- Today's Reading:
  - HMS, 6.3, 7.3.1, 10
- Today's Lecture:
  - Predictive Modeling
    - classification
    - regression
- Upcoming Due Dates:
  - Project progress report due today
  - H3 due Tue 4/16

Descriptive vs. Predictive Models

- Descriptive models summarize the data
  - ideally providing insight into the domain
  - and providing a flexible model for inferring arbitrary collections of attributes from a set of observed values

- Predictive models aim to predict the value of one variable of interest given known values of other attributes, e.g.,
  - diagnosis of patient given test results
  - purchase of product A given other product purchases
  - predicting the credit-worthiness of an applicant, given information about their credit history and income

Predictive Modeling

- Learn mapping from input vector **x** to scalar output y
- Training set, pairs **x**(i), y(i)
- Goal: estimate a function $y = f(\mathbf{x};\theta)$
  - if y is categorical, called (supervised) *classification*
  - if y is real-valued, called *regression*
- Choose:
  1. The model family, the functional form f
  2. The score function
  3. An optimization strategy for finding the best model and parameter values θ

Score Function

- Difference between the prediction of the model, $\hat{y}(i) = f(\mathbf{x};\theta)$ and the true value for $y(i)$

$$S(\theta) = \sum_{i=1}^{N} d(y(i), f(\mathbf{x}(i);\theta))$$

Score function for Classification

- Misclassification rate (also called error rate, zero-one score function):

$$S_{0/1}(\theta) = \frac{1}{N}\sum_{i=1}^{N} I(f(\mathbf{x}(i);\theta), y(i))$$

- where I in an *indicator* function such that I(a,b) = 1 if a $\neq$ b and 0 otherwise

Score function for Regression

- Sum of squared errors:

$$S_{SSE}(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left(f\left(\mathbf{x}(i);\theta\right) - y(i)\right)^2$$

Issues

- Assume errors for all individuals are weighted equally
  - we may want to give more weight to recent observations
  - we may have other information about the reliability of sets of measurements
- Assume the error is the same, regardless of the context
  - we may want to weight false positives and false negatives differently. e.g. for predicting cancer, we may want to give more weight to not detecting a true cancer
  - in an investment scenario, we may be more tolerant of underestimates of the true value rather than overestimates

More examples

- loan decisions: cost of lending to a defaulter is much greater than the lost-business cost of refusing loan to non-defaulter
- oil-slick detection: cost of failing to detect an environment-threatening real slick is greater than the cost of a false alarm
- load forecasting: the cost of gearing up electricity generators for a storm that doesn't hit is far less then the cost of being caught unprepared
- diagnosis: the cost of misidentifying problems with a machine that turns out to be fault-free is less than the cost of overlooking problems
- promotional mailing: cost of sending junk mail to a household that doesn't respond is far less than lost business cost of not sending it to one that would have responded

Cost-sensitive Models

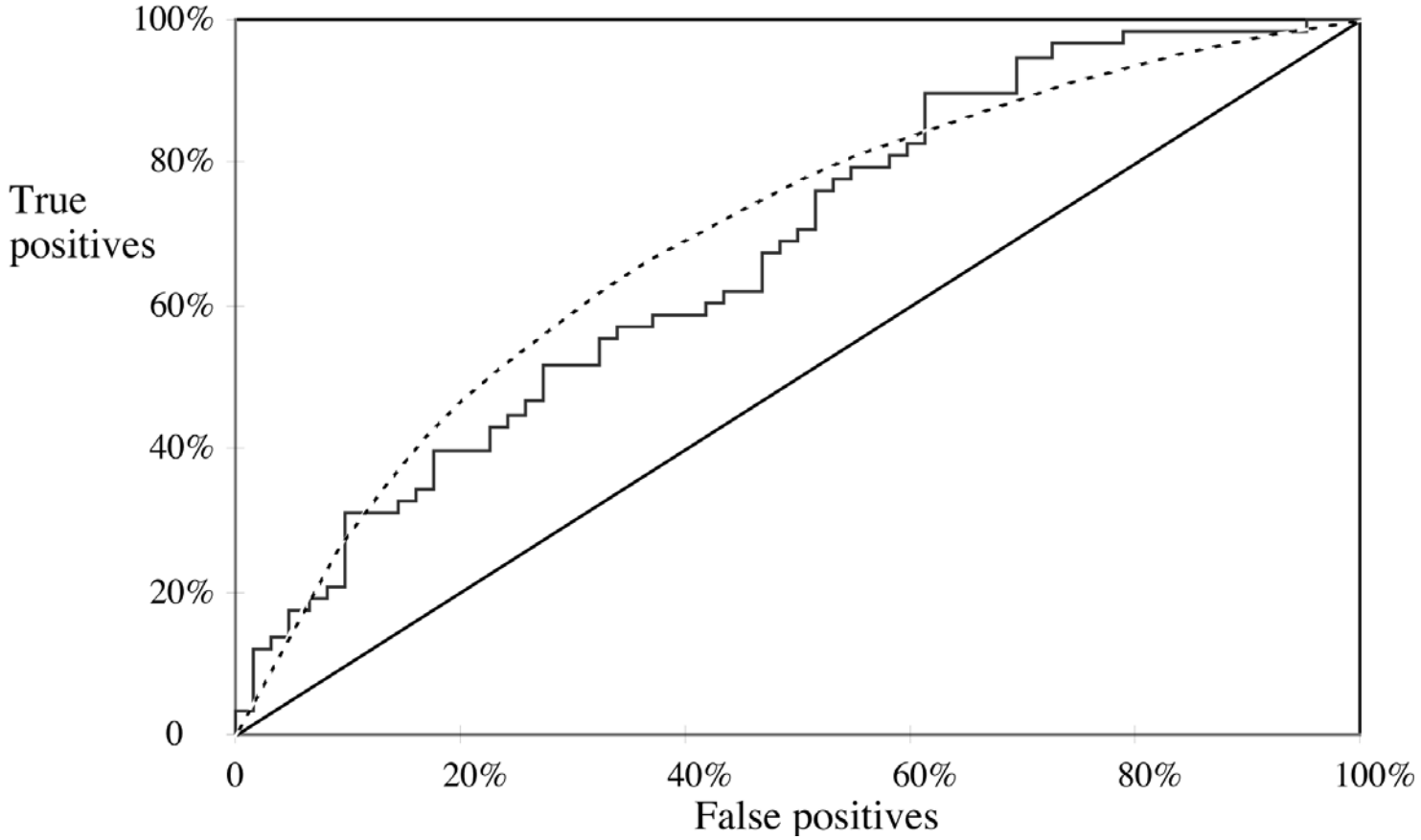
- Define a score function based on a cost matrix.
- If $\hat{y}$ is the predicted class and y is the true class define matrix of costs $C(\hat{y}, y)$
- reflects the severity of classifying a instance with true class y to class $\hat{y}$

		predicted class	
		yes	no
true class	yes	true positive	false negative
	no	false positive	true negative

ROC Curves

- What if you don’t know the cost????

Sample ROC curve



Receiver Operating Characteristic curve: plot of the true positive rate against the false positive rate for the different possible classification cutpoints

Classification

- Two general views:
  - discriminative
  - probabilistic

Discriminative Classification

- decision boundary
  - given model, we can draw decision regions
  - may seek a *discriminant* function f(**x**;θ) that maximizes measure of separation between classes
  - early example, Fisher's linear discriminant analysis: find a linear combination of the variables in x that maximally discriminates between two classes

Probabilistic Classification

- Model $p(c_k)$, the probability that a randomly chosen x belongs to class c_k
- Assuming classes are mutually exclusive and exhaustive

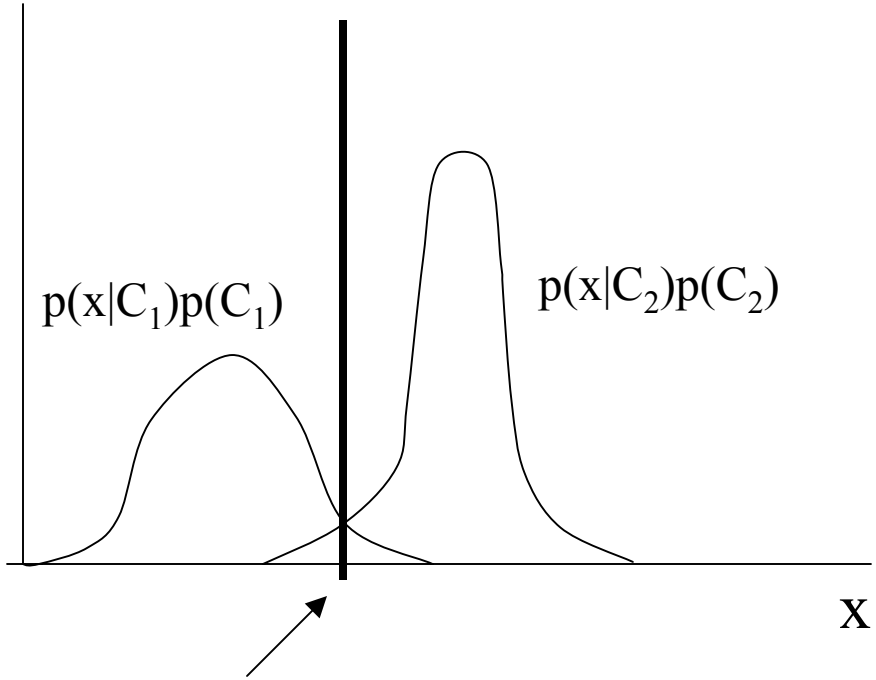
$$\sum_k p(c_k) = 1$$

- Using the training data, we learn a model for $p(\mathbf{x}|c_k,\theta)$

- Then, use Bayes rule to find posterior probabilities:

$$p(c_k \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid c_k, \theta) p(c_k)}{\sum_{k'} p(\mathbf{x} \mid c_{k'}, \theta) p(c_{k'})}$$

Two-class example



Bayes Error Rate

- The best we can hope for…

$$p_B^* = \int (1 - \max_k p(c_k \mid x)) p(x) dx$$

No other classifier can achieve a lower expected error rate on unseen new data

Lower-bound on best possible classifier for the problem

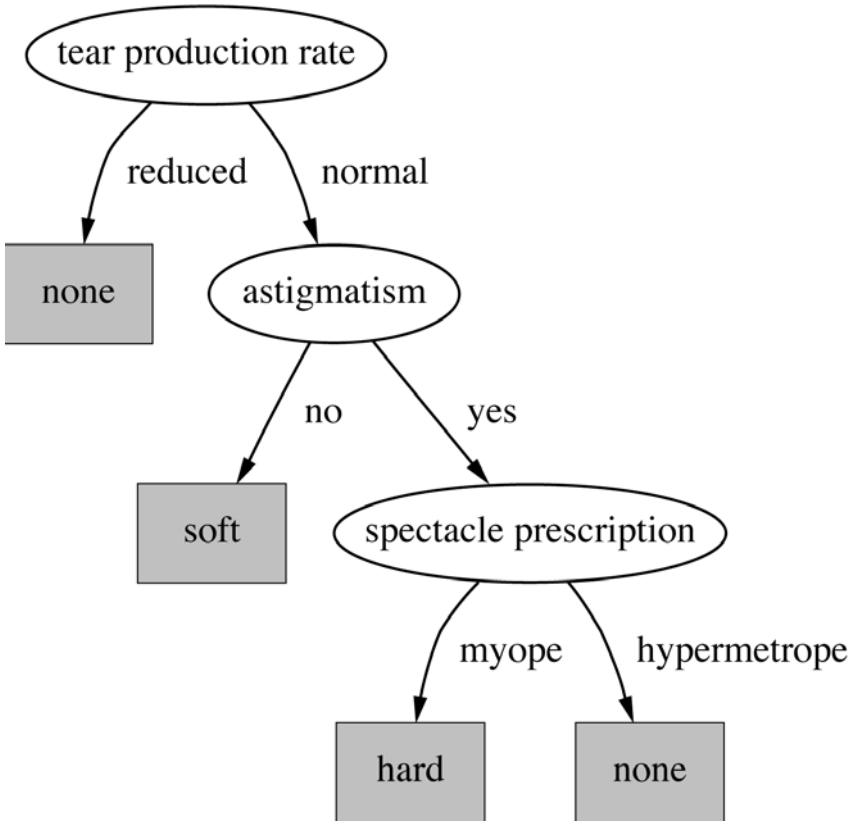
In practice…

- Methods for constructing classifiers
  1. Discrimitive approach - model decision boundary directly
     - Perceptrons, support vector machines
     - CART (if only class label at each leaf)
  2. Regression - posterior class probabilities $p(c_k|\mathbf{x})$ are modeled explicitly. For prediction, max (weighted by cost function) is chosen
     - Logistic regression
     - CART (if posterior class distribution at each leaf)
  3. Generative approach - class conditional probabilities $p(\mathbf{x}|c_k,\theta)$ and prior probabilities $p(c_k)$ are modeled explicitly. Sometimes referred to as Bayesian classifiers

Tree Models

- Idea: recursively partition the input space to maximize 'class purity' score - so that the majority of points in each cell of the partition belong to the same class
- Decision tree representation:
  - Each internal node tests an attribute
  - Each branch corresponds to a attribute value or test outcome
  - Each leaf assigns a classification
- Learning algorithm: At each step, consider splitting on any attribute. Choose the split that maximizes the score
- Once tree is build, to predict a class values for a new instance with known values of input variables, traverse the tree, at each node choosing the branch according to the outcome of the test at that node.
- Most well-known systems:
  - ID3, C4.5, Ross Quinlan
  - CART (Classification and Regression Trees), Breiman, Friedman, Olshen, Stone

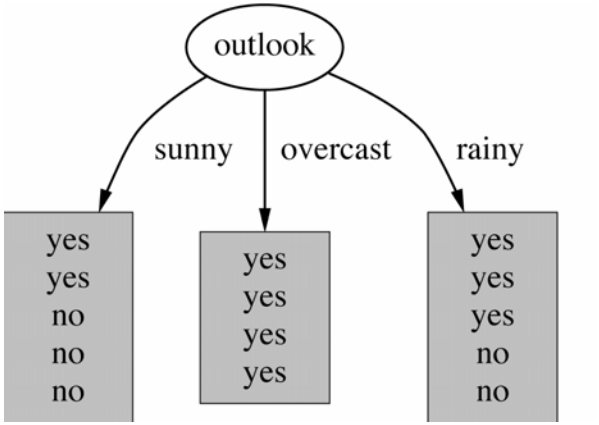
Example: Contact perscription

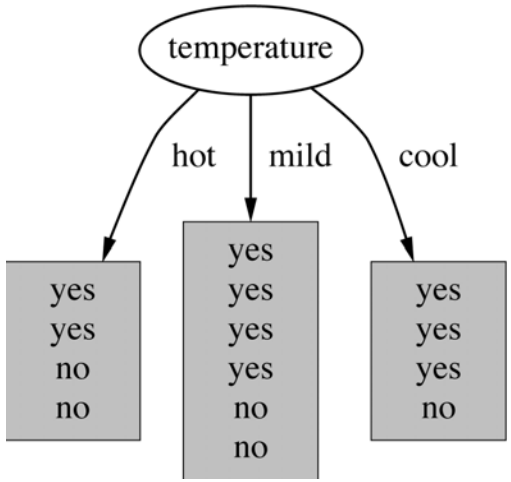


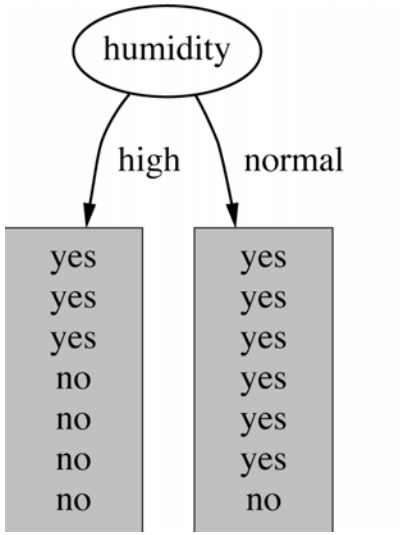
Weather/Play Example

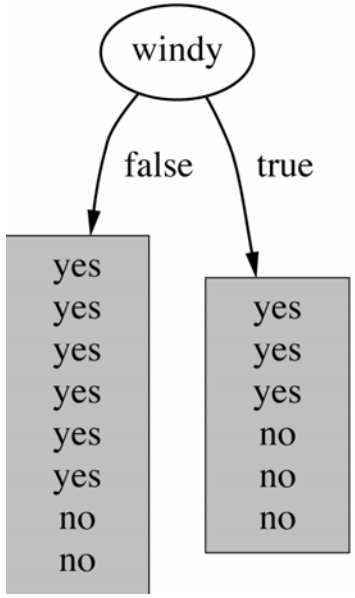
Outlook, temp, humidity, windy, play?
sunny, 85, 85, false, no
sunny, 80, 90, true, no
overcast, 83, 86, false, yes
rainy, 70, 96, false, yes
rainy, 68, 80, false, yes
rainy, 65, 70, true, no
overcast, 64, 65, true, yes
sunny, 72, 95, false, no
sunny, 69, 70, false, yes
rainy, 75, 80, false, yes
sunny, 75, 70, true, yes
overcast, 72, 90, true, yes
overcast, 81, 75, false, yes
rainy, 71, 91, true, no

First split


outlook
sunny
overcast
rainy
yes yes no no no
yes yes yes yes
yes yes yes no no


temperature
hot
mild
cool
yes yes no no
yes yes yes yes no no
yes yes yes no

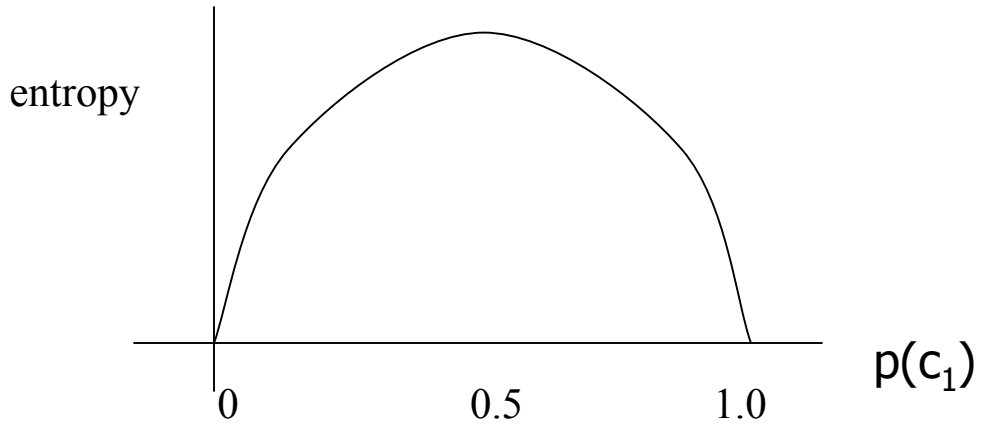

humidity
high
normal
yes yes yes no no no no
yes yes yes yes yes yes no


windy
false
true
yes yes yes yes yes yes no no
yes yes yes no no no

Scoring a Split

- Use entropy:

$$\text{Entropy}(X) = \sum_{x} -p(x)\log p(x)$$



- Two class example: $p(c_1)$ is probability of c_1

$$H(X) = -p(c_1)\log p(c_1) + -p(1-c_1)\log p(1-c_1)$$

Entropy cont.

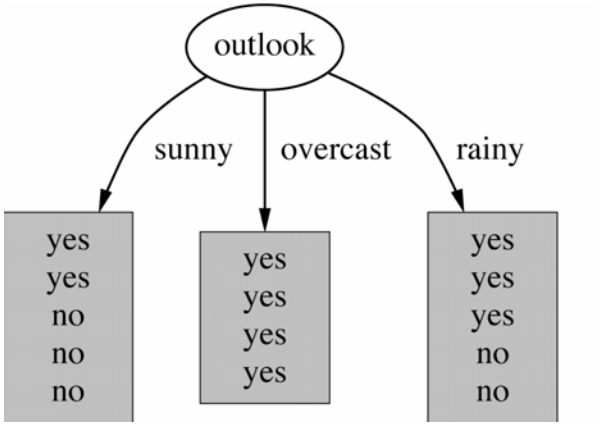
- Entropy(X) is expected number of bits to encode a randomly chosen x
- Information theory: optimal coding scheme uses $\log_2 p$ bits to encode message with probability p

Scoring a Split

- Information Gain of splitting set S on attribute A:

$$\text{Gain}(S, A) = \text{Entropy}(S) - \sum_{v \in \text{Values}(A)} \frac{|S_A|}{|S|} \text{Entropy}(S_A)$$

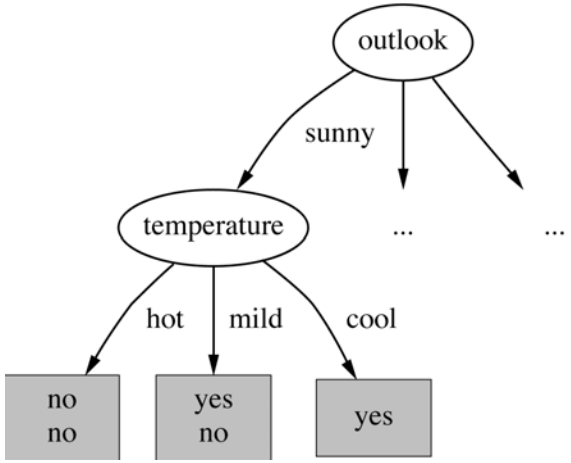
Example

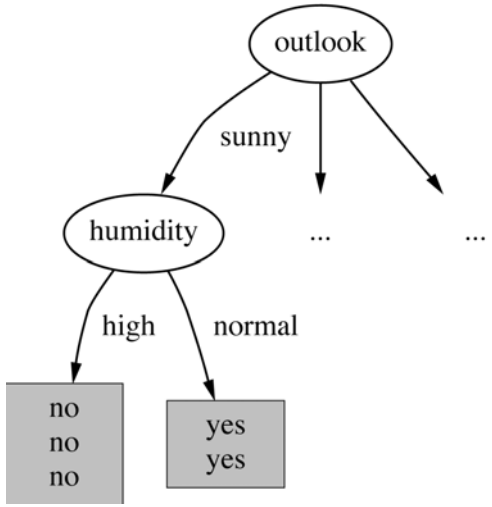


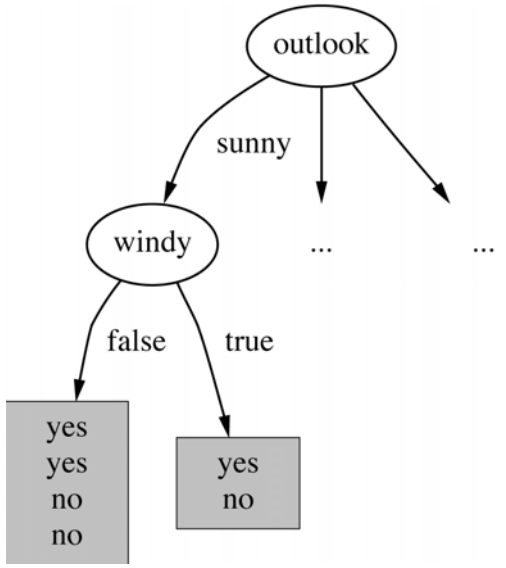
Gain(outlook) = -9/14 log 9/14 + -5/14 log 5/14 -
(5/14 (-2/5 log 2/5 + -3/5 log 3/5) +
4/14 (-4/4 log 4/4 + -0/4 log 0/4) +
5/14 (-3/5 log 3/5 + -2/5 log 2/5))
= 0.247

Gain(temperature) = 0.029, Gain(humidity) = 0.152, Gain(windy) = 0.048

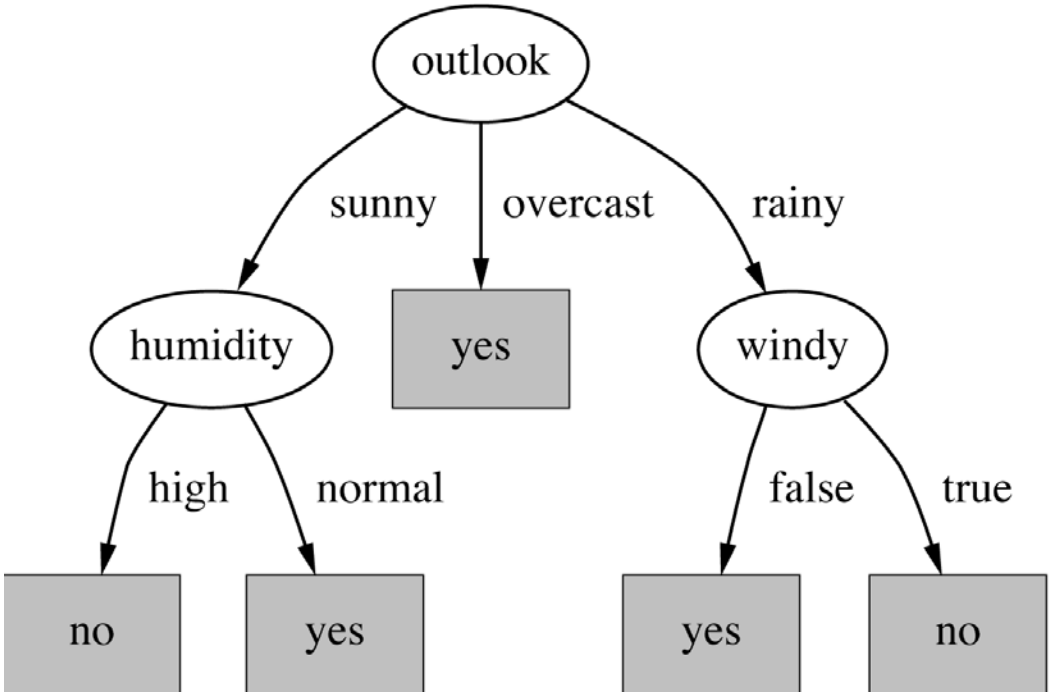
Next Split….


outlook
sunny
temperature
...
...
hot
mild
cool
no
no
yes
no
yes


outlook
sunny
humidity
...
...
high
normal
no
no
no
yes
yes


outlook
sunny
windy
...
...
false
true
yes
yes
no
no
yes
no

Final Tree

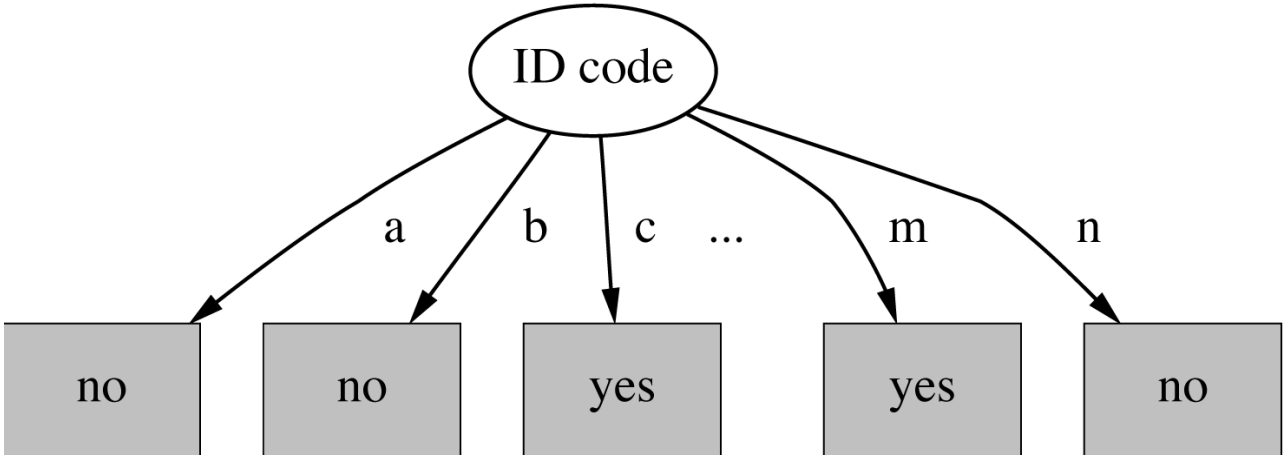

outlook
sunny
overcast
rainy
humidity
yes
windy
high
normal
false
true
no
yes
yes
no

Issues

- Overfitting:
  - Stopping criteria
    - Problem: gain non-monotonic; a poor step may be followed by a large gain
    - Most common strategy: build large tree and then prune it back. I.e. merge leaf nodes that leads to least reduction in predictive performance on the training set
  - Average predictions obtained by the leaves and the nodes leading to the leaves
  - Model averaging. The decision tree chosen is notoriously sensitive to small changes in the input data; a small change can result is a very different tree structure. To overcome this problem, average over multiple perterbations of the data to reduce variance

Problem

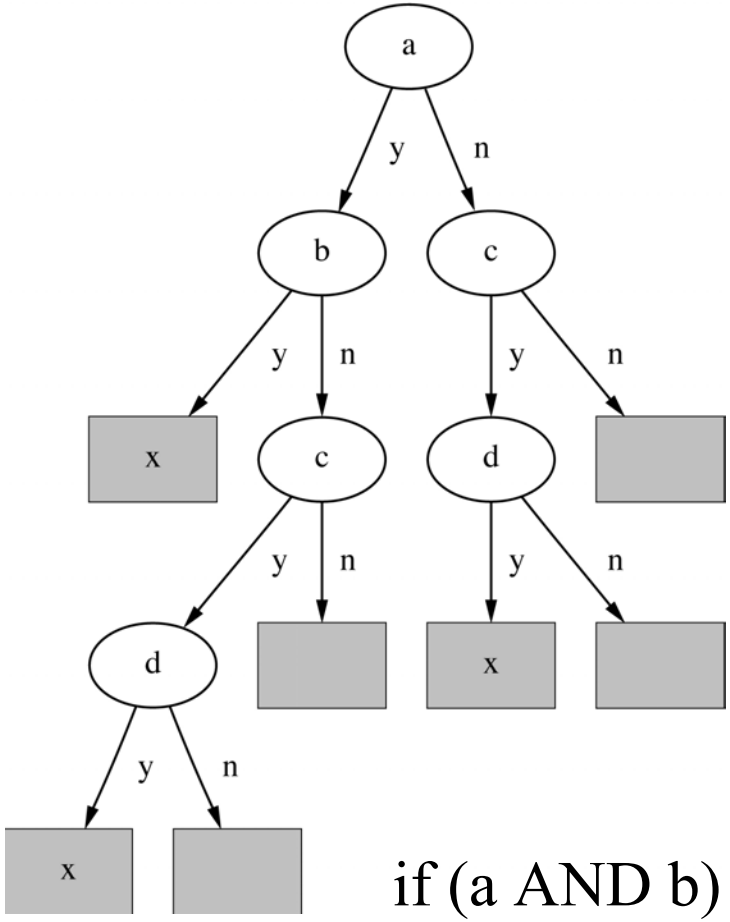
When we have some attributes with a large number of possible values:



Add ID code. Perfect predictor. Largest Info Gain

Take into account the number and size of children nodes using Information value of the attribute: - 1/14 log 1/14 * 14
Gain ratio is ratio of entropy and information value of attribute

Representing Disjunctions



if (a AND b) or (c AND d) then x

Computational complexity

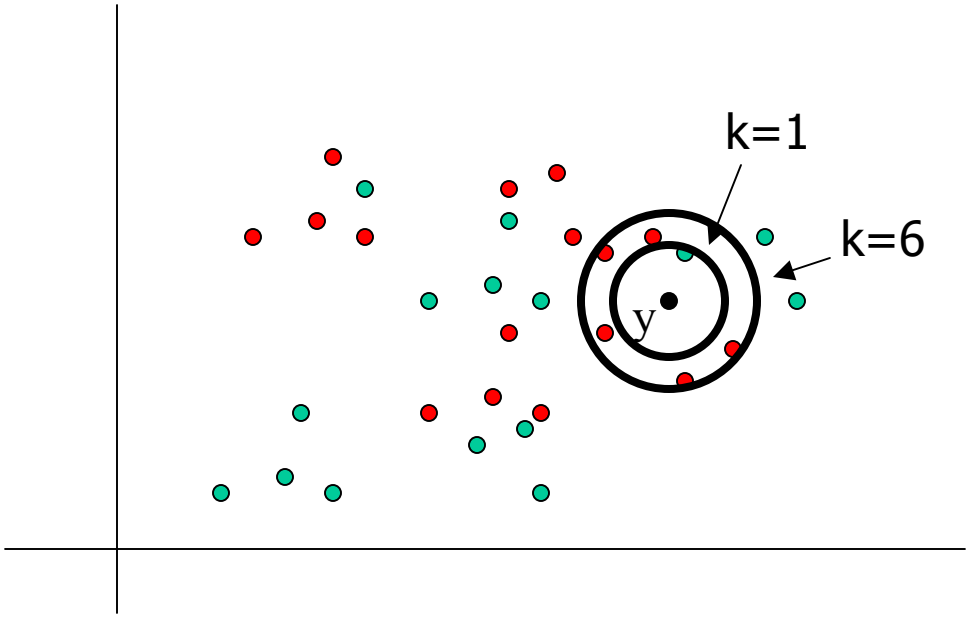
- Choosing splits
- Number of passes over data: if data does not fit into main memory can be expensive
  - Tailor algorithm to minimize seconday memory access
  - Resort to working with a random sample that can fit into main memory

Handling Missing Values

- Which branch to take?
- Treat ‘missing value’ as an attribute value
- Keep track of the number of elements in training set that go down each branch and use the most popular branch
- Split instance and weight each path down the tree by the proportion in the training set

Nearest Neighbor Methods

- To classify a new input vector y, examine the k-closest training data points to y and assign the obtect to the most frequently occurring class



Issues

- Distance measure
  - Most common: euclidean
- Choosing k
  - Increasing k reduces variance, increases bias
- For high-dimensional space, problem that the nearest neighbor may not be very close at all!
- Memory-based technique. Must make a pass through the data for each classification. This can be prohibitive for large data sets.

KNN Advantages

- Easy to program
- No optimization or training required
- Classification accuracy can be very good; can outperform more complex models
- Easy to add *reject* option
- Easy to handle missing values

Naïve Bayes

References

- *Principles of Data Mining*, Hand, Mannila, Smyth. MIT Press, 2001.
- *Data Mining*, Witten and Eibe. Academic Press, 2000.
- *Machine Learning*, T. Mitchell. McGraw-Hill, 1997.