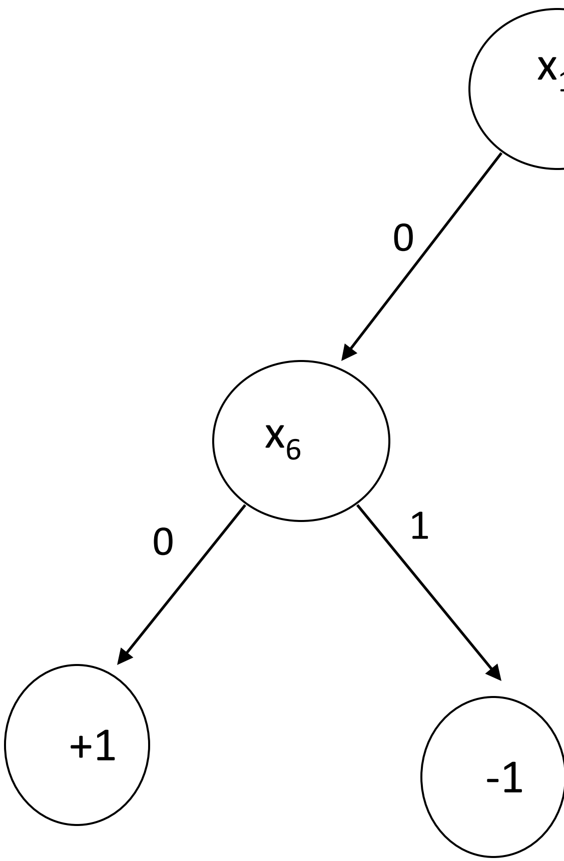
Decision Trees

Decision Trees

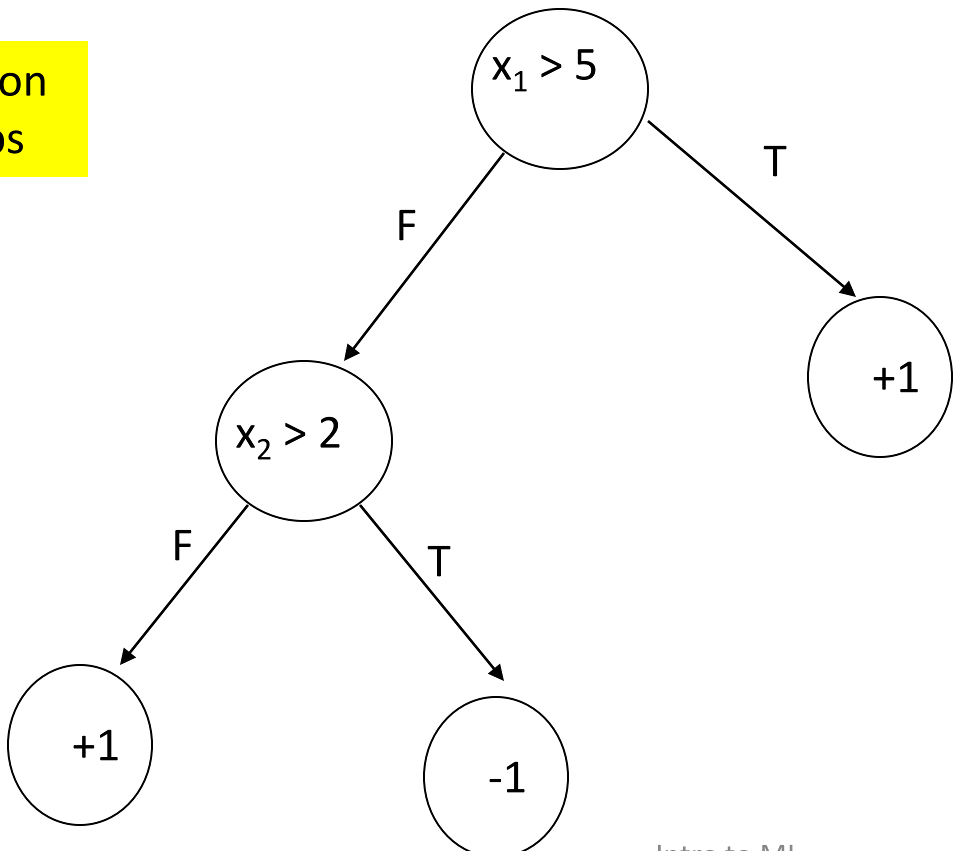
- This week:
  - Algorithms for constructing DT
- Next week:
  - Pruning DT
  - Ensemble methods
    - Random Forest

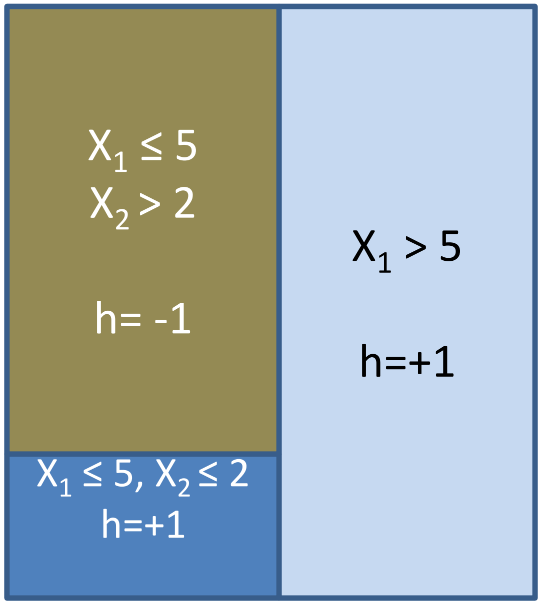
Decision Trees - Boolean



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?

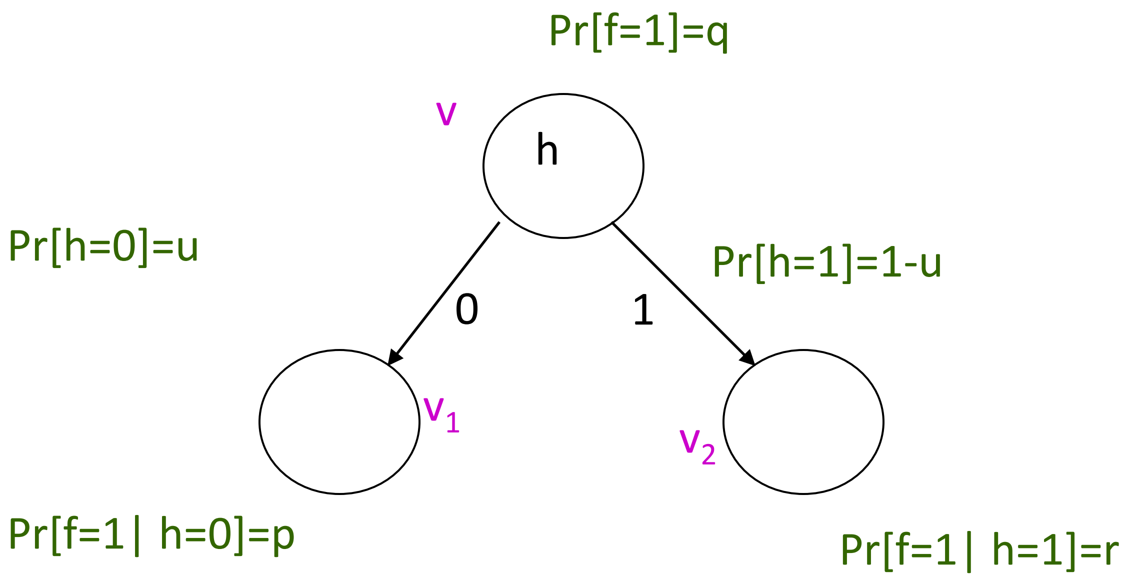
- Human interpretability
- Efficient algorithms:
  - Construction.
  - Classification
- Performance: Reasonable
- Software packages:
  - CART
  - C4.5 and C5

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 = sample size

DT: Selecting a Predicate

- Basic setting:



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

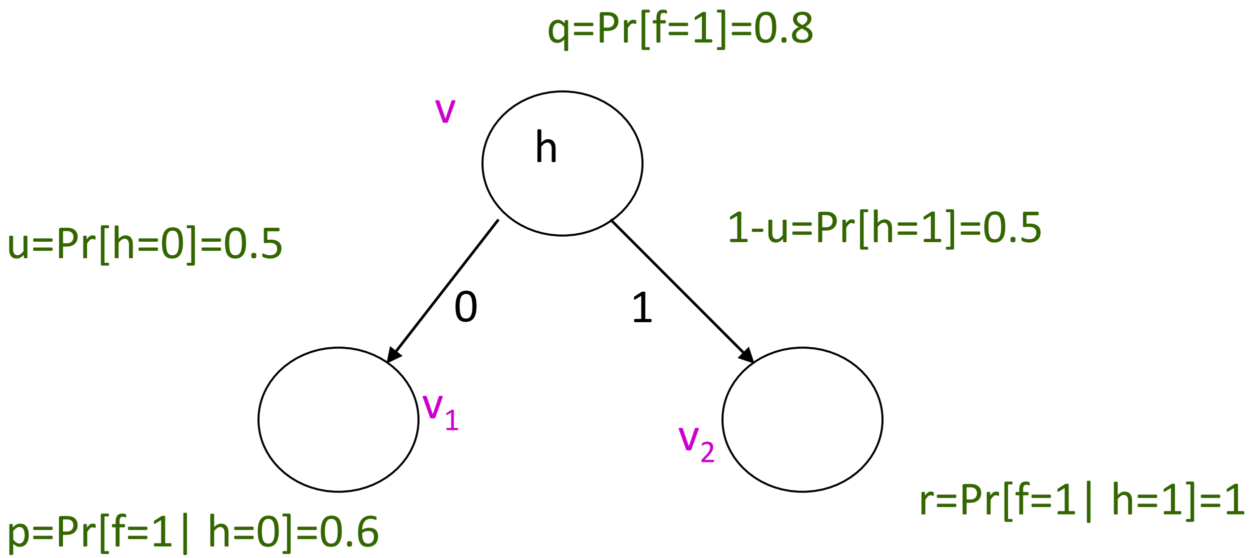
Potential function: setting

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

PF: classification error

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

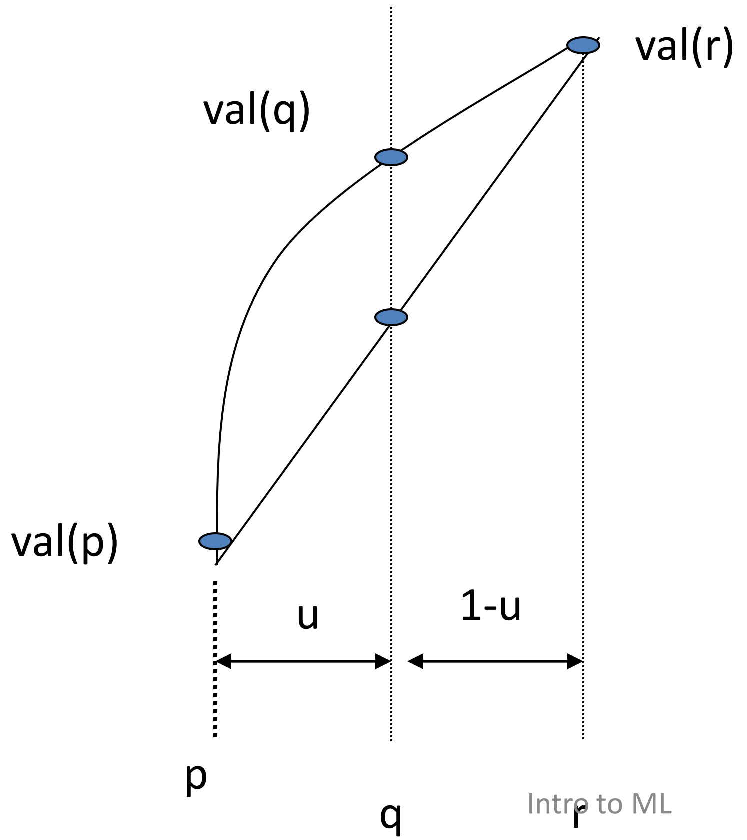
PF: classification error



- 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 in an improvement
- When zero perfect classification.
- Strictly convex.



Potential Functions: Candidates

- Assumption on val:
  - Symmetric: val(q) = val(1-q)
  - Convex
  - val(0)=val(1) = 0 and val(1/2) =1/2
- Outcome:
  - Error(T) ≤ val(T)
  - Minimizing val(T) upper bounds the error!

Potential Functions: Candidates

- Potential Functions:
  - val(q) = Gini(q)=2q(1-q) CART
  - val(q)=Etropy(q)= -q log q –(1-q) log (1-q) C4.5
  - val(q) = sqrt{2 q (1-q) } Variance
- 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 val(h,S)
  - *$val(h,S) = u_h val(p_h) + (1-u_h)\ val(r_h)$*
- Let h' = arg $\min_h$ val(h,S)
- Split S using h' to S^0 and S^1
- Recursively invoke DT(S^0) and DT(S^1)
- Q: What about termination?! What is the running time ?!

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

- 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)
    - Val= 0.5*2*3/5*2/5+ 0.5*2*2/5*3/5=0.48
  - X_4: (6,3) & (4,2)
    - Val=0.6*2*0.5*0.5+0.4*2 *0.5*0.5=0.5
  - X_5: (6,3) & (4,2)
    - 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 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

x_1 = 1	x_1 = 0
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>**x_2 = 1**</u>	<u>**x_2 = 0**</u>
x - y	x - y
11110- 1	10010- 1
11111- 1	10001- 1
	10101 -1
	10011 - 0
	10111 – 0
	10011 - 0

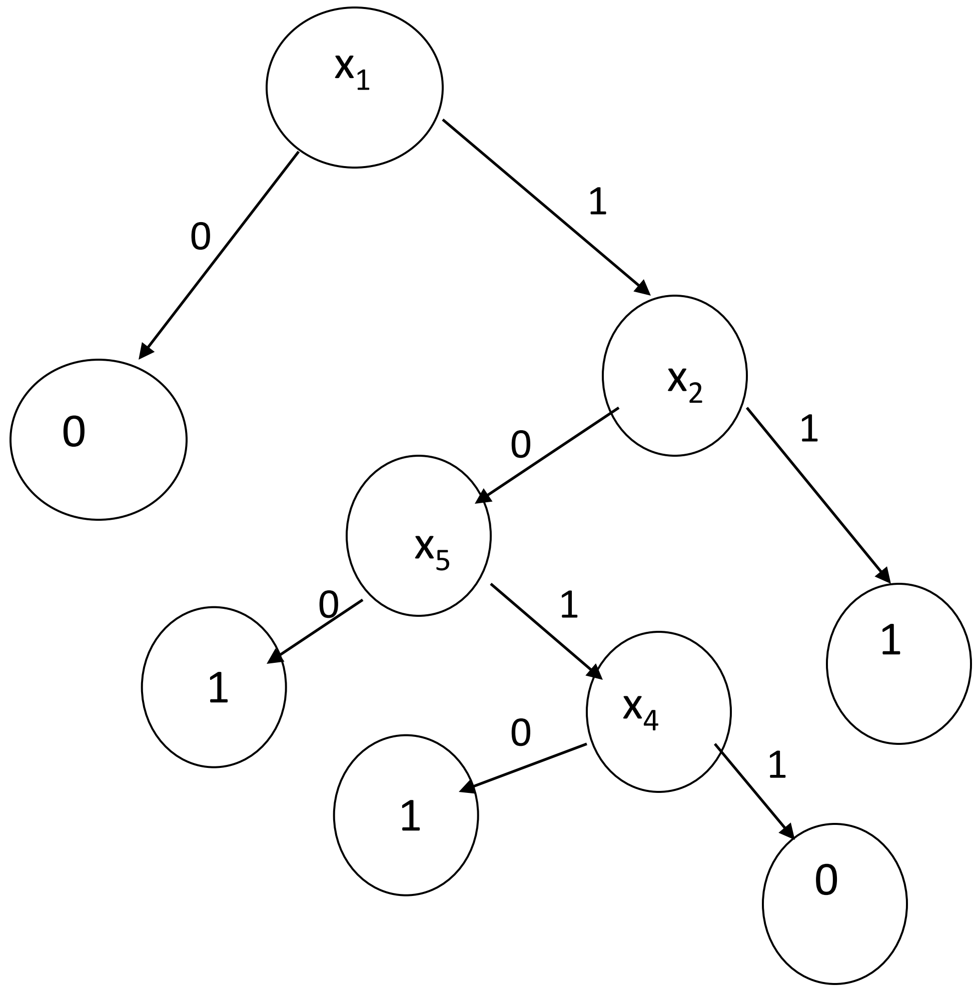
Run of the algorithm

- 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

x_5 = 1	**x_5 = 0**
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 $f(x) = x_1 + x_2 \mod 2$ with d attributes
  - Computing the smallest DT is NP-hard
- Boosting Analysis:
  - If we assume a weak learner ($1/2 + \gamma$)
  - Bound DT size
    - $\exp\{O(1/\gamma^2\ 1/\varepsilon^2 \log^2 1/\varepsilon)\}$ *Gini/CART*
    - $\exp\{O(1/\gamma^2 \log^2 1/\varepsilon)\}$ *Entropy/C4.5*
    - $\exp\{O(1/\gamma^2 \log 1/\varepsilon)\}$ *Variance*

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