Homework Remaining

• Questions about homework #3?
  – Notes: can use any package (Matlab has a good one), but we’ve only verified functionality for Weka

• Homework #4 will be about sampling & learning
  – Mostly programming (15 pts)

• Homework #5 will be project experiments, write-up, presentation
  – Worth 20 points (so +5 “extra” points)

• ...Homeworks #3-#5: the “how” of Graphical Models

• Then final, project presentations (about 6 minutes each)
Road Map

• Basics of Probability and Statistical Estimation
• Bayesian Networks
• Markov Networks
• Inference
• Learning
  – Parameters, Structure, EM
• HMMs
• Something else?
  – Candidates: Active Learning, Decision Theory, Statistical Relational Models...
  Role of Probabilistic Models in the Financial Crisis?
Road Map

• Basics of Probability and Statistical Estimation
• Bayesian Networks
• Markov Networks
• Inference
• Learning
  – Parameters, Structure, EM
• HMMs
• Something else?
  – Candidates: Active Learning, Decision Theory, Statistical Relational Models...
  Role of Probabilistic Models in the Financial Crisis?
Today: Learning

- General Rules of Thumb in Learning

- Learning in Graphical Models
  - Parameters in Bayes Nets
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

• Reminder: ignore test set until end of quarter!

• 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: two-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

<table>
<thead>
<tr>
<th>Training Data</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

\[ P_{ML}(A) = 0.714 \]
\[ P_{ML}(B \mid A=1) = 0.6 \]
Learning in Bayes Nets – details

• Problem statement (for today):
  – Given a Bayes Network structure $G$, and a set of complete training examples $\{X_i\}$
  – 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:

\[ P(X = \text{heads}) = \theta \]
Learning as Inference

• Think of learning $P(\Theta = \theta \mid \{X_i\})$ as inference

$$P(X_i = \text{heads}) = \theta$$
Next Simplest Bayes Net

X heads/tails

heads

tails

Y heads/tails

“heads”

“tails”
Next Simplest Bayes Net

heads/tails \( X \) heads/tails \( Y \)

\( \Theta_X \) \( \Theta_Y \)

\( X_1 \) \( X_2 \) \( \ldots \) \( X_N \)
toss 1 toss 2 toss \( N \)

\( X_1 \) \( X_2 \) \( \ldots \) \( X_N \)
toss 1 toss 2 toss \( N \)
Next Simplest Bayes Net

heads/tails $X$ heads/tails $Y$

$toss\ 1$ $toss\ 2$ $\ldots$ $toss\ N$

$toss\ 1$ $toss\ 2$ $\ldots$ $toss\ N$
Next Simplest Bayes Net

Parameter Independence

\[ X \quad \Theta_X \quad X_1 \quad X_2 \quad ... \quad X_N \]

toss 1

toss 2

toss \(N\)

\[ \Theta_Y \quad X_1 \quad X_2 \quad ... \quad X_N \]

toss 1

toss 2

toss \(N\)
Getting Tougher

Three probabilities to learn:

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

$X \rightarrow Y$

$X_1 \rightarrow Y_1$

$X_2 \rightarrow Y_2$

(case 1)

(case 2)
Parameter Independence

$X \rightarrow Y$

$X_1$, $X_2$, ..., $Y_1$, $Y_2$, ...

$X \overset{\Theta}{\rightarrow} Y_{|X=\text{heads}}$, $Y_{|X=\text{tails}}$

heads/tails
Three **Separate** Thumbtack Problems

heads/tails $X$ $\rightarrow$ heads/tails $Y$

case 1

$X_1$ heads $\rightarrow$ $Y_1$

$X_2$ tails $\rightarrow$ $Y_2$

\[\theta_{X} \quad \theta_{Y|X=\text{heads}} \quad \theta_{Y|X=\text{tails}}\]

\[\theta_{X}\]

\[\theta_{Y|X=\text{heads}}\]

\[\theta_{Y|X=\text{tails}}\]

case 2

\[\ldots\]

\[\ldots\]
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

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\[
P_{ML}(B \mid A=0) = \frac{2}{2} = 1.0
\]

\[
P_{MAP}(B \mid A=0) = \frac{(2+1)}{(3+2)} = 0.6
\]

“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 $X_i$
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, WinMine
      (it’s fine to use Weka for HW#3, then switch to a different package later)
Continuous CPT Example

E.g., Linear Gaussian

\[ P(X \mid u) = N(\beta_0 + \beta_1 u_1 + \ldots \beta_k u_k; \sigma^2) \]
Linear Gaussian

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

\[ E[X] = \beta_0 + \beta_1 E[u_1] + \ldots + \beta_k E[u_k] \]
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

- \( P(C \mid B) = \)

<table>
<thead>
<tr>
<th>( B=0 )</th>
<th>( 4/12 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( B=1 )</td>
<td>( 16/16 )</td>
</tr>
</tbody>
</table>

- Using \( P(C \mid A, B) \) => zero training error (vs. 17% error for \( P(C \mid B) \)), but cells have 12, 8, 4, 4 total samples

- => Very susceptible to random noise

Training data is the following, repeated 4 times:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
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

```
  Spam
  "Lottery"   "winner"   . . .   "Dear"
```
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
  – Next week
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 data

• 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 $P(Y \mid X)$ using Bayes Rule, apply Naïve Bayes assumption
  - $2n + 1$ parameters, for $n$ observed variables
Generative => Discriminative (1 of 2)

- But $P(Y \mid X)$ can be written more compactly
  
  \[
P(Y \mid X) = \frac{1}{1 + \exp(w_0 + w_1 x_1 + \ldots + w_n x_n)}
  \]

- Total of $n + 1$ parameters $w_i$
Generative => Discriminative (2 of 2)

• We reduced $2n + 1$ parameters to $n + 1$
  – Bias vs. Variance arguments says this must be better, right?

• Not exactly. If we construct $P(Y \mid X)$ 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:
  \[ P(Y \mid X) = \frac{1}{1 + \exp(w_0 + w_1 x_1 + \ldots + w_n x_n)} \]

• 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
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

<table>
<thead>
<tr>
<th>SPAM</th>
<th>Lottery</th>
<th>Winner</th>
<th>Lunch</th>
<th>Noon</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

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

$$P(x) = \frac{\prod_c \phi_c(x_c)}{Z}$$

$$Z = \sum_x \prod_c \phi_c(x_c)$$

<table>
<thead>
<tr>
<th>Grades</th>
<th>TV</th>
<th>$\phi_1(G, TV)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>Little</td>
<td>2.0</td>
</tr>
<tr>
<td>Good</td>
<td>Little</td>
<td>3.0</td>
</tr>
<tr>
<td>Low</td>
<td>Lots</td>
<td>3.0</td>
</tr>
<tr>
<td>Good</td>
<td>Lots</td>
<td>1.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TV</th>
<th>Trivia Knowledge</th>
<th>$\phi_2(TV, TK)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Little</td>
<td>Little</td>
<td>2.0</td>
</tr>
<tr>
<td>Lots</td>
<td>Little</td>
<td>1.0</td>
</tr>
<tr>
<td>Little</td>
<td>Lots</td>
<td>1.5</td>
</tr>
<tr>
<td>Lots</td>
<td>Lots</td>
<td>3.0</td>
</tr>
</tbody>
</table>
Log-linear Formulation (1 of 2)

• \( P(x) = \frac{\exp(\sum_i w_i f_i(D_i))}{Z} \)

• Example, write \( \phi_1(G, TV) \) as \( \exp(w_1 f_1(G, TV) + \ldots + w_4 f_4(G, TV)) \)
  \( w_1 = \ln 2.0 \quad w_2 = \ln 3.0 \quad w_3 = \ln 3.0 \quad w_4 = \ln 1.0 \)

<table>
<thead>
<tr>
<th>Grades</th>
<th>TV</th>
<th>( \phi_1(G, TV) )</th>
<th>( f_1(G, TV) )</th>
<th>( f_2(G, TV) )</th>
<th>( f_3(G, TV) )</th>
<th>( f_4(G, TV) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>Little</td>
<td>2.0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Good</td>
<td>Little</td>
<td>3.0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Low</td>
<td>Lots</td>
<td>3.0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Good</td>
<td>Lots</td>
<td>1.0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Log-linear Formulation (2 of 2)

- \( P(x) = \exp(\sum_i w_i f_i(D_i)) \frac{1}{Z} \)

- 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:

\[
P(\text{Data} \mid \theta) = \prod_{m \in \text{Data}} \prod_{i} P(X_i[m] \mid \text{Parents}(X_i)[m] : \theta_i)
\]

where \(X_i[m]\) is the assignment to \(X_i\) in example \(m\)

\[
= \prod_{i} \prod_{m \in \text{Data}} P(X_i[m] \mid \text{Parents}(X_i)[m] : \theta_i)
\]

• Assuming param independence, maximize global likelihood by maximizing each CPT likelihood \(P(X_i[m] \mid \text{Parents}(X_i)[m] : \theta_i)\) independently
Learning in Markov Networks

• Harder than in Bayes Nets
• In Markov Net, 
  
  Likelihood = 
  
  \[ P(\text{Data} \mid \mathbf{w}) = \prod_{m \in \text{Data}} \frac{\exp(\sum_i w_i f_i(D_i[m]))}{Z_w} \]

• But \( Z_w = \sum_{x \in \text{Val}(x)} \exp(\sum_i w_i f_i(x)) \)
  
  – 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

\[ \frac{\partial}{\partial w_i} \ln P(\text{Data} \mid w) = E_{\text{Data}}[f_i(D_i)] - E_w[f_i(D_i)] \]

• Concave (no local maxima)
• Requires inference at each step
  – Slow
Approximation: Pseudo-likelihood

- Pseudo-likelihood \( P_{\text{L}}(\text{Data} \mid \theta) = \prod_{m \in \text{Data}} \prod_{i} P(X_i[m] \mid \text{Neighbors}(X_i)[m] : \theta_i) \)
  - 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 $P(Y \mid X)$
- $\frac{\partial}{\partial w_i} \ln P(Y_{\text{Data}} \mid X_{\text{Data}}, w) = \sum_m (f_i(y[m], x[m])) - E_{w}[f_i \mid x[m]]$
- 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