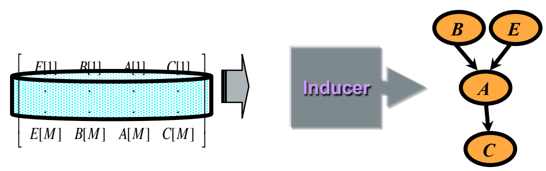
Machine Learning II: Beyond Decision Trees

AI Class 15 (Ch. 20.1–20.2)



Data D

Bookkeeping

- **Midterm Tuesday!**
- Project design: 10/31 @ 11:59
  - If you have not read the project description **carefully**, do so!
  - Phase II will be fleshed out after your designs are in.
- Blackboard bug – assume single turnins. :-/
- A note on changing grades
  - Short version: don't ask the grader or TA. Questions are okay, but grade change requests go through me
- HW4 out by 11:59; due 11/7 @ 11:59

Today's Class

- Extensions to Decision Trees
- Sources of error
- Evaluating learned models
- Bayesian Learning
- MLA, MLE, MAP
- Bayesian Networks I

Information Gain

- Concept: make decisions that increase the homogeneity of the data subsets (for outcomes)
  - Good: Bad:
- **Information gain** is based on:
  - **Decrease in entropy**
  - After a dataset is split on an attribute.
  - → High homogeneity – e.g., likelihood samples will have the same class (outcome)

Extensions of the Decision Tree Learning Algorithm

- Using gain ratios
- Real-valued data
- Noisy data and overfitting
- Generation of rules
- Setting parameters
- Cross-validation for experimental validation of performance
- C4.5 is an extension of ID3 that accounts for unavailable values, continuous attribute value ranges, pruning of decision trees, rule derivation, and so on

Using Gain Ratios

- Information gain favors attributes with a **large number of values**
  - If we have an attribute D that has a distinct value for each record, then *Info*(D,T) is 0, thus *Gain*(D,T) is maximal
- To compensate, use the following ratio instead of Gain:
  GainRatio(D,T) = Gain(D,T) / SplitInfo(D,T)
- SplitInfo(D,T) is the information due to the split of T on the basis of value of categorical attribute D
  $SplitInfo(D,T) = I(|T_1|/|T|, |T_2|/|T|, .., |T_m|/|T|)$

where $\{T_1, T_2, .. T_m\}$ is the partition of T induced by value of D

Real-Valued Data

- Select a set of thresholds defining intervals
  - Each interval becomes a discrete value of the attribute
- How?
  - Use simple heuristics…
    - Always divide into quartiles
  - Use domain knowledge…
    - Divide age into infant (0-2), toddler (3 - 5), school-aged (5-8)
  - Or treat this as another learning problem
    - Try a range of ways to discretize the continuous variable and see which yield "better results" w.r.t. some metric
    - E.g., try midpoint between every pair of values

Noisy Data

- Many kinds of "noise" can occur in the examples:
  - Two examples have same attribute/value pairs, but different classifications
  - Some values of attributes are incorrect
    - Errors in the data acquisition process, the preprocessing phase, //
  - Classification is wrong (e.g., + instead of -) because of some error
  - Some attributes are irrelevant to the decision-making process, e.g., color of a die is irrelevant to its outcome
  - Some attributes are missing (are pangolins bipedal?)

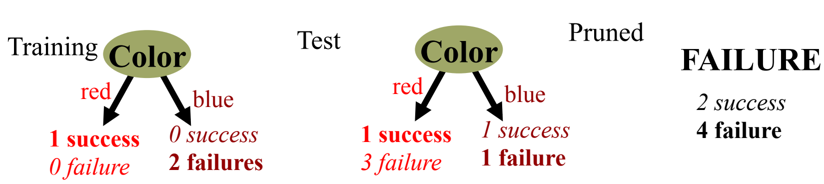
Overfitting

- Overfitting: coming up with a model that is TOO specific to your training data
  - Does well on training set but not new data
  - How can this happen?
- Too little training data
- Irrelevant attributes
  - high-dimensional (many attributes) hypothesis space → meaningless regularity in the data irrelevant to important, distinguishing features
  - Fix by pruning lower nodes in the decision tree
  - For example, if Gain of the best attribute at a node is below a threshold, stop and make this node a leaf rather than generating children nodes

13

Pruning Decision Trees

- Replace a whole subtree by a leaf node
- If: a **decision rule** establishes that he expected error rate in the subtree is greater than in the single leaf. E.g.,
  - Training: one training red success and two training blue failures
  - Test: three red failures and one blue success
  - Consider replacing this subtree by a single Failure node. (leaf)
- After replacement we will have only two errors instead of five:



14

Converting Decision Trees to Rules

- It is easy to derive a rule set from a decision tree:
  - Write a rule for **each path** in the decision tree from the root to a leaf
- Left-hand side is label of nodes and labels of arcs
- The resulting rules set can be simplified:
  - Let LHS be the left hand side of a rule
  - Let LHS' be obtained from LHS by eliminating some conditions
  - We can replace LHS by LHS' in this rule if the subsets of the training set that satisfy respectively LHS and LHS' are equal
- A rule may be eliminated by using metaconditions such as "if no other rule applies"

15

Measuring Model Quality

- How good is a model?
  - Predictive accuracy
  - False positives / false negatives for a given cutoff threshold
    - Loss function (accounts for cost of different types of errors)
  - Area under the (ROC) curve
  - Minimizing loss can lead to problems with overfitting

17

Measuring Model Quality

- Training error
  - Train on all data; measure error on all data
  - Subject to overfitting (of course we'll make good predictions on the data on which we trained!)
- Regularization
  - Attempt to avoid overfitting
  - Explicitly minimize the complexity of the function while minimizing loss
  - Tradeoff is modeled with a *regularization parameter*

Cross-Validation

- Holdout cross-validation:
  - Divide data into training set and test set
  - Train on training set; measure error on test set
  - Better than training error, since we are measuring *generalization to new data*
  - To get a good estimate, we need a reasonably large test set
  - But this gives less data to train on, reducing our model quality!

Cross-Validation, cont.

- k-fold cross-validation:
  - Divide data into *k* folds
  - Train on *k-1* folds, use the *k*th fold to measure error
  - Repeat *k* times; use average error to measure generalization accuracy
  - Statistically valid and gives good accuracy estimates
- Leave-one-out cross-validation (LOOCV)
  - *k*-fold cross validation where *k*=*N* (test data = 1 instance!)
  - Quite accurate, but also quite expensive, since it requires building *N* models

20

Bayesian Learning

Chapter 20.1-20.2

Some material adapted from lecture notes by Lise Getoor and Ron Parr

26

Naïve Bayes

- Use Bayesian modeling
- Make the simplest possible independence assumption:
  - Each attribute is independent of the values of the other attributes, given the class variable
  - In our restaurant domain: Cuisine is independent of Patrons, *given* a decision to stay (or not)

Bayesian Formulation

- The probability of class C given $F_1, ..., F_n$

  $$p(C \mid F_1, ..., F_n) = p(C)\, p(F_1, ..., F_n \mid C) / P(F_1, ..., F_n)$$
  $$= \alpha\, p(C)\, p(F_1, ..., F_n \mid C)$$

- Assume that each feature F_i is conditionally independent of the other features given the class C. Then:

  $$p(C \mid F_1, ..., F_n) = \alpha\, p(C)\, \Pi_i\, p(F_i \mid C)$$

- We can estimate each of these conditional probabilities from the observed counts in the training data:

  $$p(F_i \mid C) = N(F_i \wedge C) / N(C)$$

  - One subtlety of using the algorithm in practice: When your estimated probabilities are zero, ugly things happen
  - The fix: Add one to every count (aka "Laplacian smoothing")

Naive Bayes: Example

- p(Wait | Cuisine, Patrons, Rainy?)
  = α p(Cuisine $\wedge$ Patrons $\wedge$ Rainy? | Wait)
  = α p(Wait) p(Cuisine | Wait) p(Patrons | Wait) p(Rainy? | Wait)

naive Bayes assumption: is it reasonable?

Naive Bayes: Analysis

- Naïve Bayes is amazingly easy to implement (once you understand the bit of math behind it)
- Naïve Bayes can outperform many much more complex algorithms—it's a baseline that should pretty much always be used for comparison
- Naive Bayes can't capture interdependencies between variables (obviously)—for that, we need Bayes nets!

Learning Bayesian Networks

Bayesian Learning: Bayes' Rule

- Given some **model space** (set of hypotheses h_i) and **evidence** (data D):
  - $P(h_i | D) = \alpha \, P(D | h_i) \, P(h_i)$
- We assume observations are independent of each other, given a model (hypothesis), so:
  - $P(h_i | D) = \alpha \prod_j P(d_j | h_i) \, P(h_i)$
- To predict the value of some unknown quantity X (e.g., the class label for a future observation):
  - $P(X | D) = \sum_i P(X | D, h_i) \, P(h_i | D) = \sum_i P(X | h_i) \, P(h_i | D)$

These are equal by our independence assumption

Bayesian Learning, 3 Ways

- **BMA (Bayesian Model Averaging)**
  - Don't just choose one hypothesis; instead, make predictions based on the weighted average of all hypotheses (or some set of best hypotheses)
- **MAP (Maximum *A Posteriori*) hypothesis**
  - Choose hypothesis with highest *a posteriori* probability, given data
  - **Maximize $p(h_i \mid D)$**
  - Generally easier than Bayesian learning
  - Closer to Bayesian prediction as more data arrives
- **MLE (Maximum Likelihood Estimate)**
  - Assume all hypotheses are equally likely *a priori*; best hypothesis maximizes the **likelihood** (i.e., probability of data given hypothesis)
  - **Maximize $p(D \mid h_i)$**

33

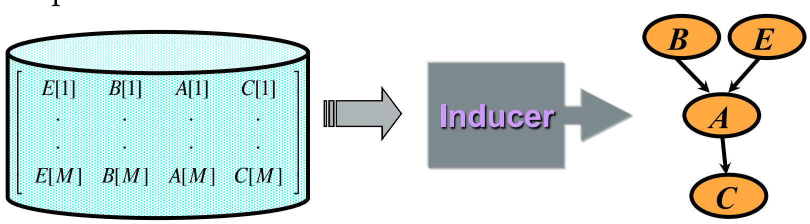
Bayesian Learning

- **BMA (Bayesian Model Averaging) –** average predictions of hypotheses
- **MAP (Maximum *A Posteriori*) hypothesis –** Maximize $p(h_i \mid D)$
- **MLE (Maximum Likelihood Estimate) –** Maximize $p(D \mid h_i)$
- **MDL (Minimum Description Length) principle:** Use some encoding to model the **complexity** of the hypothesis, and the fit of the data to the hypothesis, then **minimize** the overall description of $h_i + D$

34

Learning Bayesian Networks

- Given training set $D = \{x[1], \ldots, x[M]\}$
- Find B that best matches D
  - model selection
  - parameter estimation



Data D

Parameter Estimation

i.i.d. samples
independent and identically distributed (i.i.d.) if each random variable has the same probability distribution as the others and all are mutually independent

- Assume known structure
- Goal: estimate BN param
  - entries in local probability models, P(X | Paren
- A good parameterization **q** is likely to ge te observed data:

$$L(\theta : D) = P(D \mid \theta) = \prod_m P(x[m] \mid \theta)$$

- Maximum Likelihood Estimation (MLE) Principle: Choose $\mathbf{q}^*$ so as to maximize L

Parameter Estimation II

- The likelihood **decomposes** according to the structure of the network
  → we get a separate estimation task for each parameter
- The MLE (maximum likelihood estimate) solution:
  - for each value *x* of a node *X*
  - and each instantiation ***u*** of *Parents(X)*

$$\theta^*_{x|u} = \frac{N(x,u)}{N(u)}$$

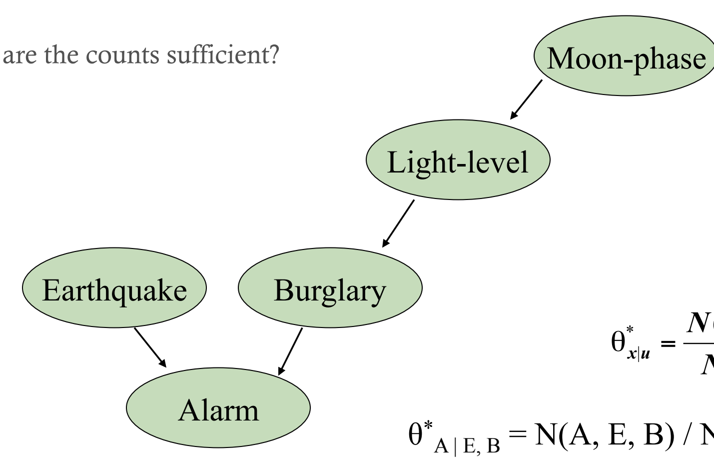
sufficient statistics

  - Just need to collect the counts for every combination of parents and children observed in the data
  - MLE is equivalent to an assumption of a uniform prior over parameter values

37

Sufficient Statistics: Example

- Why are the counts sufficient?



$$\theta^*_{x|u} = \frac{N(x,u)}{N(u)}$$

$$\theta^*_{A\,|\,E,\,B} = N(A, E, B) \,/\, N(E, B)$$

38

Model Selection

Goal: Select the best network structure, given the data

Input:

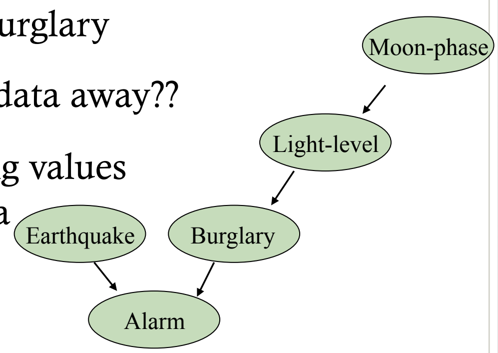
- Training data
- Scoring function

Output:

- A network that maximizes the score

Handling Missing Data

- Suppose that in some cases, we observe earthquake, alarm, light-level, and moon-phase, but not burglary
- Should we throw that data away??
- **Idea**: Guess the missing values based on the other data



EM (Expectation Maximization)

- **Guess** probabilities for nodes with **missing values** (e.g., based on other observations)
- **Compute the probability distribution** over the missing values, given our guess
- **Update the probabilities** based on the guessed values
- **Repeat** until convergence

EM Example

- Suppose we have observed Earthquake and Alarm but not Burglary for an observation on November 27
- We estimate the CPTs based on the *rest* of the data
- We then estimate P(Burglary) for November 27 from those CPTs
- Now we recompute the CPTs as if that estimated value had been observed
- Repeat until convergence!