CSE 573: Artificial Intelligence  
Spring 2012  
Structure Learning, EM, Cotraining  
Dan Weld

Some Typical Biases

- Occam’s razor
- MDL – Minimum description length
- Concepts can be approximated by
  - conjunctions of predicates,
  - linear functions
  - short decision trees
- Maximal conditional independence
- Minimum cross-validation error
- Minimum number of features
- Etc..

Overfitting

Learning as Optimization

- Methods
  - Closed form
  - Greedy search
  - Gradient ascent
- Loss Function (preference bias)
  - Minimize loss over training data (test data)
  - Loss(h, data) = error(h, data) + complexity(h)

Effect of Regularization

Regularization term \( \lambda ||w||^2 \)
Bias / Variance Tradeoff

- **Variance:** \( E[(h(x^*) - h(x^*))^2] \)
  How much \( h(x^*) \) varies between training sets
  Reducing variance risks underfitting

- **Bias:** \([h(x^*) - f(x^*)]\)
  Describes the *average* error of \( h(x^*) \)
  Reducing bias risks overfitting

Note: inductive bias vs estimator bias

Bias-Variance Tradeoff

- High Variance
- High Bias

Topics

- **Learning Parameters for a Bayesian Network**
  - Fully observable
  - Hidden variables (EM algorithm)
- **Learning Structure of Bayesian Networks**
- **Cool Stuff**
  - Learning Ensembles
  - Cotraining

Parameter Estimation and Bayesian Networks

Learning with Continuous Variables

- \( \hat{\mu}_{MLE} = \frac{1}{N} \sum_{i=1}^{N} x_i \)
- \( \hat{\sigma}^2_{MLE} = \frac{1}{N} \sum_{i=1}^{N} (x_i - \hat{\mu})^2 \)

Summary

<table>
<thead>
<tr>
<th>Prior</th>
<th>Hypothesis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform</td>
<td>The most likely</td>
</tr>
<tr>
<td>Any</td>
<td>The most likely</td>
</tr>
<tr>
<td>Any</td>
<td>Weighted combination</td>
</tr>
</tbody>
</table>

Minimizes error

Great when data is scarce

Potentially much harder to compute
A Popular Structure: Naïve Bayes

\[ P(Y, F_1 \ldots F_n) = P(Y) \prod_i P(F_i | Y) \]

Assume that features are conditionally independent given class variable.
Works surprisingly well for classification (predicting the right class).
But forces probabilities towards 0 and 1.

What if we don't know structure?

Learning The Structure of Bayesian Networks

- Search thru the space…
  - of possible network structures!
  - (for now still assume can observe all values)
- For each structure, learn parameters
  - As just shown…
- Pick the one that fits observed data best
  - Calculate P(data)

Two problems:
- Fully connected graph will be most probable
- Exponential number of structures

Learning The Structure of Bayesian Networks

- Search thru the space…
  - of possible network structures!
- For each structure, learn parameters
  - As just shown…
- Pick the one that fits observed data best
  - Calculate P(data)

Two problems:
- Fully connected will be most probable
  - Add penalty term (regularization) \( \propto \) model complexity
  - Exponential number of structures
  - Local search
Score Functions

- **Bayesian Information Criterion (BIC)**
  - $P(D | BN) - \text{penalty}$
  - Penalty = $\frac{1}{2} \times \text{(parameters)} \times \log(\text{data points})$

- **MAP score**
  - $P(BN | D) = P(D | BN) P(BN)$
  - $P(BN)$ must decay exponentially with # of parameters for this to work well

- Loss(h, data) = $\text{error}(h, \text{data}) + \text{complexity}(h)$

Topics

- Learning Parameters for a Bayesian Network
  - Fully observable
  - Hidden variables (EM algorithm)

- Learning Structure of Bayesian Networks
- Cool Stuff
  - Learning Ensembles
  - Cotraining

Why Learn Hidden Variables?

Chicken & Egg Problem

- If we knew whether patient had disease
  - It would be easy to learn CPTs
  - But we can’t observe states, so we don’t!

- If we knew CPTs
  - It would be easy to predict if patient had disease
  - But we don’t, so we can’t!

Continuous Variables

Earthquake

- $\Pr(E | A)$
  - $\mu = 6$
  - $\sigma = 2$
  - $\mu = 1$
  - $\sigma = 3$

Aliens

- $\Pr(A | \text{OUT})$
Simplest Version

- Mixture of two distributions
- Know: form of distribution & variance, \( \sigma = 1 \)
- Just need mean of each distribution

Input Looks Like

We Want to Predict

Chicken & Egg

Note that coloring instances would be easy if we knew Gaussian parameters....

Chicken & Egg

And finding the Gaussians would be easy if we knew the coloring

Expectation Maximization (EM)

- Pretend we do know the parameters
- Initialize randomly: set \( \theta_1 = ?; \theta_2 = ? \)
Expectation Maximization (EM)

- Pretend we do know the parameters
  - Initialize randomly
- [E step] Compute probability of instance having each possible value of the hidden variable

ML Mean of Single Gaussian

$$U_{ml} = \arg\min_u \sum (x_i - u)^2$$
Expectation Maximization (EM)

- **E step**: Compute probability of instance having each possible value of the hidden variable
- **M step**: Treating each instance as fractionally having both values compute the new parameter values

### Topics

- Learning Parameters for a Bayesian Network
  - Fully observable
  - Hidden variables (EM algorithm)
- Learning Structure of Bayesian Networks
- Cool Stuff
  - Learning Ensembles
  - Co-training

### Ensembles of Classifiers

- Traditional approach: Use one classifier
- Can one do better?
- Approaches:
  - Cross-validated committees
  - Bagging
  - Boosting
  - Stacking

- Ensembles: Assume errors are independent (suppose 30% error)
  - Probability that majority is wrong:
    
    \[
    \text{Prob} = \frac{\text{area under binomial distribution}}{\text{Number of classifiers in error}}
    \]

  - If individual area is 0.3
  - Area under curve for ≥11 wrong is 0.026
  - Order of magnitude improvement!
Constructing Ensembles

Cross-validated committees

- Partition examples into \( k \) disjoint equiv classes
- Now create \( k \) training sets
  - Each set is union of all equiv classes \textit{except one}
  - So each set has \((k-1)/k\) of the original training data
- Now train a classifier on each set

Ensemble Construction II

Bagging

- Generate \( k \) sets of training examples
- For each set
  - Draw \( m \) examples randomly (with replacement)
  - From the original set of \( m \) examples
  - Each training set corresponds to \( 63.2\% \) of original (+ duplicates)
  - Now train classifier on each set
- Intuition: Sampling helps algorithm become more robust to noise/outliers in the data

Boosting \[\text{[Schapire, 1989]}\]

- Idea: run weak learner multiple times on (reweighted!) training data; weight learned classifiers \( \propto \) their accuracy
- On each iteration \( t \):
  - Learn a hypothesis, \( h_t \), using distribution to weight examples
  - Compute a strength for this hypothesis \( \alpha_t \)
  - Reweight training examples by how well they were classified
- Final classifier:
  \[
  h(x) = \text{sign} \left( \sum_t \alpha_t h_t(x) \right)
  \]
- Practically useful
- Theoretically interesting
Bagging vs Boosting

Train several base learners
Next train meta-learner
- Learns when base learners are right / wrong
- Now meta learner arbitrates

Train using cross validated committees
- Meta-L inputs = base learner predictions
- Training examples = 'test set' from cross validation

Topics

- Learning Parameters for a Bayesian Network
  - Fully observable
  - Hidden variables (EM algorithm)
- Learning Structure of Bayesian Networks
- Cool Stuff
  - Learning Ensembles
  - Semi-supervised learning (Cotraining)