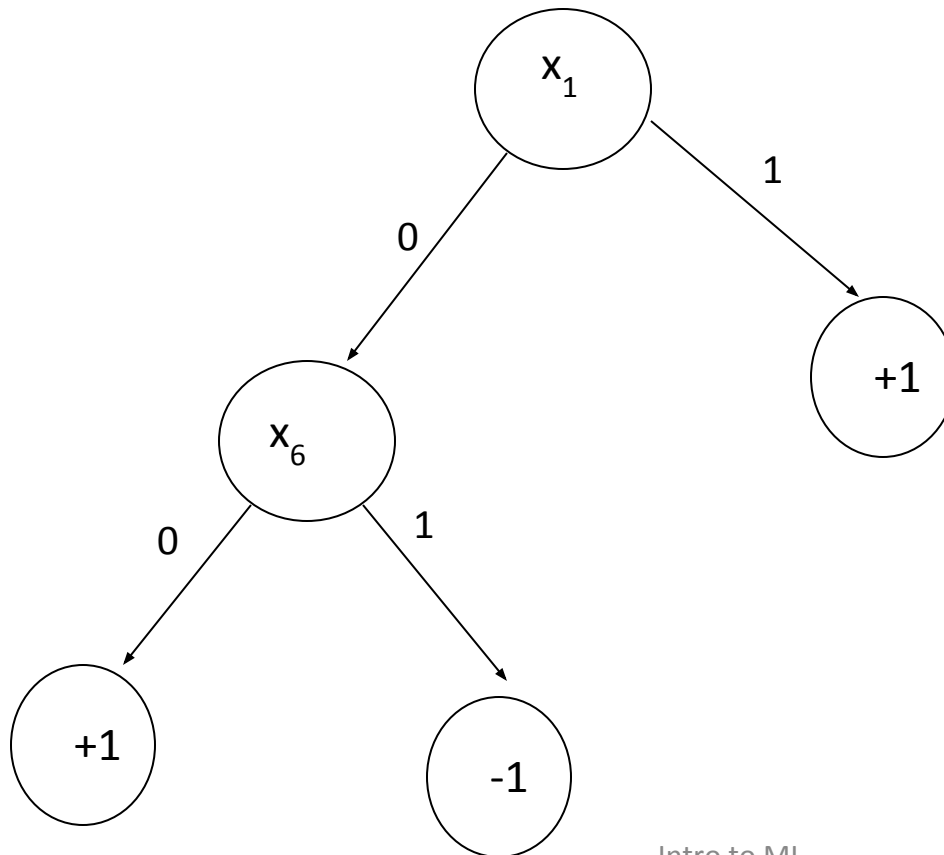


# Decision Trees

# Decision Trees

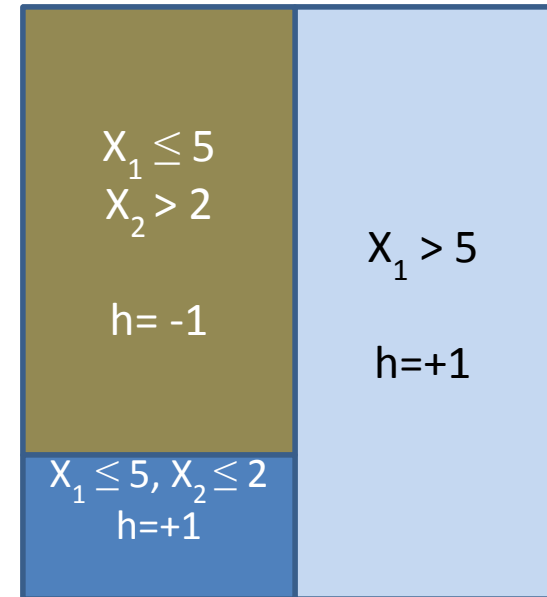
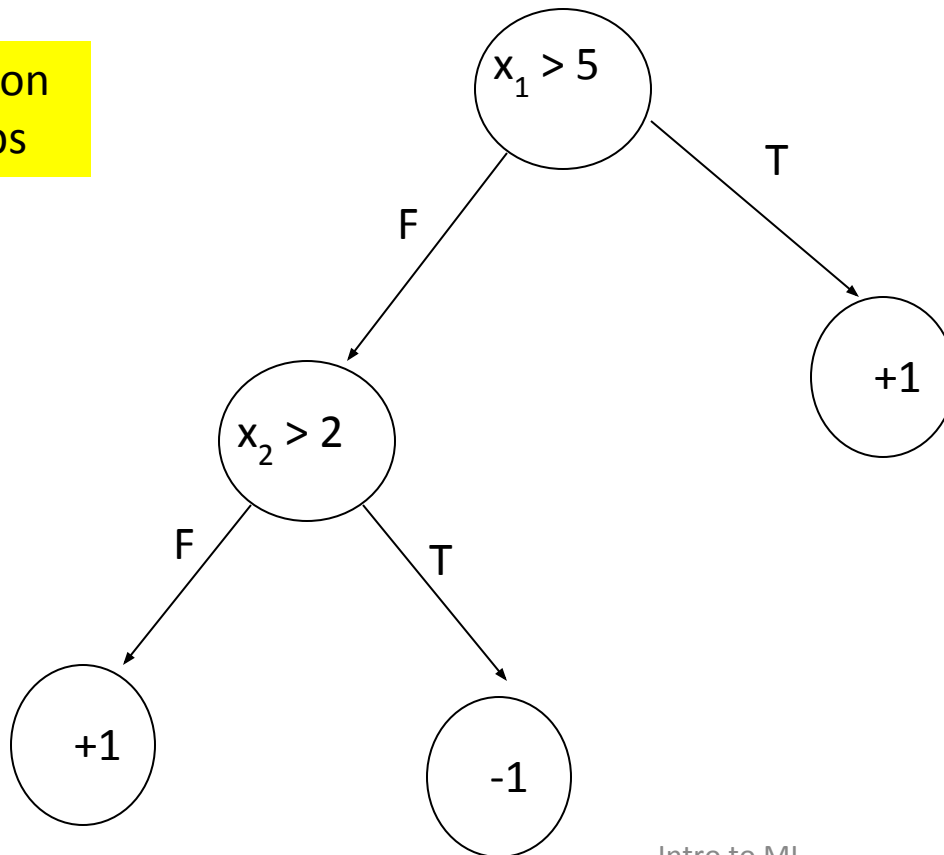
- This week:
  - Algorithms for constructing DT
- Next week:
  - Pruning DT
  - Ensemble methods
    - Random Forest

# Decision Trees - Boolean



# Decision Trees - Continuous

Decision  
stumps




# Decision Trees: Basic Setup

- Basic class of hypotheses  $H$ .
  - For example  $H=\{x_i\}$  or  $H=\{x_i>a\}$
- Input: Sample of examples
  - $S=\{(x,b)\}$
- Output: Decision tree
  - Each internal node from  $H$
  - Each leaf a classification value
- Goal (Occam Razor):
  - Small decision tree
  - Classifies all (most) examples correctly.

# Decision Tree: Why?

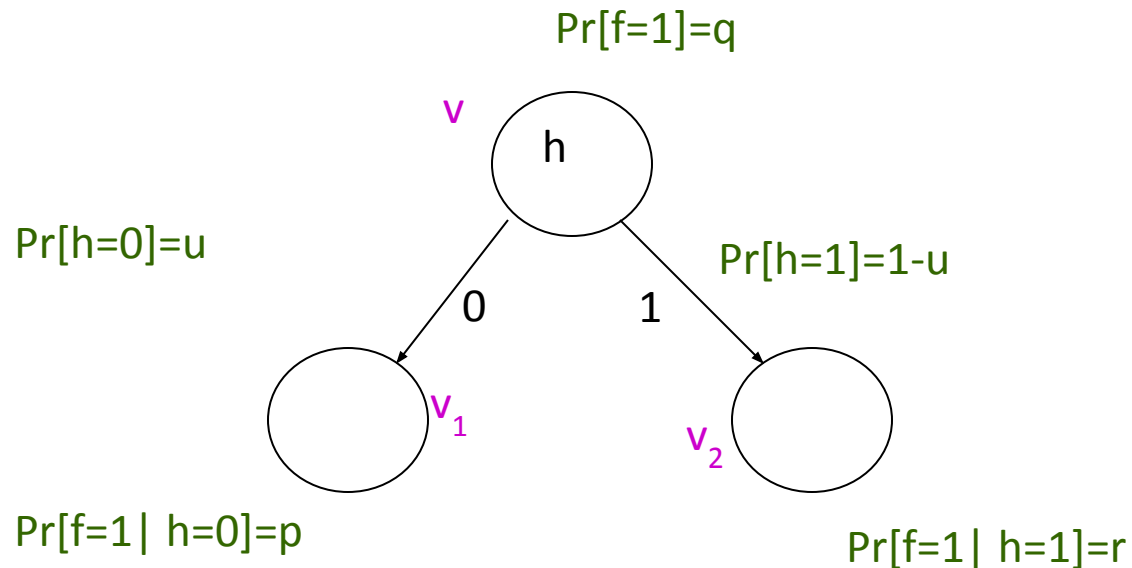
- Human interpretability
- Efficient algorithms:
  - Construction.
  - Classification
- Performance: Reasonable
- Software packages:
  - CART
  - C4.5 and C5

# Decision Trees Algorithm: Outline

- A natural recursive procedure.
- Decide a predicate  $h$  at the root. 
- Split the data using  $h$
- Build right subtree (for  $h(x)=1$ )
- Build left subtree (for  $h(x)=0$ )
- Running time
  - $\text{Time}(m) = O(m) + \text{Time}(m^+) + \text{Time}(m^-) \approx O(m \log m)$ 
    - Tree size  $< m = \text{sample size}$

# DT: Selecting a Predicate

- Basic setting:



- Clearly:  $q=up + (1-u)r$



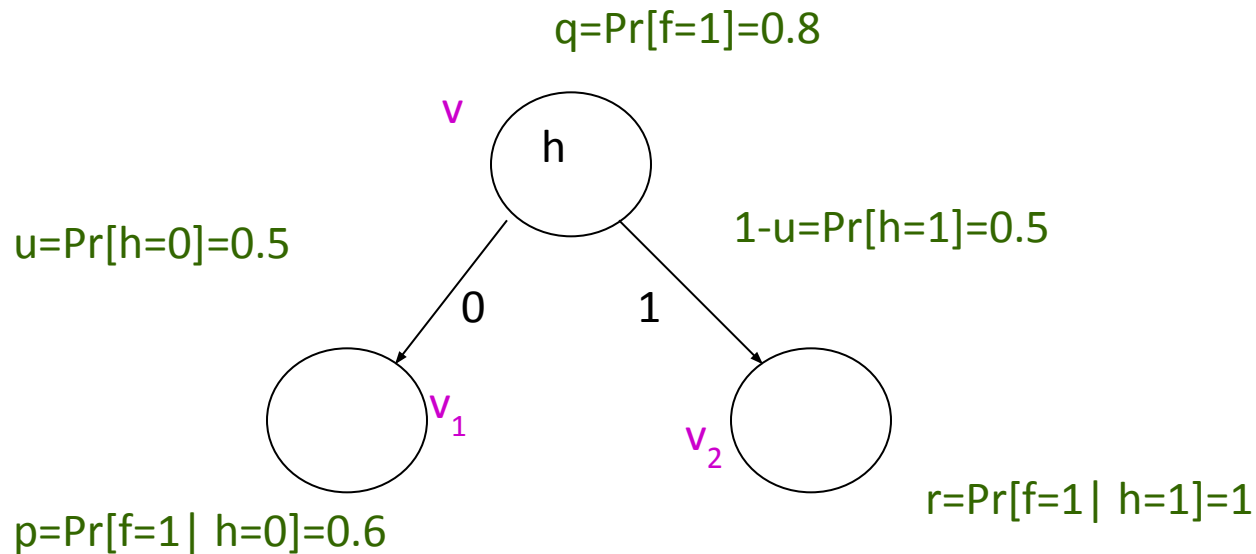
# Potential function: setting

- Compare predicates using potential function.
  - Inputs:  $q, u, p, r$
  - Output: value
- Node dependent:
  - For each node  $v$  and predicate  $h$  assign a value.
    - $\text{val}(v) = \text{val}(u, p, r)$ 
      - Q: what about  $q$ ?! What about the probability of reaching  $v$  ?!
  - Given a split:  $\text{val}(v) = u \text{val}(v_1) + (1-u) \text{val}(v_2)$
  - For a tree: weighted sum over the leaves.
    - $\text{Val}(T) = \sum_{v \text{ leaf}} q_v \text{val}(v)$

# PF: classification error

- Misclassification potential
  - $\text{val}(v) = \min\{q, 1-q\}$ 
    - Classification error.
    - $\text{val}(T) =$  fraction of errors using  $T$  on sample  $S$ 
      - In leaves, select the error minimizing label
- Termination:
  - Perfect Classification
    - $\text{Val}(T) = 0$
- Dynamics: The potential only drops

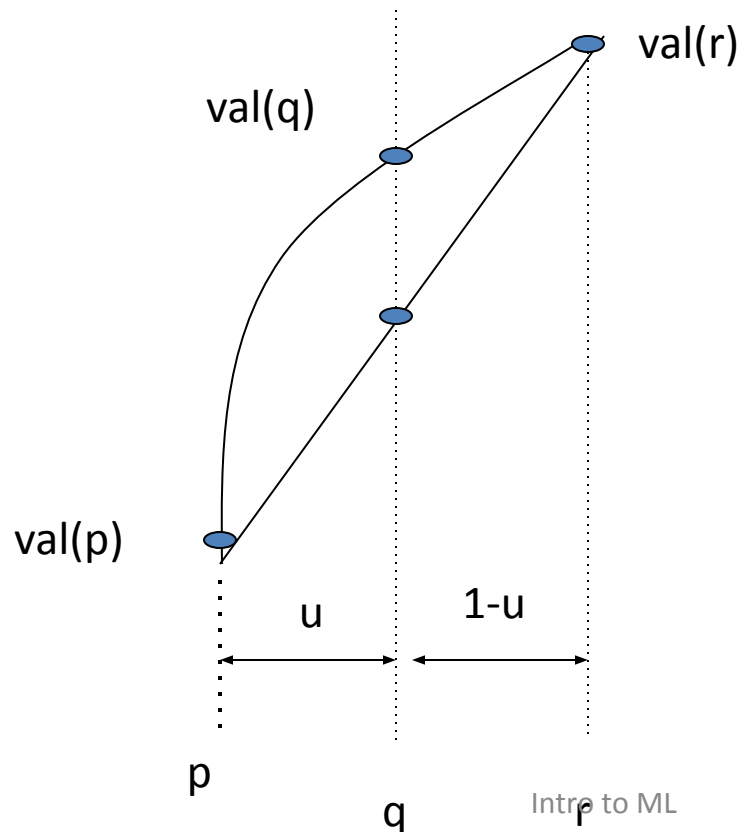
# PF: classification error



- Initial error **0.2**
- After split  **$0.5 (0.4) + 0.5(0) = 0.2$**
- **Is this a good split?**

# Potential Function: requirements

- Every change is an improvement
- When zero perfect classification.
- Strictly concave.



# Potential Functions: Candidates

- Assumption on  $\text{val}$ :
  - Symmetric:  $\text{val}(q) = \text{val}(1-q)$
  - Convex
  - $\text{val}(0)=\text{val}(1) = 0$  and  $\text{val}(1/2) = 1/2$
- Outcome:
  - $\text{Error}(T) \leq \text{val}(T)$
  - Minimizing  $\text{val}(T)$  upper bounds the error!

# Potential Functions: Candidates

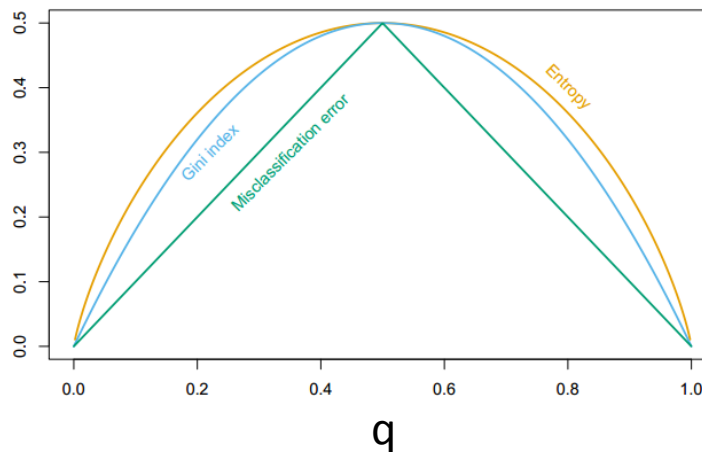
- Potential Functions:

- $\text{val}(q) = \text{Gini}(q) = 2q(1-q)$

CART

- $\text{val}(q) = \text{Entropy}(q) = -q \log q - (1-q) \log (1-q)$

C4.5



\*The entropy is scaled to fit the [0, 0.5] requirement

# Potential Functions: Candidates

- Potential Functions:
  - $\text{val}(q) = \text{Gini}(q) = 2q(1-q)$  **CART**
  - $\text{val}(q) = \text{Entropy}(q) = -q \log q - (1-q) \log (1-q)$  **C4.5**
- Differences:
  - Slightly different behavior
  - Same high level intuition

# DT: Construction Algorithm

Procedure DT(S) : S - sample

- If all the examples in S have the classification b
  - Create a leaf with label b and return
- For each h compute  $\text{val}(h,S)$ 
  - $\text{val}(h,S) = u_h \text{val}(p_h) + (1-u_h) \text{val}(r_h)$
- Let  $h' = \arg \min_h \text{val}(h,S)$
- Split S using  $h'$  to  $S^0$  and  $S^1$
- Recursively invoke DT( $S^0$ ) and DT( $S^1$ )
- **Termination: when S is pure; other criteria possible**



# Run of the algorithm

- Function  $val=2q(1-q)$
- Basic hypothesis: attrib.
- Initially:  $val = 0.5$
- At the root:
  - $X_1: (8,5) \& (2,0)$ 
    - $Val= 0.8*2*(5/8)(3/8)+0.2*0 = 0.375$
  - $X_2: (2,2) \& (8,3)$ 
    - $Val=0.2*0+0.8*2*3/8*5/8=0.375$

Reads (8pass,5pos), etc

- Example

x	- y	x	- y
11110	- 1	10011	- 0
10010	- 1	10111	- 0
11111	- 1	10011	- 0
10001	- 1	00100	- 0
10101	- 1	00000	- 0

# Run of the algorithm

- At the root:
  - $X_3: (5,3) \& (5,2)$ 
    - $\text{Val} = 0.5 * 2 * 3 / 5 * 2 / 5 + 0.5 * 2 * 2 / 5 * 3 / 5 = 0.48$
  - $X_4: (6,3) \& (4,2)$ 
    - $\text{Val} = 0.6 * 2 * 0.5 * 0.5 + 0.4 * 2 * 0.5 * 0.5 = 0.5$
  - $X_5: (6,3) \& (4,2)$ 
    - $\text{Val} = 0.5$

- Select  $x_1$ 
  - Reduction:  
 $0.5 - 0.375 = 0.125$

- Example

x	- y	x	- y
11110	- 1	10011	- 0
10010	- 1	10111	- 0
11111	- 1	10011	- 0
10001	- 1	00100	- 0
10101	- 1	00000	- 0

# Run of the algorithm

- Root  $x_1$
- Split the sample
- For  $x_1=0$  DONE ! (why?)
- For  $x_1=1$  continue.
  - What about  $\text{val}(x_1)$ ?!
  - For  $x_2$  (2,2) & (6,3) 0.375
  - For  $x_3$  (4,3) & (4,2) 0.4375
  - For  $x_4$  (6,3) & (2,2) 0.375
  - For  $x_5$  (6,3) & (2,2) 0.375
- **Select  $x_2$** 
  - Reduction
  - $0.375 - 0.8 * 0.375 = 0.015$

- Example

<u><math>x_1 = 1</math></u>	<u><math>x_1 = 0</math></u>
$x$ - $y$	$x$ - $y$
11110 - 1	00100 - 0
10010 - 1	00000 - 0
11111 - 1	
10001 - 1	
10101 - 1	
10011 - 0	
10111 - 0	
10011 - 0	

# Run of the algorithm

- Node  $x_2$
- Split the sample
- For  $x_2=1$  DONE !
- For  $x_2=0$  continue.
  - For  $x_3$  (2,1) & (4,2) 0.5
  - For  $x_4$  (3,1) & (3,2) 0.444
  - For  $x_5$  (5,2) & (1,1) 0.4
- **Select  $x_5$**

- Example

<u><math>x_2 = 1</math></u>	<u><math>x_2 = 0</math></u>
x - y	x - y
11110- 1	10010- 1
11111- 1	10001- 1
	10101 -1
	10011 - 0
	10111 - 0
	10011 - 0

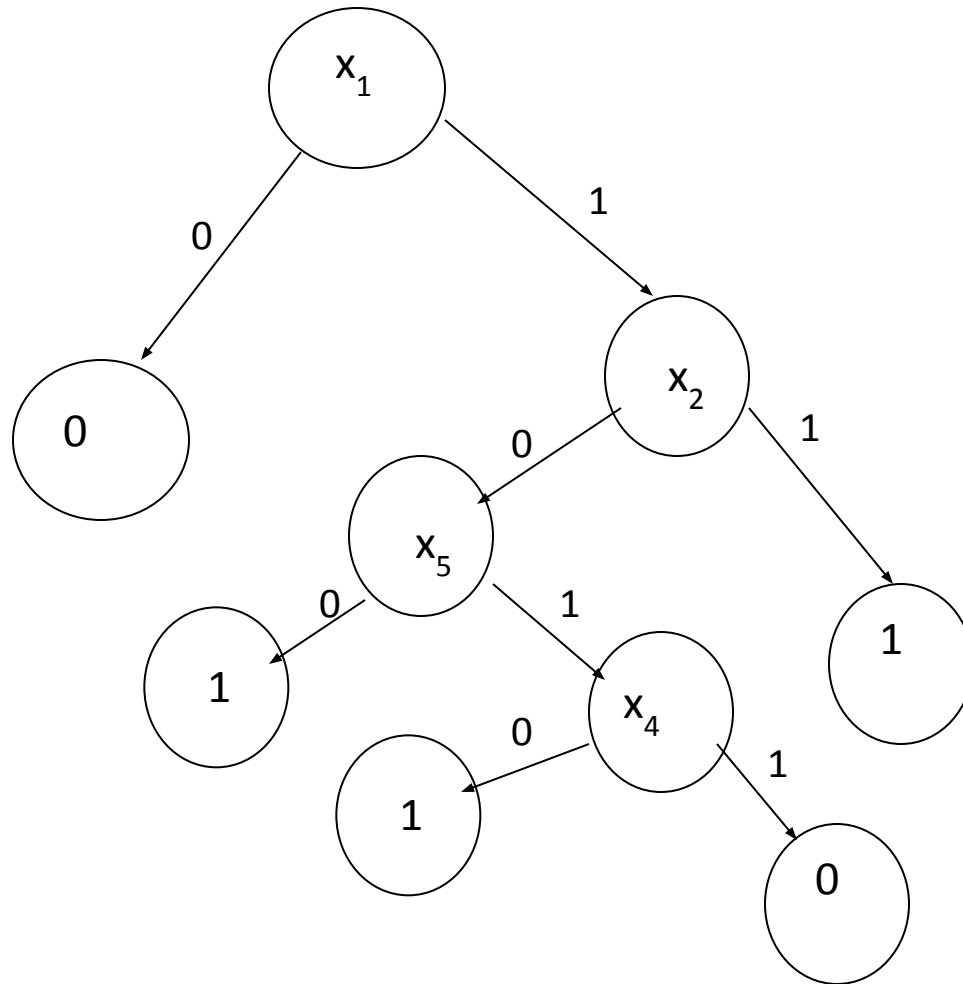
# Run of the algorithm

- Node  $x_5$
- Split the sample
- For  $x_5=0$  DONE !
- For  $x_5=1$  continue.
  - For  $x_3$  (2,1) & (3,1) 0.266
  - For  $x_4$  (3,0) & (2,2) 0
- **Select  $x_4$  DONE !!**

- Example

<u><math>x_5 = 1</math></u>	<u><math>x_5 = 0</math></u>
$x - y$	$x - y$
10001- 1	10010- 1
10101 -1	
10011 - 0	
10111 - 0	
10011 - 0	

# Resulting tree



# DT: Performance

- DT size guarantee
  - Greedy does not have a DT size guarantee
    - Consider  $f(x) = x_1 + x_2 \bmod 2$  with  $d$  attributes
  - Computing the smallest DT is NP-hard
- Boosting Analysis:
  - If we assume a weak learner  $(1/2 + \gamma)$
  - Bound DT size
    - $\exp\{O(1/\gamma^2 1/\epsilon^2 \log^2 1/\epsilon)\}$  Gini/CART
    - $\exp\{O(1/\gamma^2 \log^2 1/\epsilon)\}$  Entropy/C4.5

# Something to think about

- AdaBoost: very good bounds
  - Grows like  $1/\gamma^2$
- DT : exponential bounds in  $1/\gamma^2$
- Comparable results in practice
- How can it be?



# Decision Trees and Forests

- This week:
  - Algorithms for constructing DT
    - Greedy Algorithm
    - Potential Function
      - upper bounds the error
- Next week:
  - Pruning DT
  - Ensemble Methods
    - Random Forest

# Playing with trees

- Regression
- FERNS
- Feature building for sets