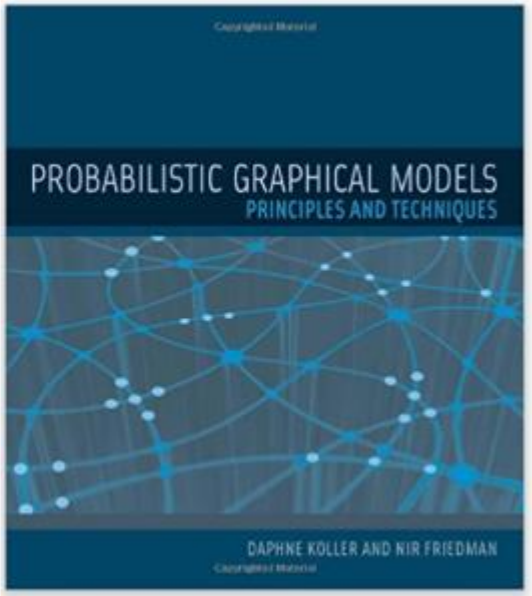
Markov network



Correlation

Markov Network

- Markov network
  - Representation
  - Inference
    - Exact inference
      - Belief propagation
      - Max-Sum elimination
    - Approximate inference
      - Loopy belief propagation
      - Variational methods

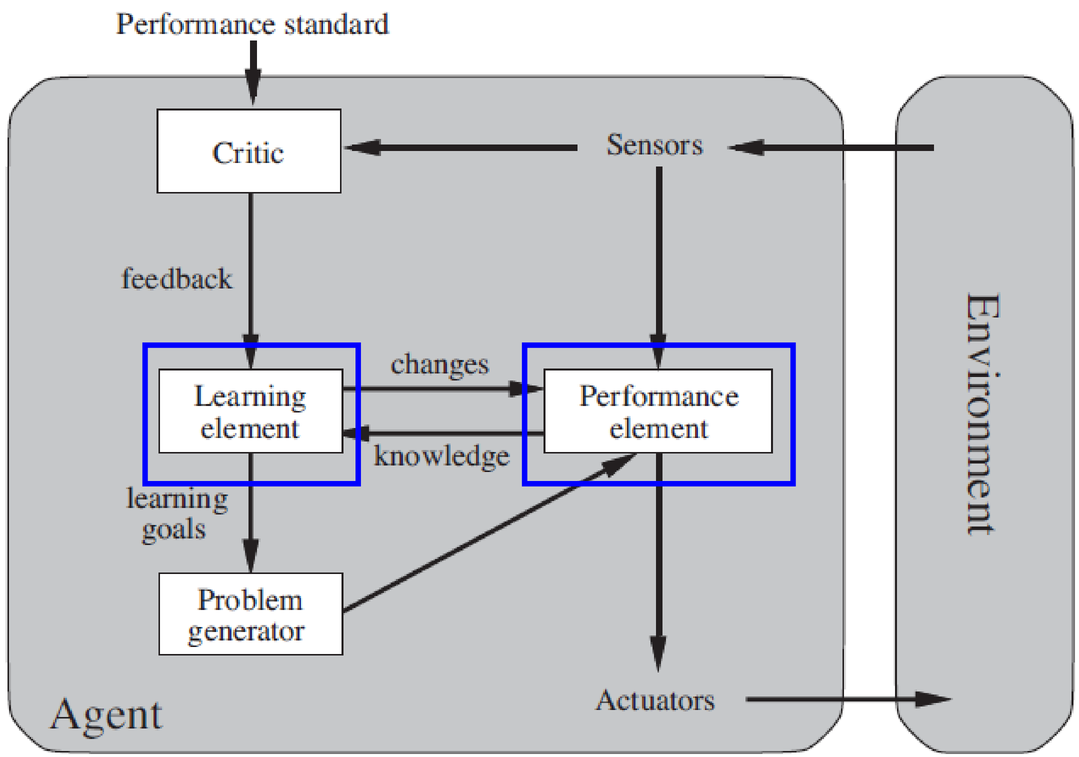


Outline

- Learning for Probabilistic Graphical Models
  - Parameter Estimation
    - Maximum Likelihood Estimation
    - Bayesian Estimation
    - Application: Supervised Classification
  - Structure Learning
    - Learning Metrics
  - Learning with Hidden variables
    - Expectation Maximization
    - Application: Unsupervised Clustering

Learning Agents

- An agent is learning if it improves its performance on future tasks after making observations about the world.

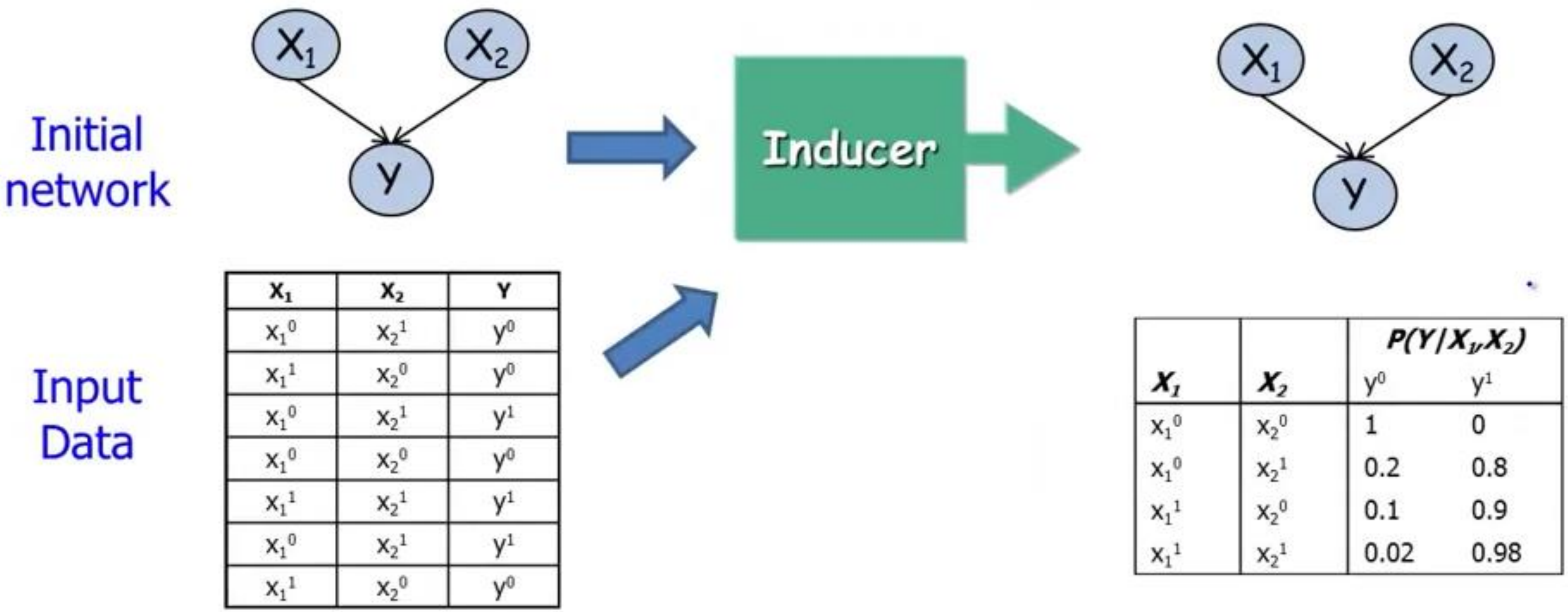


Learning Agents

- The improvements depend on four major factors:
  - Which component is to be improved.
  - What prior knowledge the agent already has.
  - What representation is used for the data and the component.
    - Probabilistic graphical models
  - What feedback is available to learn from.
    - Supervised learning
    - Unsupervised learning
    - Reinforcement learning

Learning Scenarios

- Known structure, complete data

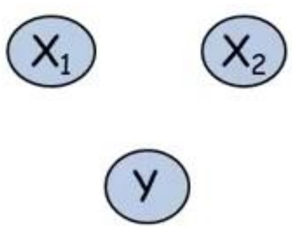


X_1	X_2	Y
x_1^0	x_2^1	y^0
x_1^1	x_2^0	y^0
x_1^0	x_2^1	y^1
x_1^0	x_2^0	y^0
x_1^1	x_2^1	y^1
x_1^0	x_2^1	y^1
x_1^1	x_2^0	y^0

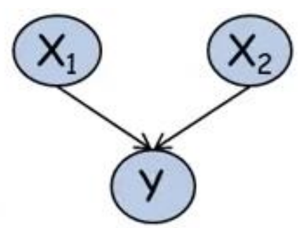
		$P(Y/X_1,X_2)$	
X_1	X_2	y^0	y^1
x_1^0	x_2^0	1	0
x_1^0	x_2^1	0.2	0.8
x_1^1	x_2^0	0.1	0.9
x_1^1	x_2^1	0.02	0.98

Learning Scenarios

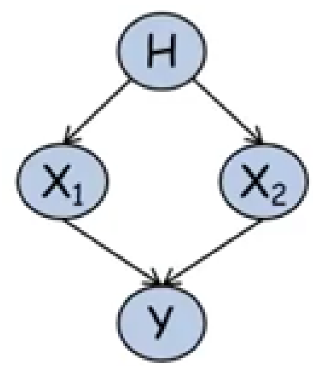
- Known structure, complete data
- Unknown structure, complete data
- Known structure, incomplete data
- Unknown structure, incomplete data
- Hidden variables, incomplete data



X_1	X_2	Y
x_1^0	x_2^1	y^0
x_1^1	x_2^0	y^0
x_1^0	x_2^1	y^1
x_1^0	x_2^0	y^0
x_1^1	x_2^1	y^1
x_1^0	x_2^1	y^1
x_1^1	x_2^0	y^0



X_1	X_2	Y
?	x_2^1	y^0
x_1^1	?	y^0
?	x_2^1	?
x_1^0	x_2^0	y^0
?	x_2^1	y^1
x_1^0	x_2^1	?
x_1^1	?	y^0



Learning Tasks

- True distribution P^*, corresponding to a PGM $\mathcal{M}^* = (\mathcal{K}^*, \theta^*)$
- Learning task:
  - Given a data set $\mathcal{D} = \{\xi[1], \ldots, \xi[M]\}$ of M i.i.d samples from P^*
    - Complete / incomplete data
  - To learn a model $\widetilde{\mathcal{M}}$ that best approximates $\mathcal{M}^*$
    - Learn θ^* : Parameter estimation
    - Learn $\mathcal{K}^*$: Structure learning
  - Incorporate prior knowledge or constraints about $\widetilde{\mathcal{M}}$
    - Priors over parameters
    - Priors over structures

Parameter Estimation

- Density estimation:
  - Minimize relative entropy distance / KL-divergence

$$\boldsymbol{D}(P^* \| \tilde{P}) = \boldsymbol{E}_{\xi \sim P^*}\left[\log\left(\frac{P^*(\xi)}{\tilde{P}(\xi)}\right)\right] = -\boldsymbol{H}_{P^*}(\mathcal{X}) - \boxed{\boldsymbol{E}_{\xi \sim P^*}\left[\log \tilde{P}(\xi)\right]}$$

Expected log-likelihood

- Learning Metrics:
  - Likelihood

$$P(\mathcal{D} : \mathcal{M}) = \prod_{m=1}^{M} P(\xi[m] : \mathcal{M})$$

  - Log-likelihood

$$\log P(\mathcal{D} : \mathcal{M}) = \sum_{m=1}^{M} \log P(\xi[m] : \mathcal{M})$$

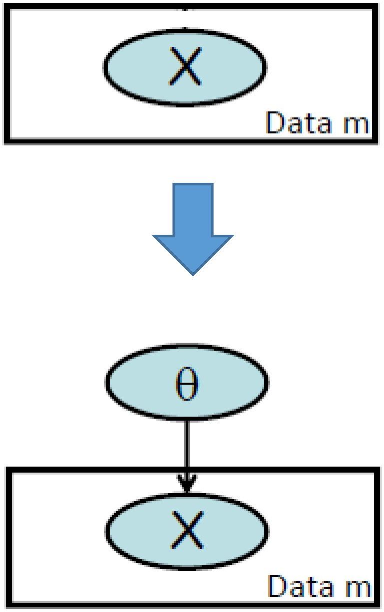
  - Conditional likelihood

$$P(\mathcal{D}: \mathcal{M}) = \prod P(y[m] | x[m]: \mathcal{M})$$

Maximum Likelihood Estimation

- Biased Coin Example
  - P is a Bernoulli distribution

$$P(x[m]) = \begin{cases} \theta & x[m] = x^1 \\ 1-\theta & x[m] = x^0 \end{cases}$$



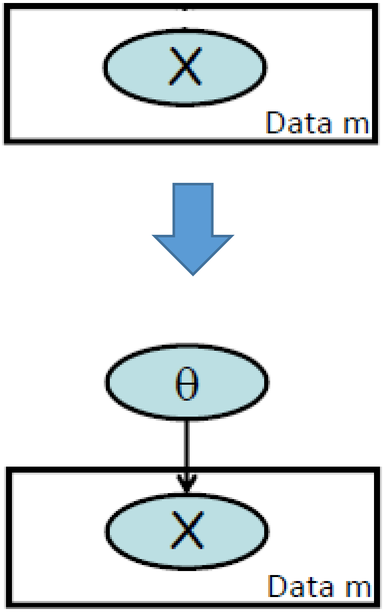
  - $\mathcal{D} = \{x[1], \ldots, x[M]\}$ sampled i.i.d from P
  - Goal: find $\theta \in [0,1]$ that predicts $\mathcal{D}$ well
  - Metric: likelihood of $\mathcal{D}$ given θ

$$L(\theta : \mathcal{D}) = \prod_m P(x[m] : \theta)$$

θ is not a random variable

Maximum Likelihood Estimation

- Biased Coin Example (cont'd)



  - Find θ maximizing likelihood

$$L(\theta : \mathcal{D}) = \theta^{M[1]}(1-\theta)^{M[0]}$$

  - Equivalent to maximizing log-likelihood

$$\ell(\theta : \mathcal{D}) = M[1]\log\theta + M[0]\log(1-\theta)$$

  - Differentiating the log-likelihood and solving for θ

$$\hat{\theta} = \frac{M[1]}{M[1]+M[0]}$$

Maximum Likelihood Estimation

- P is a Multinomial distribution

$$P(x:\boldsymbol{\theta}) = \theta_k \qquad \text{if } x = x^k \qquad \boldsymbol{\theta} \in [0,1]^K : \sum_i \theta_i = 1$$

  - Likelihood

$$L(\mathcal{D}:\boldsymbol{\theta}) = \prod_k \theta_k^{M[k]}$$

  - Constrained cost function with a Lagrange multiplier

$$l = \log L(D:\theta) + \lambda(1 - \sum_k \theta_k)$$

$$\Rightarrow \quad \frac{\partial l}{\partial \theta_k} = \frac{M[k]}{\theta_k} - \lambda = 0 \qquad \Rightarrow \quad \hat{\theta}_k = \frac{M[k]}{M}$$

$$\sum M[k] = \sum \lambda\theta_k = \lambda$$

Maximum Likelihood Estimation

- P is a Gaussian distribution

$$P(x:\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

  - Log likelihood

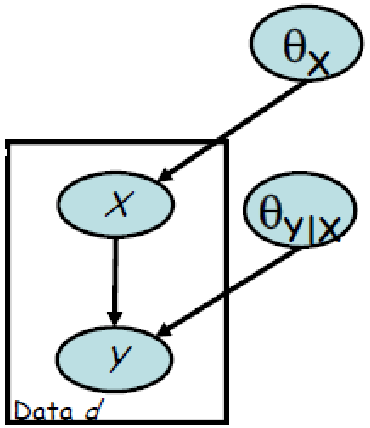
$$L = \sum_{j=1}^{N} \log \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x_j-\mu)^2}{2\sigma^2}} = N(-\log\sqrt{2\pi} - \log\sigma) - \sum_{j=1}^{N} \frac{(x_j-\mu)^2}{2\sigma^2}$$

  - Differentiating the log-likelihood and solving

$$\begin{aligned} \frac{\partial L}{\partial \mu} &= -\frac{1}{\sigma^2}\sum_{j=1}^{N}(x_j-\mu) = 0 & \Rightarrow \quad \mu = \frac{\sum_j x_j}{N} \\ \frac{\partial L}{\partial \sigma} &= -\frac{N}{\sigma} + \frac{1}{\sigma^3}\sum_{j=1}^{N}(x_j-\mu)^2 = 0 & \Rightarrow \quad \sigma = \sqrt{\frac{\sum_j (x_j-\mu)^2}{N}} \end{aligned}$$

MLE for Bayesian Network

- Likelihood for BN



$$
\begin{aligned}
L(\boldsymbol{\theta} : \mathcal{D}) &= \prod_m P_{\mathcal{G}}(\xi[m] : \boldsymbol{\theta}) \\
&= \prod_m \prod_i P(x_i[m] \mid \mathrm{pa}_{X_i}[m] : \boldsymbol{\theta}) \\
&= \prod_i \boxed{\left[\prod_m P(x_i[m] \mid \mathrm{pa}_{X_i}[m] : \boldsymbol{\theta}_{X_i \mid \mathrm{Pa}_{X_i}}) \right]}
\end{aligned}
$$

$$||$$

$$L_i(\boldsymbol{\theta}_{X_i \mid \mathrm{Pa}_{X_i}} : \mathcal{D})$$

  - If the parameter sets $\theta_{X_i \mid \mathrm{Pa}_{X_i}}$ are disjoint, MLE can be computed by maximizing each local likelihood separately

MLE for Bayesian Network

- Example
  - Given data instance (*x[m]*, *y[m]*)
  - Estimate parameters

$$\theta_{x^0}, \theta_{x^1}, \theta_{y^0|x^0}, \theta_{y^1|x^0}, \theta_{y^0|x^1}, \theta_{y^1|x^1}$$

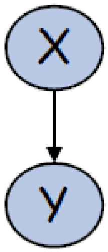
  - Likelihood:

$$L(\boldsymbol{\theta} : \mathcal{D}) = \prod_{m=1}^{M} P(x[m], y[m] : \boldsymbol{\theta})$$

$$= \prod_{m} P(x[m] : \boldsymbol{\theta}) P(y[m] \mid x[m] : \boldsymbol{\theta})$$

$$= \left(\prod_{m} P(x[m] : \boldsymbol{\theta}_X) \right) \left(\prod_{m} P(y[m] \mid x[m] : \boldsymbol{\theta}_{Y|X}) \right)$$

X	
x^0	x^1
0.7	0.3

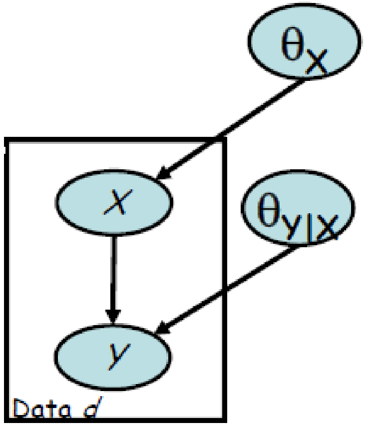


	Y	
X	y^0	y^1
x^0	0.95	0.05
x^1	0.2	0.8

MLE for Bayesian Network

- Example (cont'd)
  - Likelihood:

$$L(\boldsymbol{\theta} : \mathcal{D}) = \left(\prod_m P(x[m] : \boldsymbol{\theta}_X)\right) \boxed{\left(\prod_m P(y[m] \mid x[m] : \boldsymbol{\theta}_{Y|X})\right)}$$



$$\Rightarrow \prod_m P(y[m] \mid x[m] : \boldsymbol{\theta}_{Y|X})$$

$$= \boxed{\prod_{m:x[m]=x^0} P(y[m] \mid x[m] : \boldsymbol{\theta}_{Y|x^0})} \cdot \prod_{m:x[m]=x^1} P(y[m] \mid x[m] : \boldsymbol{\theta}_{Y|x^1})$$

$$\Rightarrow \prod_{m:x[m]=x^0} P(y[m] \mid x[m] : \boldsymbol{\theta}_{Y|x^0}) = \theta_{y^1|x^0}^{M[x^0,y^1]} \cdot \theta_{y^0|x^0}^{M[x^0,y^0]}$$

$$\Rightarrow \theta_{y^1|x^0} = \frac{M[x^0,y^1]}{M[x^0,y^1] + M[x^0,y^0]} = \frac{M[x^0,y^1]}{M[x^0]}$$

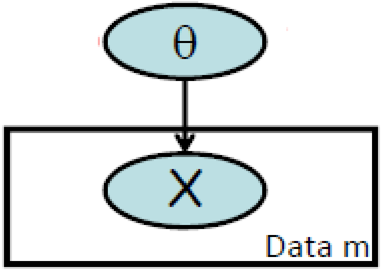
Learning Approaches

- Maximum likelihood estimation
  - Limit: do not encode any prior knowledge & deficiencies with small data sets
    - A coin is tossed 10 times, and comes out ‘heads’ 7 of the 10 tosses
    - A coin is tossed 10000 times, and comes out ‘heads’ 7000 of the 10000 tosses
- Bayesian estimation
  - Basic idea: to encode prior knowledge
    - Treat parameters as random variables
    - Learning is then a special case of inference

Bayesian Estimation

- Bayesian prediction:
  - Calculates the probability of each hypothesis, given the data, and make predictions on that basis
  - The predictions are made by using all the hypotheses, weighted by their probabilities, rather than by using just a single best hypothesis
  - Learning is reduced to probabilistic inference

Bayesian Estimation



- Biased Coin Example
  - Treat parameter θ as random variables
    - Given a fixed θ, tosses are independent
    - If θ is unknown, tosses are not marginally independent
  - Joint probability

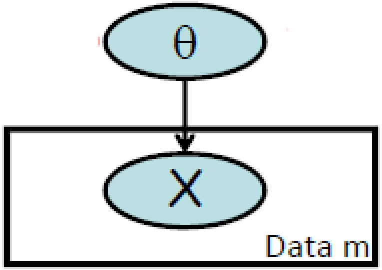
$$
\begin{aligned}
P(x[1], \ldots, x[M], \theta) &= \underbrace{\boxed{P(x[1], \ldots, x[M] \mid \theta)}}_{\text{Likelihood}} \underbrace{\boxed{P(\theta)}}_{\text{Prior}} \\
&= P(\theta) \prod_{m=1}^{M} P(x[m] \mid \theta) \\
&= P(\theta) \theta^{M[1]} (1-\theta)^{M[0]}
\end{aligned}
$$

  - The posterior distribution over θ

$$
P(\theta \mid x[1], \ldots, x[M]) = \frac{P(x[1], \ldots, x[M] \mid \theta) P(\theta)}{P(x[1], \ldots, x[M])}
$$

Bayesian Estimation

- Biased Coin Example
  - Bayesian prediction:
    - Predict the probability over the next toss



$$P(x[M+1] \mid x[1], \ldots, x[M]) =$$

$$= \int P(x[M+1] \mid \theta, x[1], \ldots, x[M]) P(\theta \mid x[1], \ldots, x[M]) d\theta$$

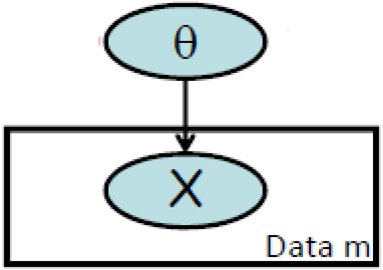
$$= \int P(x[M+1] \mid \theta) P(\theta \mid x[1], \ldots, x[M]) d\theta$$

$$= \int P(x[M+1] \mid \theta) \frac{P(x[1], \ldots, x[M] \mid \theta) \boxed{P(\theta)}}{P(x[1], \ldots, x[M])} d\theta$$

- The probability depends on the prior distribution
  - Uniform prior
  - Dirichlet prior

Bayesian Estimation

- Biased Coin Example
  - Bayesian prediction with uniform prior

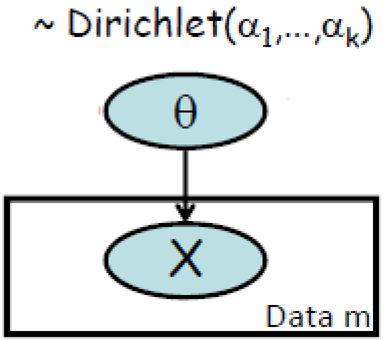


$$P(x[M+1] \mid x[1], \ldots, x[M])$$

$$= \int P(x[M+1] \mid \theta) \frac{P(x[1], \ldots, x[M] \mid \theta) P(\theta)}{P(x[1], \ldots, x[M])} \, d\theta$$

$$\boxed{\begin{aligned} & P(X[M+1] = x^1 \mid x[1], \ldots, x[M]) \\ & = \frac{1}{P(x[1], \ldots, x[M])} \int \theta \cdot \theta^{M[1]} (1-\theta)^{M[0]} d\theta \\ & = \frac{M[1]+1}{M[1]+M[0]+2} \end{aligned}}$$

Bayesian Estimation



- Biased Coin Example
  - Bayesian prediction with Dirichlet prior

$$P(x[M+1] \mid x[1], \ldots, x[M])$$

$$= \int P(x[M+1] \mid \theta) \frac{P(x[1], \ldots, x[M] \mid \theta) P(\theta)}{P(x[1], \ldots, x[M])} d\theta$$

$$\boxed{P(x[M+1] = x^k \mid \mathcal{D}) = \frac{M[k] + \alpha_k}{M + \alpha}}$$

  - Larger $\alpha \Rightarrow$ more confidence in prior

Bayesian Estimation

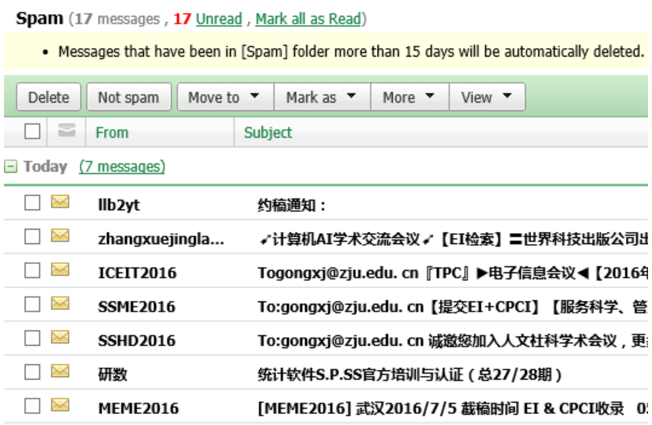
- Bayesian prediction:
  - Pros:
    - Bayesian prediction is optimal, whether the data set be small or large
  - Cons:
    - For real learning problems, the hypothesis space is usually very large or infinite, we must resort to approximate or simplified methods such as MAP
- Maximum a posteriori (MAP)

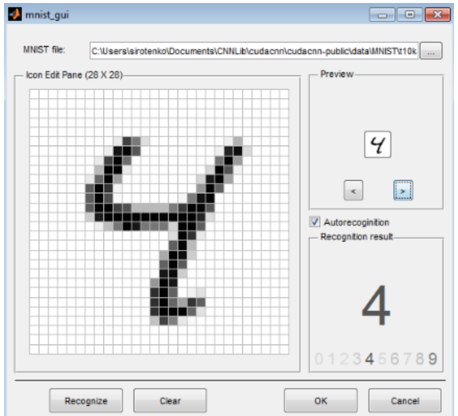
$$P(\theta \mid x[1], \ldots, x[M]) = \frac{P(x[1], \ldots, x[M] \mid \theta) P(\theta)}{P(x[1], \ldots, x[M])}$$

  - When the prior is a uniform distribution, MAP reduces to MLE

Application: Supervised Classification

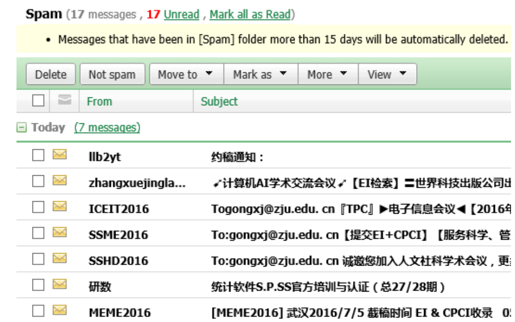
- Model-based classification with Naive Bayes





Example: Spam Filter

- Input: an email
- Output: spam / ham



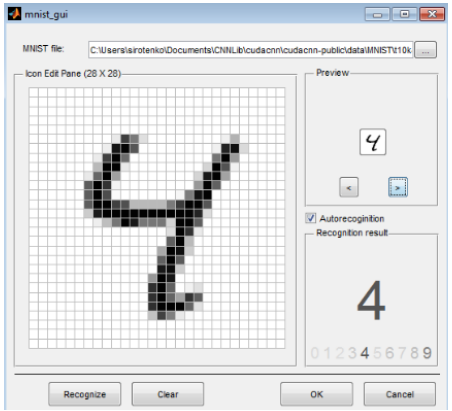
- Setup:
  - Get a large collection of example emails, each labeled “spam” or “ham”
  - Note: someone has to hand label all this data!
  - Want to learn to predict labels of new, future emails

- Features: The attributes used to make the spam/ham decision
  - Words: FREE!
  - Text Patterns: $dd, CAPS
  - Non-text: SenderInContacts
  - …

Example: Digit Recognition

- Input: images / pixel grids
- Output: a digit 0-9

MNIST database



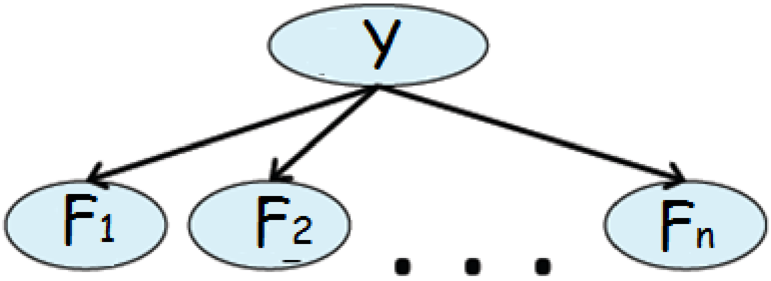
- Setup:
  - Get a large collection of example images, each labeled with a digit
  - Note: someone has to hand label all this data!
  - Want to learn to predict labels of new, future digit images

- Features: The attributes used to make the digit decision
  - Pixels: (6,8)=ON
  - Shape Patterns: NumComponents, AspectRatio, NumLoops
  - …

Model-based Classification

- Model-based approach
  - Build a model (e.g. Bayes' net) where both the label and features are random variables
  - Instantiate any observed features
  - Query for the distribution of the label conditioned on the features
- Challenges
  - What structure should the BN have?
  - How should we learn its parameters?

Naive Bayes Model



- Independence:

$$\forall F_i, F_j, \qquad \left(F_i \perp F_j \middle| Y\right)$$

- Joint distribution:

$$P(Y, F_1, \dots, F_n) = P(Y) \prod_i P(F_i|Y)$$

- Classifier:

$$\max P\left(Y = y_j \middle| f_1, \cdots, f_n\right) = \max \frac{P(Y=y_j) \prod_i P(f_i|Y=y_j)}{P(f_1, \dots, f_n)}$$

Naive Bayes for Digits

- Assume all features are independent effects of the label

- Simple digit recognition version:
  - One feature (variable) F_{ij} for each grid position $< i, j >$
  - Feature values are on / off, based on whether intensity is more or less than 0.5 in underlying image
  - Each input maps to a feature vector, e.g.

$$\rightarrow \langle F_{0,0} = 0 \;\; F_{0,1} = 0 \;\; F_{0,2} = 1 \;\; F_{0,3} = 1 \;\; F_{0,4} = 0 \;\; \ldots F_{15,15} = 0 \rangle$$

  - Here: lots of features, each is binary valued

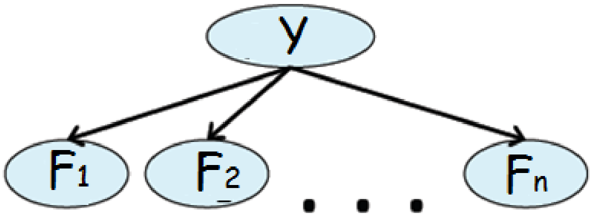
- Naïve Bayes model: $P(Y, F_1, \ldots, F_n) \propto P(Y) \prod_i P(F_i | Y)$

- What do we need to learn?

Naive Bayes for Digits

- A general Naive Bayes model:

$$P(Y, F_1, \ldots, F_n) \propto P(Y) \prod_i P(F_i|Y)$$



|Y| x $|F|^n$ values

n x |F| x |Y| parameters

- We only have to specify how each feature depends on the class
- Total number of parameters is linear in n
- Model is very simplistic, but often works anyway

Naive Bayes for Digits

- Inference:
  - Goal: compute posterior distribution over label variable Y
    - Step 1: get joint probability of label and evidence for each label

$$P(Y, f_1 \ldots f_n) = \begin{bmatrix} P(y_1, f_1 \ldots f_n) \\ P(y_2, f_1 \ldots f_n) \\ \vdots \\ P(y_k, f_1 \ldots f_n) \end{bmatrix} \Rightarrow \begin{bmatrix} P(y_1) \prod_i P(f_i|y_1) \\ P(y_2) \prod_i P(f_i|y_2) \\ \vdots \\ P(y_k) \prod_i P(f_i|y_k) \end{bmatrix}$$

$$\overline{} \; +$$

$$P(f_1 \ldots f_n)$$

    - Step 2: sum to get probability of evidence

    - Step 3: normalize by dividing Step 1 by Step 2

$$\Downarrow$$

$$P(Y|f_1 \ldots f_n)$$

Naive Bayes for Digits

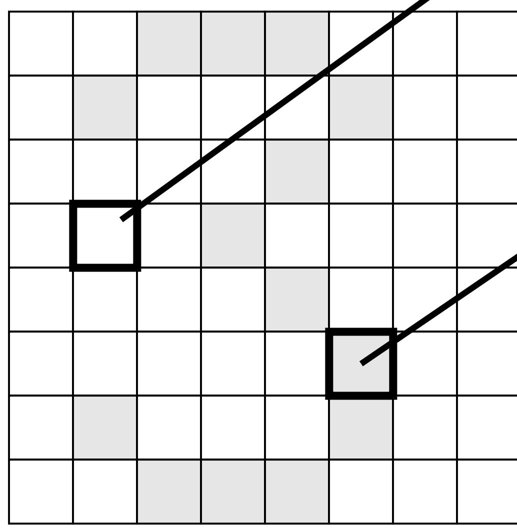
- What do we need in order to use Naïve Bayes?
  - Inference method
    - Start with a bunch of probabilities: $P(Y)$ and the $P(F_i|Y)$ tables
    - Use standard inference to compute $P(Y|F_1 \ldots F_n)$
    - Nothing new here
  - Estimates of local conditional probability tables
    - $P(Y)$, the prior over labels
    - $P(F_i|Y)$ for each feature (evidence variable)
    - These probabilities are collectively called the parameters of the model and denoted by θ
    - They typically come from training data counts

Naive Bayes for Digits

- Conditional probability tables

$P(Y)$

1	0.1
2	0.1
3	0.1
4	0.1
5	0.1
6	0.1
7	0.1
8	0.1
9	0.1
0	0.1



$P(F_{3,1} = on|Y)$

1	0.01
2	0.05
3	0.05
4	0.30
5	0.80
6	0.90
7	0.05
8	0.60
9	0.50
0	0.80

$P(F_{5,5} = on|Y)$

1	0.05
2	0.01
3	0.90
4	0.80
5	0.90
6	0.90
7	0.25
8	0.85
9	0.60
0	0.80

Training and Testing

- Data: labeled instances
  - Training set
  - Validation set
  - Test set

- Features: attribute-value pairs which characterize each x

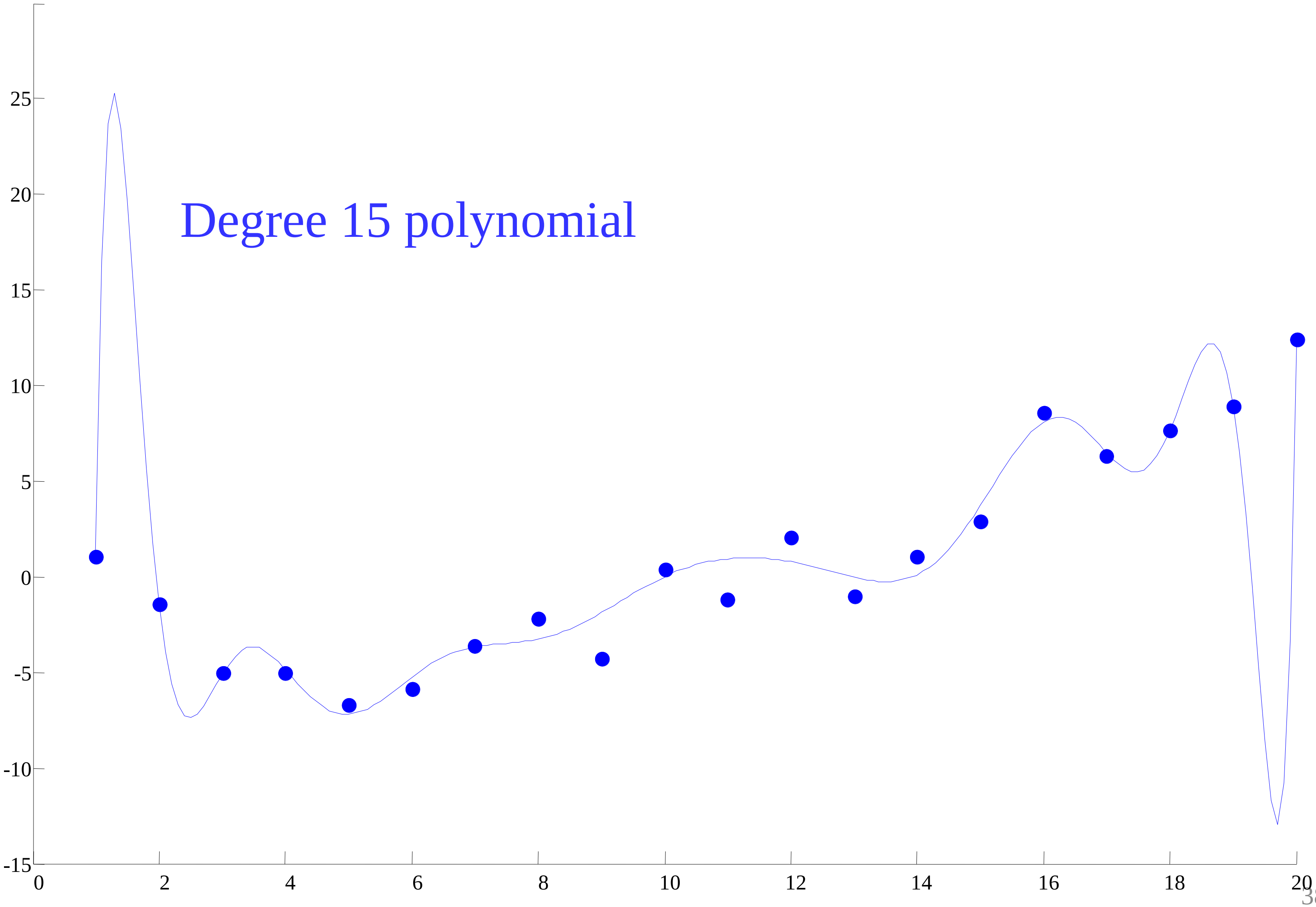
- Experimentation cycle
  - Learn parameters (e.g. model probabilities) on training set
  - (Tune hyperparameters on validation set)
  - Compute accuracy of test set
  - Very important: never "peek" at the test set!

- Evaluation
  - Accuracy: fraction of instances predicted correctly

Generalization and Overfitting

- Generalization:
  - Want a classifier which does well on test data
- Overfitting:
  - Fitting the training data very closely, but not generalizing well

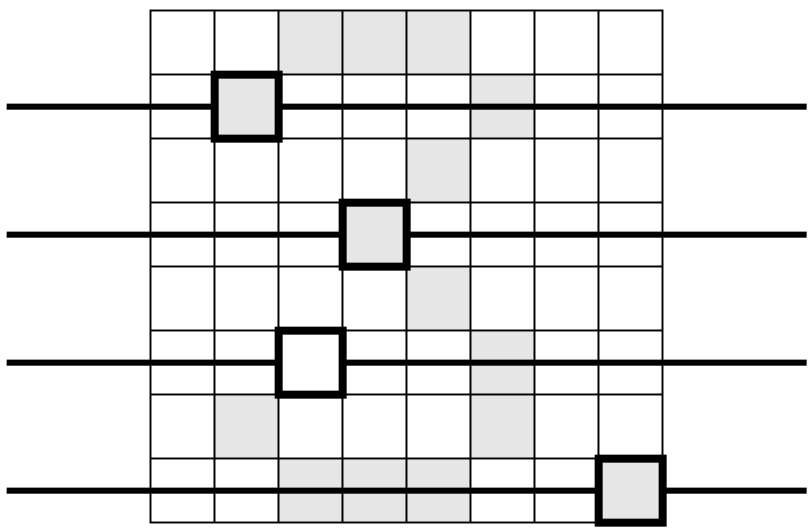
Example: Overfitting



Example: Overfitting

$P(\text{features}, C = 2)$ $\qquad$ $P(\text{features}, C = 3)$

$P(C = 2) = 0.1$ $\qquad$ $P(C = 3) = 0.1$



$P(\text{on}\mid C = 2) = 0.8$	$P(\text{on}\mid C = 3) = 0.8$
$P(\text{on}\mid C = 2) = 0.1$	$P(\text{on}\mid C = 3) = 0.9$
$P(\text{off}\mid C = 2) = 0.1$	$P(\text{off}\mid C = 3) = 0.7$
$P(\text{on}\mid C = 2) = 0.01$	$P(\text{on}\mid C = 3) = 0.0$

2 wins!!

What went wrong here?

Generalization and Overfitting

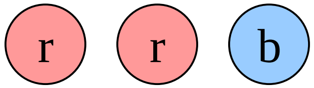
- Relative frequency parameters will overfit the training data!
  - Just because we never saw a 3 with pixel (15,15) on during training doesn't mean we won't see it at test time
  - In general, we can't go around giving unseen events zero probability

- To generalize better:
  - we need to smooth or regularize the estimates

Parameter Estimation

- Estimating the distribution of a random variable
- Empirically: use training data (learning!)
  - E.g.: for each outcome x, look at the empirical rate of that value:

$$P_{\mathsf{ML}}(x) = \frac{\mathsf{count}(x)}{\mathsf{total\ samples}}$$



$$P_{\mathsf{ML}}(\mathsf{r}) = 2/3$$

  - This is the estimate that maximizes the likelihood of the data

$$L(x, \theta) = \prod_i P_\theta(x_i)$$

Maximum Likelihood Estimation

- Relative frequencies are the MLE

$$\theta_{ML} = \arg\max_{\theta} P(\mathbf{X}|\theta) = \arg\max_{\theta} \prod_i P_{\theta}(X_i)$$

$$P_{\mathsf{ML}}(x) = \frac{\mathsf{count}(x)}{\mathsf{total\ samples}}$$

- Another option is to consider the most likely parameter value given the data - MAP

$$\theta_{MAP} = \arg\max_{\theta} P(\theta|\mathbf{X})$$

$$= \arg\max_{\theta} P(\mathbf{X}|\theta)P(\theta)/P(\mathbf{X})$$

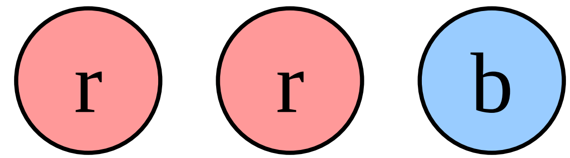
$$= \arg\max_{\theta} P(\mathbf{X}|\theta)P(\theta)$$

Laplace Smoothing

- Laplace's estimate:
  - Pretend you saw every outcome once more than you actually did

$$P_{LAP}(x) = \frac{c(x)+1}{\sum_x [c(x)+1]}$$

$$= \frac{c(x)+1}{N+|X|}$$



$$P_{ML}(X) = \left\langle \frac{2}{3}, \frac{1}{3} \right\rangle$$

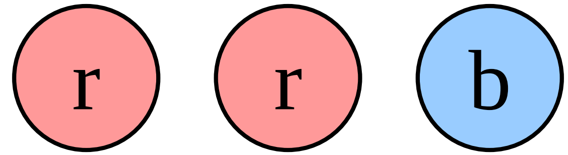
$$P_{LAP}(X) = \left\langle \frac{3}{5}, \frac{2}{5} \right\rangle$$

  - Can derive this estimate with Dirichlet priors
    - Bayesian estimation

Laplace Smoothing

- Laplace's estimate (extended):
  - Pretend you saw every outcome k extra times

$$P_{LAP,k}(x) = \frac{c(x) + k}{N + k|X|}$$



  - What's Laplace with k = 0?
  - k is the strength of the prior

$$P_{LAP,0}(X) =$$

$$P_{LAP,1}(X) =$$

$$P_{LAP,100}(X) =$$

- Laplace for conditionals:
  - Smooth each condition independently:

$$P_{LAP,k}(x|y) = \frac{c(x,y) + k}{c(y) + k|X|}$$

Linear Interpolation

- In practice, Laplace often performs poorly for P(X|Y):
  - When |X| is very large
  - When |Y| is very large

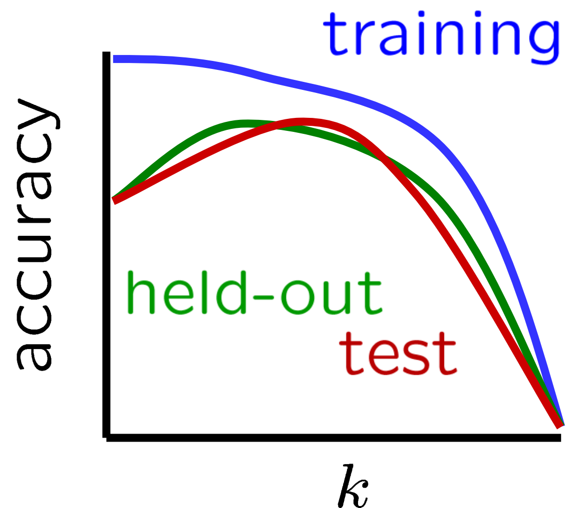
- Another option: linear interpolation
  - Also get the empirical $P(X)$ from the data
  - Make sure the estimate of $P(X|Y)$ isn't too different from the empirical $P(X)$

$$P_{LIN}(x|y) = \alpha \hat{P}(x|y) + (1.0 - \alpha)\hat{P}(x)$$

  - What if α is 0? 1?

Tuning on Validation Set

- Now we've got two kinds of unknowns
  - Parameters: the probabilities $P(X|Y), P(Y)$
  - Hyperparameters: e.g. the amount / type of smoothing to do, $k,\ \alpha$
- What should we learn where?
  - Learn parameters from training data
  - Tune hyperparameters on different data
  - For each value of the hyperparameters, train and test on validation se
  - Choose the best value and do a final test on the test data



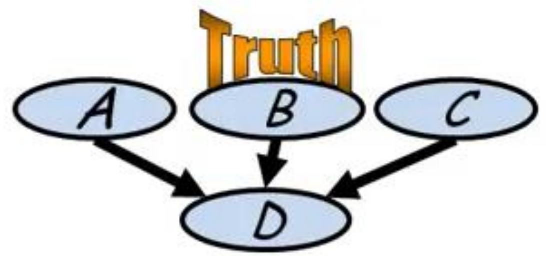
Model-based Classification

- Bayes rule lets us do diagnostic queries with causal probabilities
- The naïve Bayes assumption takes all features to be independent given the class label
- We can build classifiers out of a naïve Bayes model using training data
- Smoothing estimates is important in real systems

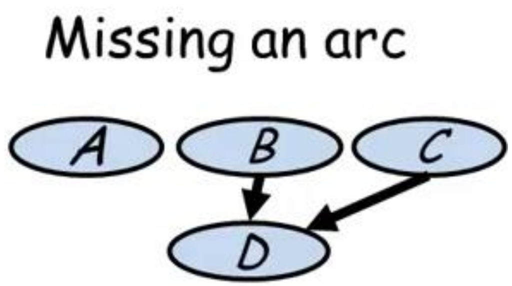
Structure Learning

- Importance of Accurate Structure

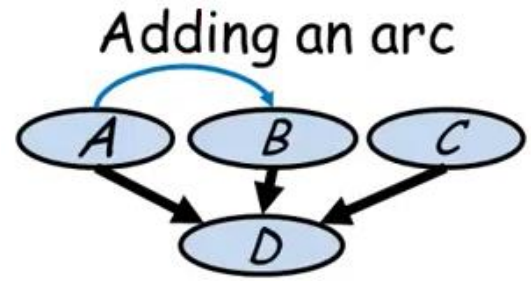
G^*



G_1



G_2



- Incorrect independencies
- Correct distribution P* cannot be learned
- But could generalize better

- Spurious dependencies
- Can correctly learn P*
- Increases # of parameters
- Worse generalization

Structure learning: To recover I-equivalence class as G*

Learning Metrics

- Likelihood structure score
- Bayesian Information Criterion (BIC) score
- Bayesian score

Likelihood Score

- Maximum likelihood estimation

$$\max_{\mathcal{G},\boldsymbol{\theta}_\mathcal{G}} L(\langle \mathcal{G}, \boldsymbol{\theta}_\mathcal{G}\rangle : \mathcal{D}) = \max_{\mathcal{G}}[\max_{\boldsymbol{\theta}_\mathcal{G}} L(\langle \mathcal{G}, \boldsymbol{\theta}_\mathcal{G}\rangle : \mathcal{D})]$$
$$= \max_{\mathcal{G}}[L(\langle \mathcal{G}, \hat{\boldsymbol{\theta}}_\mathcal{G}\rangle : \mathcal{D})]$$

- Likelihood score

$$\boxed{\text{score}_L(\mathcal{G} : \mathcal{D}) = \ell(\hat{\boldsymbol{\theta}}_\mathcal{G} : \mathcal{D})}$$

- Limits:
  - Adding edges can't hurt, and almost always helps
  - Score maximized for fully connected network
- Avoid overfitting
  - Restrict # of parents or # of parameters
  - Penalize complexity

BIC Score

- Bayesian Information Criterion (BIC) score

$$\boxed{\mathrm{score}_{BIC}(\mathcal{G} : \mathcal{D}) = \ell(\hat{\boldsymbol{\theta}}_{\mathcal{G}} : \mathcal{D}) - \frac{\log M}{2}\mathrm{Dim}[\mathcal{G}]}$$

$$= M \sum_{i=1}^{n} \boldsymbol{I}_{\hat{P}}(X_i; \mathrm{Pa}_{X_i}) - M \sum_{i=1}^{n} \boldsymbol{H}_{\hat{P}}(X_i) - \frac{\log M}{2}\mathrm{Dim}[\mathcal{G}]$$

  - Also known as minimum description length (MDL)
  - To explicitly penalize complexity
  - Mutual information grows linearly with M while complexity grows logarithmically with M
    - As M grows, more emphasis is given to fit to data
  - BIC is asymptotically consistent:
    - If data generated by G^*, networks I-equivalent to G^* will have highest score as M grows to ∞

Bayesian Score

- Bayesian score

$$\boxed{\mathrm{score}_B(\mathcal{G} : \mathcal{D}) = \log P(\mathcal{D} \mid \mathcal{G}) + \log P(\mathcal{G})}$$

  - Incorporate structure priors and parameter priors

Marginal likelihood

Prior over structures

$$P(\mathcal{G} \mid \mathcal{D}) = \frac{\boxed{P(\mathcal{D} \mid \mathcal{G})}\boxed{P(\mathcal{G})}}{\boxed{P(\mathcal{D})}}$$

Marginal probability of data

Likelihood

Prior over parameters

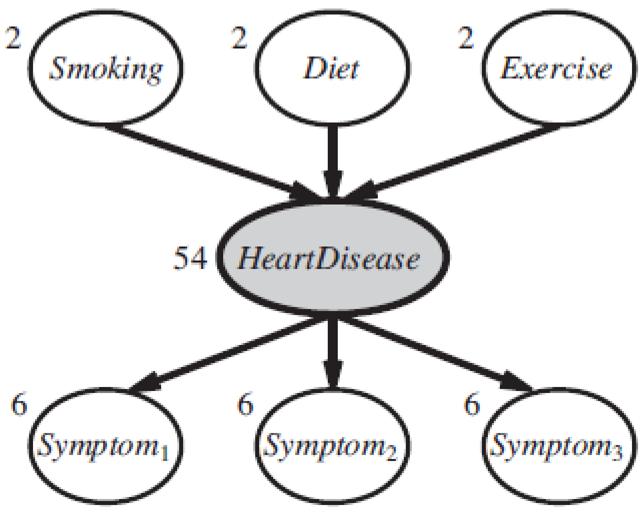
$$P(\mathcal{D} \mid \mathcal{G}) = \int_{\Theta_{\mathcal{G}}} \boxed{P(\mathcal{D} \mid \boldsymbol{\theta}_{\mathcal{G}}, \mathcal{G})}\boxed{P(\boldsymbol{\theta}_{\mathcal{G}} \mid \mathcal{G})} d\boldsymbol{\theta}_{\mathcal{G}}$$

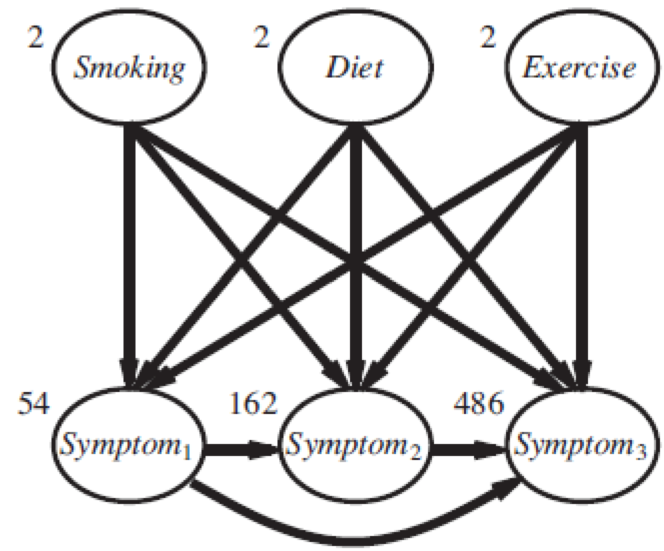
Bayesian Score

- Bayesian score averages over parameters to avoid overfitting
  - Can naturally incorporate prior knowledge
  - I-equivalent networks have same score
- Bayesian score
  - Asymptotically equivalent to BIC
  - Asymptotically consistent
  - But for small M, BIC tends to underfit

Learning with Hidden Variables

- Why hidden variables?
  - Reduce the number of parameters
  - Reduce the amount of data needed to learn the parameters





Each variable has 3 values

78 parameters

Incomplete data

708 parameters

Expectation Maximization (EM)

- EM is designed for optimizing likelihood functions
- Intuition:
  - Parameter estimation is easy given complete data
  - Computing probability of missing data (= inference) is easy given parameters

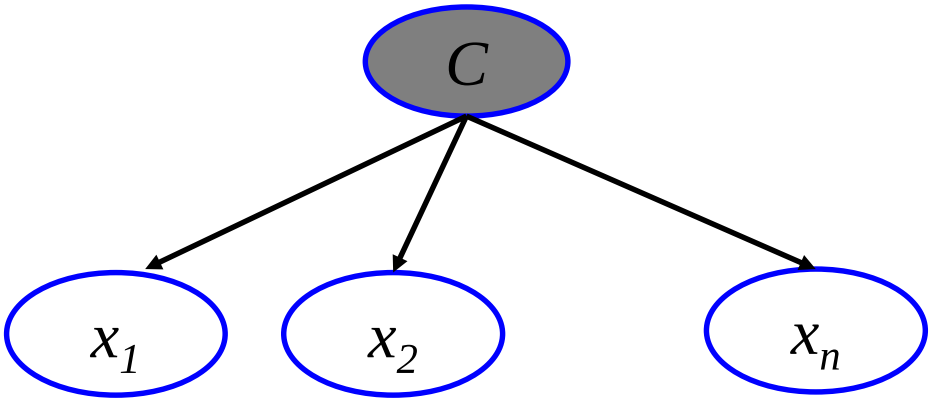
Expectation Maximization (EM)

- Algorithm:

  - Pick a start point for parameters
  - Iterate
    - E-Step (Expectation): Complete the data using current parameters
    - M-Step (Maximization): Estimate parameters relative to data completion, using MLE

Applications: Unsupervised Clustering

- Learning Mixtures of Gaussians



  - The mixture distribution:

$$P(\mathbf{x}) = \sum_{i=1}^{k} P(C=i)\; P(\mathbf{x} \mid C=i)$$

Applications: Unsupervised Clustering

- EM algorithm
  - E-step: Compute the probabilities

$$p_{ij} = P(C = i \mid \mathbf{x}_j)$$

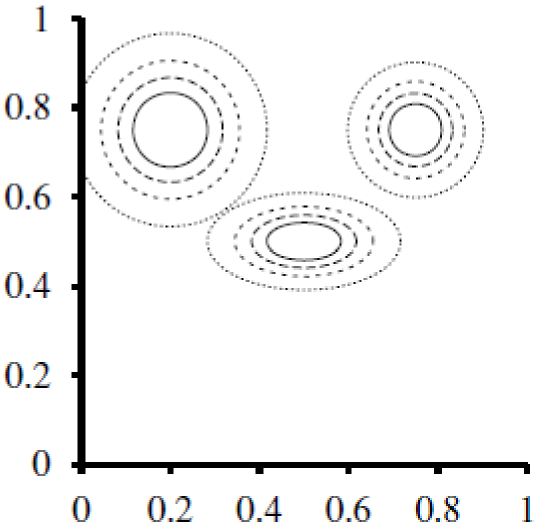
$$= \alpha P(\mathbf{x}_j \mid C = i) P(C = i)$$

  - M-step: Compute the new mean, covariance and component weights

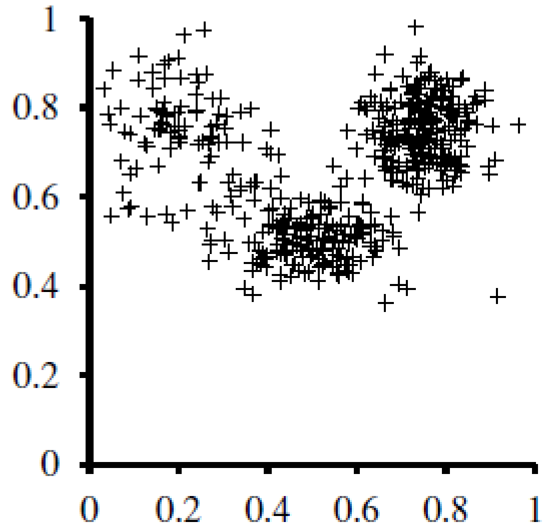
$$\begin{aligned} \boldsymbol{\mu}_i &\leftarrow \sum_j p_{ij} \mathbf{x}_j / n_i \\ \boldsymbol{\Sigma}_i &\leftarrow \sum_j p_{ij} (\mathbf{x}_j - \boldsymbol{\mu}_i)(\mathbf{x}_j - \boldsymbol{\mu}_i)^\top / n_i \\ w_i &\leftarrow n_i / N \end{aligned}$$

Applications: Unsupervised Clustering

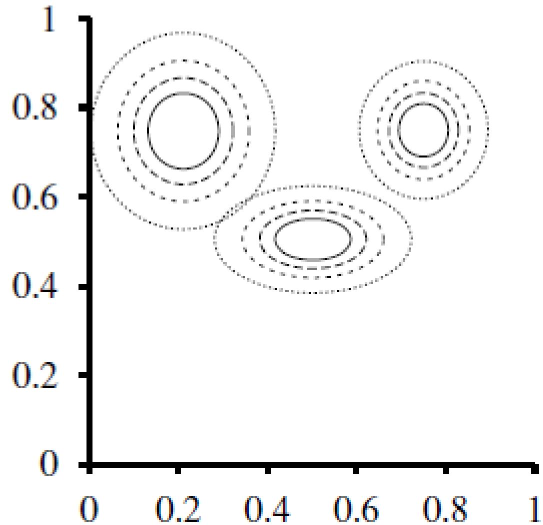
- Learning Mixtures of Gaussians



A Gaussian mixture model
with 3 components



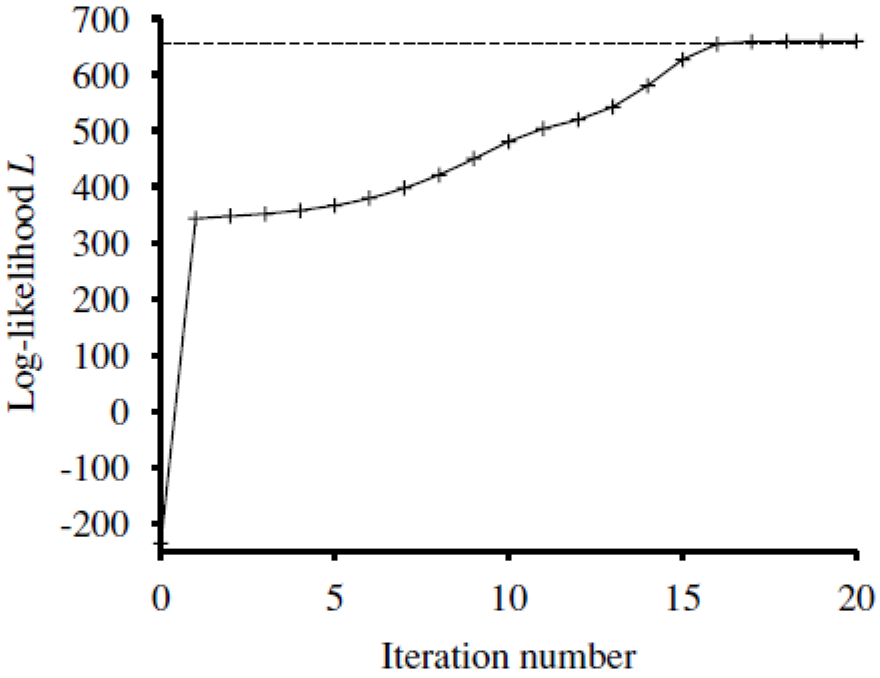
500 samples



EM results

Applications: Unsupervised Clustering

- EM algorithm:
  - Increases the log-likelihood of the data at every iteration
  - Local optima are unavoidable, local optima can be very different
  - Initialization is critical



Readings

- Artificial Intelligence
  - Chapter 18.1-18.2, 18.4
  - Chapter 20

- Assignment:
  - Homework 5