Machine Learning

Learning From Data with Decision Trees

bad news …

Supervised (Function) Learning

- $y = F(x_1 \ldots x_n)$: true function (usually not known)
- D: training sample drawn from F(x)

57,M,195,0,125,95,39,25,0,1,0,0,0,1,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0	0
78,M,160,1,130,100,37,40,1,0,0,0,1,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0	1
69,F,180,0,115,85,40,22,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0	0
18,M,165,0,110,80,41,30,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	0
54,F,135,0,115,95,39,35,1,1,0,0,0,1,0,0,0,1,0,0,0,0,1,0,0,0,1,0,0,0,0	1
84,F,210,1,135,105,39,24,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0	0
89,F,135,0,120,95,36,28,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,1,0,0	1
49,M,195,0,115,85,39,32,0,0,0,1,1,0,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0	0
40,M,205,0,115,90,37,18,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	0
74,M,250,1,130,100,38,26,1,1,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0	0
77,F,140,0,125,100,40,30,1,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,1	1

…

Big Picture: Supervised Learning

Train Set:

57,M,195,0,125,95,39,25,0,1,0,0,0,1,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0	0
78,M,160,1,130,100,37,40,1,0,0,0,1,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0	1
69,F,180,0,115,85,40,22,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0	0
18,M,165,0,110,80,41,30,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	0
54,F,135,0,115,95,39,35,1,1,0,0,0,1,0,0,0,1,0,0,0,0,1,0,0,0,1,0,0,0,0	1
84,F,210,1,135,105,39,24,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0	0
89,F,135,0,120,95,36,28,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,1,0,0	1
49,M,195,0,115,85,39,32,0,0,0,1,1,0,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0	0
40,M,205,0,115,90,37,18,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	0
74,M,250,1,130,100,38,26,1,1,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0	0
77,F,140,0,125,100,40,30,1,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,1	1

…

Test Set:

71,M,160,1,130,105,38,20,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0	?

Big Picture: Supervised Learning

- F(x): true function (usually not known)
- D: training sample drawn from F(x)

```
57,M,195,0,125,95,39,25,0,1,0,0,0,1,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0    0
78,M,160,1,130,100,37,40,1,0,0,0,1,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0   1
69,F,180,0,115,85,40,22,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0    0
18,M,165,0,110,80,41,30,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0    0
54,F,135,0,115,95,39,35,1,1,0,0,0,1,0,0,0,1,0,0,0,0,1,0,0,0,1,0,0,0,0  1
```

- G(x): model learned from training sample D

```
71,M,160,1,130,105,38,20,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0   ?
```

- Goal: $E<(F(x)-G(x))^2>$ is small (near zero) for future test samples drawn from F(x)
- Think of supervised learning as super regression

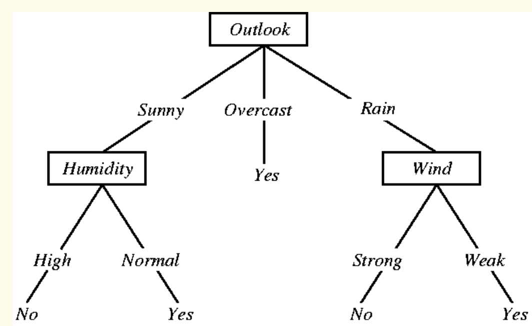
What Data Structure(s) for Data?

- 2-D array
  - T training cases
  - N attributes per case
- List of vectors
  - N attributes per vector
- Attributes of different types:
  - Boolean: 0/1
  - Nominal: chevrolet, chrysler, ford, subaru, toyota, volvo, …
  - Integer: 0, 1, 2, 3, …
  - Ordinal: low, medium, high
  - Continuous: [0, 1], [0, 100], [-1000, +1000], …
- In languages like C, sometimes coded as floats/ints (with hash tables to look up values). JAVA is better for this.

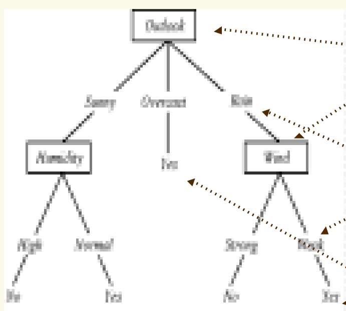
Decision Trees

(Neural Nets, Support Vector Machines, …)

A Simple Decision Tree for **Slope Day**



A Decision Tree is Represented as a Tree

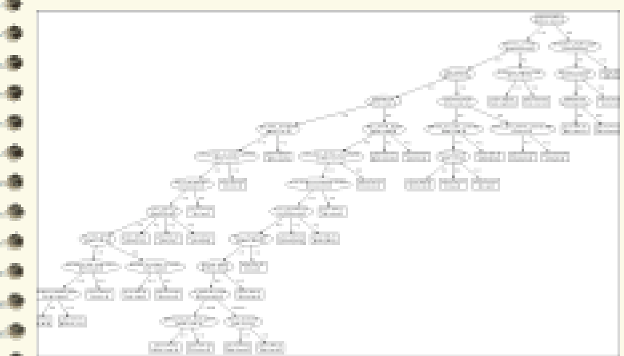


internal node = *attribute test*

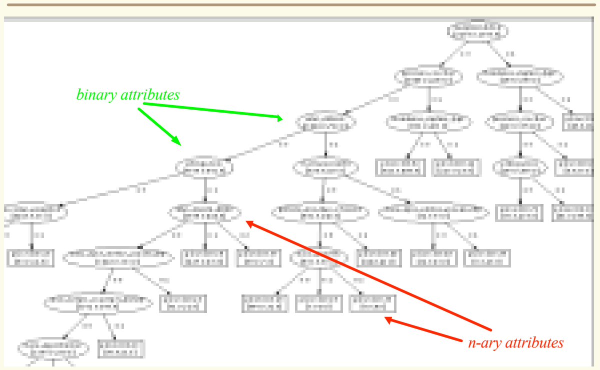
branch = *attribute value*

leaf node = *classification*

A Real Decision Tree



A Real Decision Tree



what about continuous attributes such as height, weight, blood pressure, temperature?

A Real (but very small) Decision Tree

Decision Tree Trained on 1000 Patients:

```
+833+167 (tree) 0.8327 0.1673 0
fetal_presentation = 1: +822+116 (tree) 0.8759 0.1241 0
|  previous_csection = 0: +767+81 (tree) 0.904 0.096 0
|  |  primiparous = 0: +399+13 (tree) 0.9673 0.03269 0
|  |  primiparous = 1: +368+68 (tree) 0.8432 0.1568 0
|  |  |  fetal_distress = 0: +334+47 (tree) 0.8757 0.1243 0
|  |  |  |  birth_weight < 3349: +201+10.555 (tree) 0.9482 0.05176 0
|  |  |  |  birth_weight >= 3349: +133+36.445 (tree) 0.783 0.217 0
|  |  |  fetal_distress = 1: +34+21 (tree) 0.6161 0.3839 0
|  previous_csection = 1: +55+35 (tree) 0.6099 0.3901 0
fetal_presentation = 2: +3+29 (tree) 0.1061 0.8939 1
fetal_presentation = 3: +8+22 (tree) 0.2742 0.7258 1
```

What Data Structures(s) for DTree?

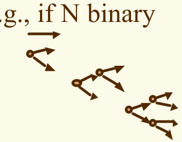
- Not binary tree
- N-ary tree
  - Left child and Right siblings
  - List of child nodes
- Info at each node:
  - Number of cases of each class
  - List pointing to cases at that node?
  - Array for cases at that node?
  - Attribute tested at node
  - Node's prediction

Computational Complexity?

- N attributes
- T training cases
- How expensive is it to "grow" a tree?

Generate & Test ***All*** Trees to Find Best

- all possible sequences of all possible tests
- very large search space, e.g., if N binary attributes:
  - 1 null tree
  - N trees with 1 (root) test
  - N*(N-1) trees with 2 tests
  - N*(N-1)*(N-1) trees with 3 tests (if balanced)
  - $O(N^4)$ trees with 4 tests
  - maximum height of tree is N
  - $O(N^N)$!!!!!



Real Data, Real People, Real Trees: C-Section Prediction

Demo summary:

- Fast
- Reasonably intelligible
- Larger training sample => larger tree
- Different training sample => different tree

collaboration with Magee Hospital, Siemens Research, Tom Mitchell

Recursive Induction of Decision Trees

- TDIDT (Top-Down Induction of Decision Trees)
- Greedy Tree Growing
- Recursive Partitioning
  - *find "best" attribute test to install at root*
  - *split data on root test*
  - *find "best" attribute test to install at each new node*
  - *split data on new test*
  - *repeat until:*
    - all nodes are pure
    - all nodes contain fewer than k cases
    - distributions at nodes indistinguishable from chance
    - tree reaches predetermined max depth
    - no more attributes to test

Recursive Partitioning PseudoCode

```
public static boolean split (dataList d, attributeList a) {
  if (d == null ||
      a == null ||
      countPos(d) == 0 ||
      countNeg(d) == 0)
  return false;
  else {
      float bestPerf = 0; attribute bestAttr = null;
      for (attribute attr = a; attr != null; attr = attr.getNext())
          if (a.perf(d) > bestPerf)
            { bestPerf = a.perf(d);  bestAttr = a; }
      if (bestAttr != null) {
          split (left  (d, bestAttr), remove(a, bestAttr));
          split (right(d, bestAttr), remove(a, bestAttr));
         return true;
      } else return false;
```

Complexity of Recursive Partitioning?

- N attributes (usually N < 1000)
- T training cases (typically 100 < T < 1,000,000)

- Best Case: ~ $O(N^2)$, $O(T\log T)$
- Worst Case: ~ $O(N^2)$, $O(T^2)$

- With clever data structures, can reduce to $O(T\log T)$

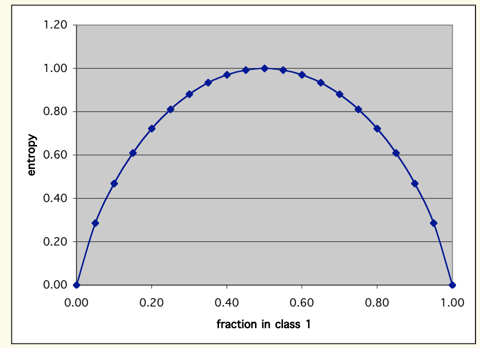
Splitting Rules

- Information Gain = reduction in entropy due to splitting on an attribute
- Entropy = expected number of bits needed to encode the class of a randomly drawn + or – example using the optimal info-theory coding

$$Entropy = -p_+ \log_2 p_+ - p_- \log_2 p_-$$

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

Entropy



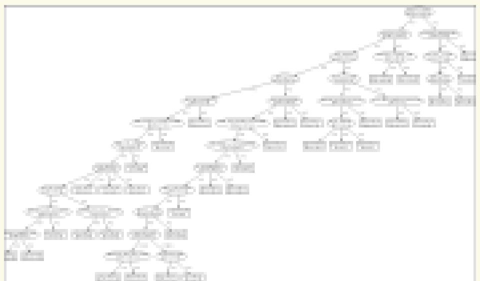
Greedy vs. Optimal

- Optimal
  - Maximum expected accuracy (test set)
  - Minimum size tree
  - Minimum depth tree
  - Fewest attributes tested
  - Easiest to understand
- XOR problem!

Attribute Types

- Boolean
- Nominal
- Ordinal
- Integer
- Continuous
  - Sort by value, then find best threshold for binary split
  - Cluster into n intervals and do n-way split

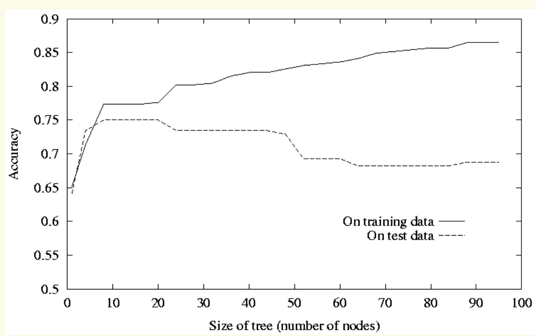
Decision Trees are Intelligible



Not *ALL* Decision Trees Are Intelligible

Part of Best Performing C-Section Decision Tree

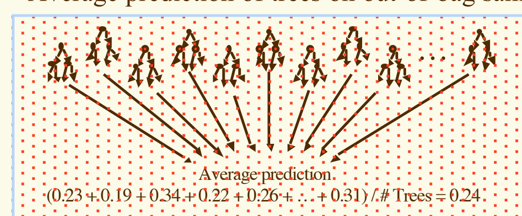
Overfitting



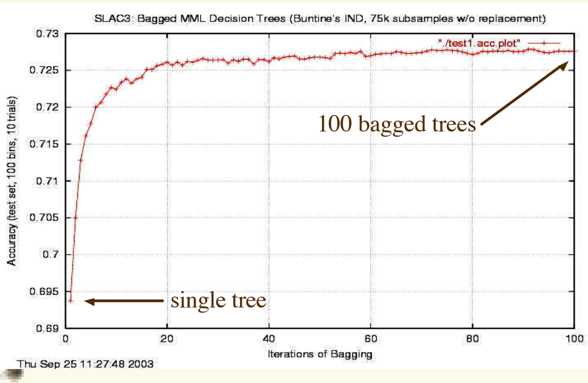
©Tom Mitchell, McGraw Hill, 1997

Bagged Decision Trees

- Draw 100 bootstrap samples of data
- Train trees on each sample -> 100 trees
- Average prediction of trees on out-of-bag samples



Bagging Results



Summary of Decision Tree Learning

- Arrays
- Lists
- Trees
- Computational complexity (big-O)
- Recursion
- Hash tables
- Advantages of JAVA over C
  - Arrays/lists of different types: boolean, nominal, ordinal, integer, continuous

Computer Science

- CS is not programming
  - most machine learning textbooks don't include code or discuss programming or data structures. sometimes discuss algorithms
  - what math is to engineering, programming is to computer science
  - many computer scientists feel that they don't get to do enough programming
- In 20 years, we'll have as much raw computational power as a human brain in a single computer
- Build "things" millions of people will use
- CS is the driving force behind the internet, the largest social change in 100 years (since the telephone)

Computer Science

- CS is many things: system design, database design, understanding user needs, human factors engineering, information retrieval, , …
- Computers change everything they touch
- CS is becoming critical to every field of human endeavor
  - NASA, space travel, Hubble space telescope, …
  - Genomics and Bioinformatics
  - Medicine
  - Meteorology
  - Engineering: mechanical, electrical, civil, aeronautical…
  - Libraries
  - Games and Entertainment
  - Arts
  - …

THE END

Thank You!

Good Luck in your finals

Have a good summer.