Tree-based methods

Hastie, Tibshirani, Friedman Ch 9.2, Ch 10

Kevin Murphy Ch. 16.1-16.4

James, Witten, Hastie, Tibshirani Ch 8

CS 6140

Machine Learning

Professor Olga Vitek

March 23, 2017

Decision trees

See handout (no author)

Overview: recursive partitioning

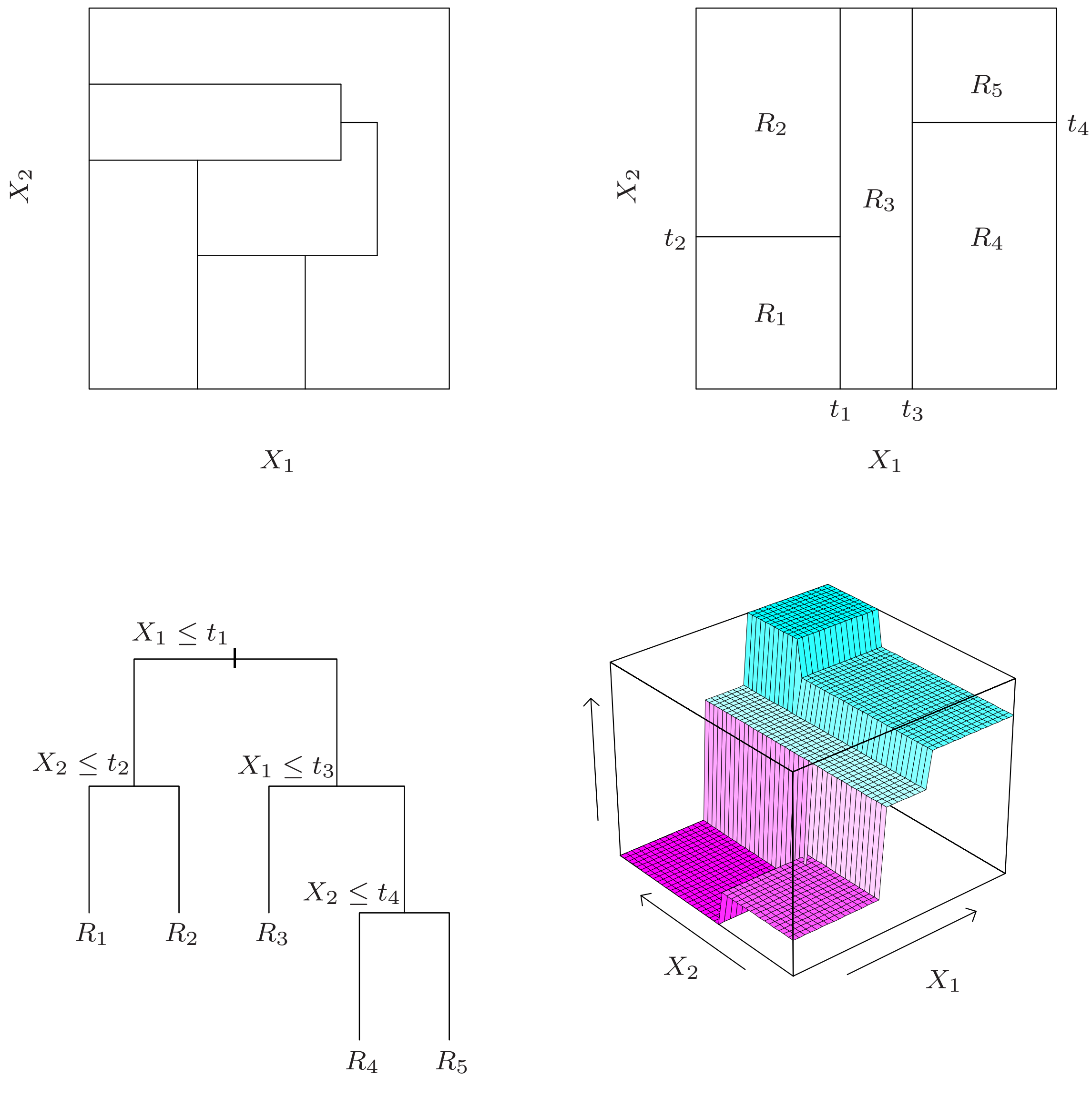


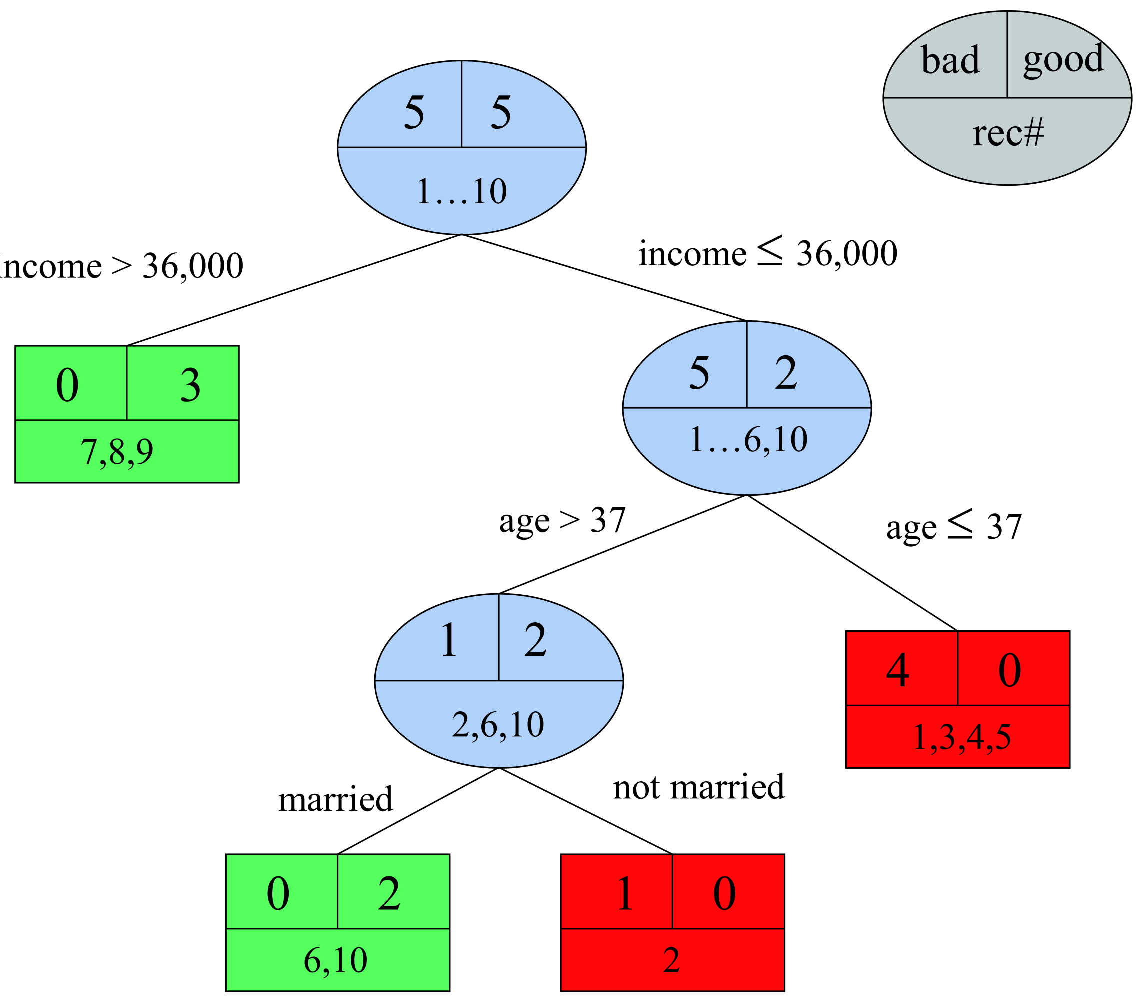
Fig. 9.2. Hastie, Tibshirani, Friedman
The Elements of Statistical Learning 2008

Example:
categorical response
credit score classification

Record	age	married?	own house	income	gender	class
1	22	no	no	28,000	male	bad
2	46	no	yes	32,000	female	bad
3	24	yes	yes	24,000	male	bad
4	25	no	no	27,000	male	bad
5	29	yes	yes	32,000	female	bad
6	45	yes	yes	30,000	female	good
7	63	yes	yes	58,000	male	good
8	36	yes	no	52,000	male	good
9	23	no	yes	40,000	female	good
10	50	yes	yes	28,000	female	good

Anonymous. See handout.

Credit score classification



Anonymous. See handout.

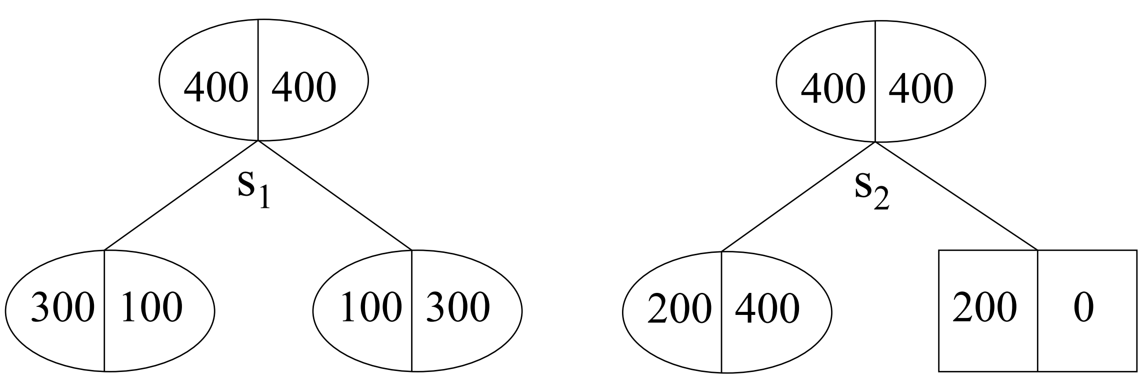
Quality of split

- Choose split to minimize node "impurity"
  - Define as function of relative class frequencies
    $i(t) = \phi(p_1, p_2, \ldots, p_J)$ with J classes
  - $i(t)$ maximized at $\left(\frac{1}{J}, \frac{1}{J}, \ldots, \frac{1}{J}\right)$
  - $i(t)$ minimized at $(0, 0, \ldots, 1)$ (for some class)
  - $i(t)$ is symmetric function of $(p_1, p_2, \ldots, p_J)$

- Quality of split s at node t
  - $\Delta i(s,t) = i(t) - \pi(l)\, i(l) - \pi(r)\, i(r)$, where
    $\pi(l)$ is the proportion of points sent to the left
    $\pi(r)$ is the proportion of points sent to the right

Measures of node impurity

- Resubstitution error: $i(t) = 1 - \max_j p(j|t)$
  - $p(j|t)$: relative frequency of class j in node t
  - $i(t)$: % of misclassified cases
  - Node impurity simplifies to $\Delta i(s,t) =$ $\max_j p(j|l)\,\pi(l) + \max_j p(j|r)\,\pi(r) - \max_j p(j|t)$
  - Problem: ignores where misclassification occurs
  - Below: same impurity; prefer split to the right
  - Ideally, ϕ would be concave (impurity would decrease faster than linearly)



Concave measures of impurity

- Gini index
  - Two classes:

$$i(t) = p(0|t)\,p(1|t) = p(0|t)\;(1 - p(0|t))$$

  - Multiple classes

$$i(t) = \sum_{j=1}^{J} p(j|t)\;(1 - p(j|t))$$

  - Variance of Bernoulli drawing from this class

- Entropy
  - Two classes:

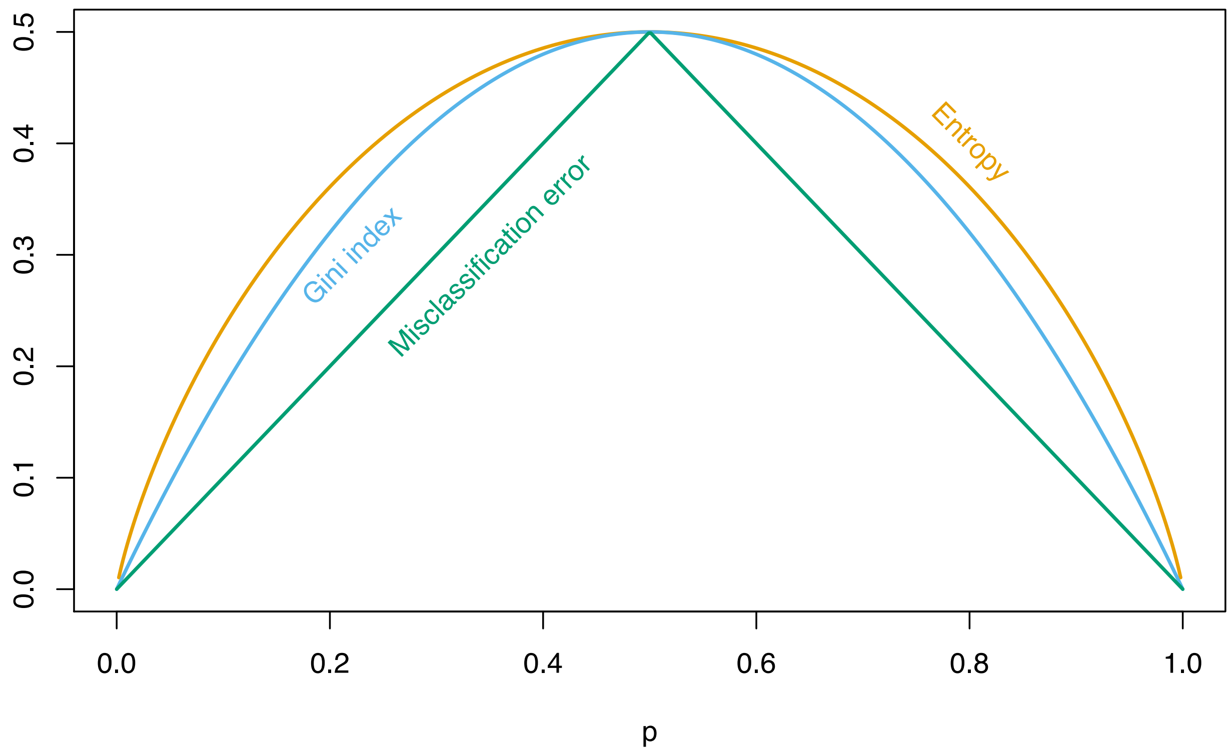
$$i(t) = -p(0|t)\,log\,p(0|t) - p(1|t)\,log\,p(1|t)$$

  - Multiple classes

$$i(t) = -\sum_{j=1}^{J} p(j|t)\,log\,p(j|t)$$

  - Average amount of info gathered by drawing a point from the node. Pure node $\rightarrow$ no info.

Measures of node impurity



Two-class example; Entropy scaled to max=0.5

General properties:

- $\phi(0) = \phi(1) = 0$
- $\phi(p) = \phi\,(p(1-p)$
- $\phi''(p) < 0,\ 0 < p < 1$

Fig. 9.3. Hastie, Tibshirani, Friedman
The Elements of Statistical Learning 2008

Search for splits

Income	Class	Quality (split after) 0.25−
24	B	0.1(1)(0)+0.9(4/9)(5/9) = 0.03
27	B	0.2(1)(0) + 0.8 (3/8)(5/8) = 0.06
28	B,G	0.4(3/4)(1/4) + 0.6(2/6)(4/6) = 0.04
30	G	0.5(3/5)(2/5) + 0.5(2/5)(3/5) = 0.01
32	B,B	0.7(5/7)(2/7) + 0.3(0)(1) = 0.11
40	G	0.8(5/8)(3/8) + 0.2(0)(1) = 0.06
52	G	0.9(5/9)(4/9) + 0.1(0)(1) = 0.03
58	G	

- Split on one predictor at a time
  - X numeric:

    $X \leq constant$ for $constant$ in range of X

    In practice, split wrt distinct values of X
  - X categorical with values in V:

    $X \in S$, S any subset of V
  - Number of possible splits is finite

Anonymous. See handout.

Tree construction

Algorithm: Construct tree

nodelist $\leftarrow$ {training sample}
Repeat
    current node $\leftarrow$ select node from nodelist
    nodelist $\leftarrow$ nodelist $-$ current node
    if impurity(current node) > 0
    then
        S $\leftarrow$ candidate splits in current node
        s* $\leftarrow \arg\max_{s \in S}$ impurity reduction(s,current node)
        child nodes $\leftarrow$ apply(s*,current node)
        nodelist $\leftarrow$ nodelist $\cup$ child nodes
    fi
Until nodelist $= \varnothing$

- Local greedy search; globally suboptimal.
- (Partial) remedy: grow maximal tree, then prune

P	*Q*	*P xor Q*
1	*1*	*0*
1	*0*	*1*
0	*1*	*1*
0	*0*	*0*

Anonymous. See handout.

Cost-complexity pruning

- Notation
  - Tree T, and maximal tree T_{max}
  - $R(T)$ % of misclassified training set cases
  - $|T|$ tree size (i.e. # of terminal nodes)
  - Total cost of the tree $C_\alpha(T) = R(T) + \alpha|T|$
  - α: parameter penalizing tree complexity

- For every α, there exists a smallest subtree $T(\alpha)$ of T_{max}, such that:
  - No subtree of T_{max} has lower cost than $T(\alpha)$:
    $C_\alpha(T(\alpha)) = min_{T \leq T_{max}} C_\alpha(T)$
  - If there is a tie, we pick the smallest tree:
    If $C_\alpha(T) = C_\alpha(T(\alpha))$, then $T(\alpha) \leq T$

- Consequence
  - It is impossible to have two non-nested subtrees with same min cost
  - Although α is continuous, only need a final set that changes tree structure
    $\{\alpha_1, \alpha_2, \ldots\} \to T_1 > T_2 > \ldots \{t_1\}$

Cost-complexity pruning

- Cost of T_t:
  - as an intermediate node: $C_\alpha(T_t) = R(T_t) + \alpha|T_t|$
  - as a terminal node: $C_\alpha(t) = R(t) + \alpha \cdot 1$

- The pruned tree has the same cost-complexity as the original tree when $C_\alpha(t) = C_\alpha(T_t)$

$$\begin{aligned} R(T_t) + \alpha|T_t| &= R(t) + \alpha \cdot 1 \\ \alpha &= \frac{R(t) - R(T_t)}{T - 1} \end{aligned}$$

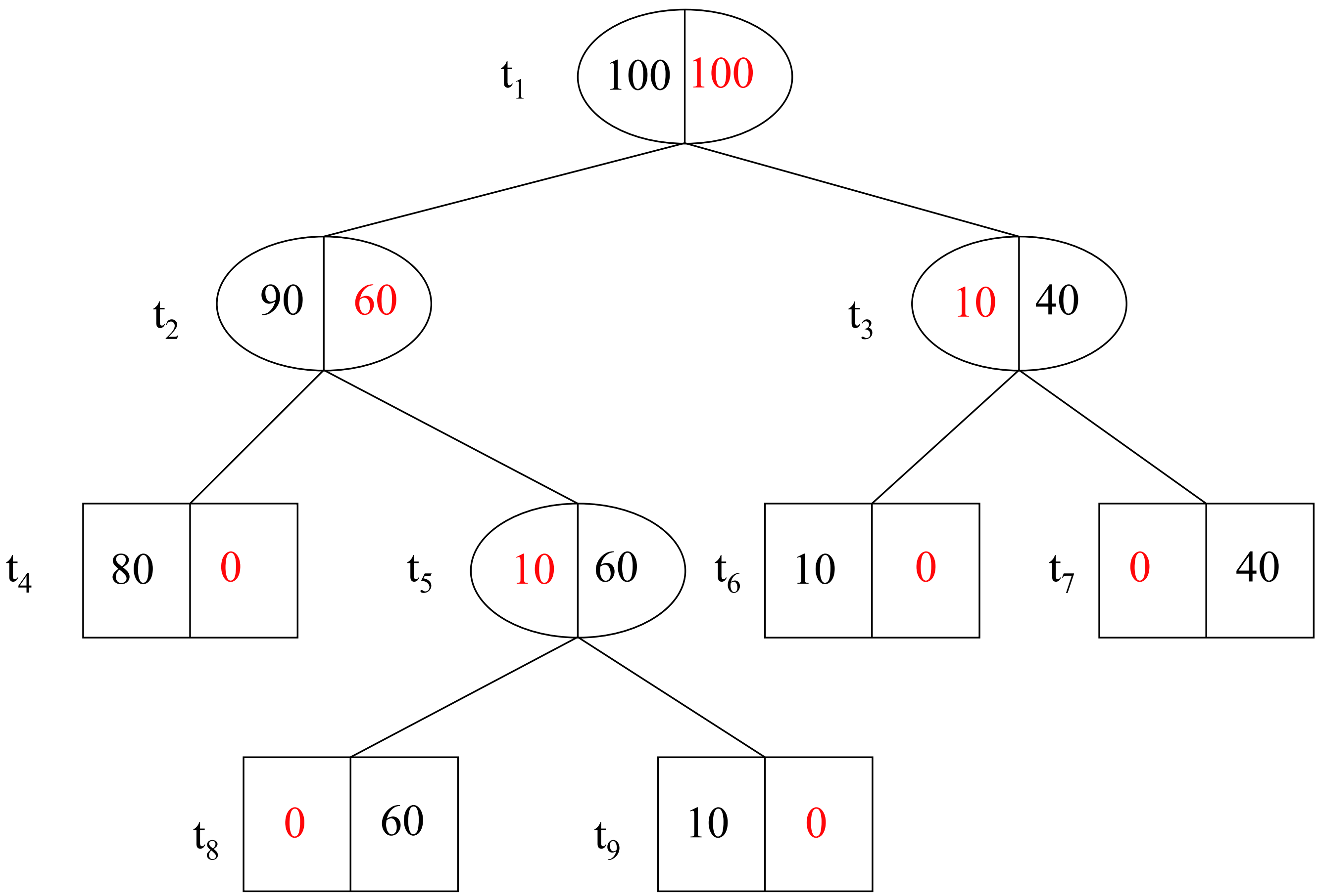
  - For any t, when we increase α beyond this level, pruned tree is better

Cost-complexity pruning

Algorithm: Compute tree sequence

$T_1 \leftarrow T(0)$
$\alpha_1 \leftarrow 0$
$k \leftarrow 1$
While $T_k > \{t_1\}$ do
  For all non-terminal nodes $t \in T_k$
    $g_k(t) \leftarrow \frac{R(t)-R(T_{k,t})}{(|\tilde{T}_{k,t}|-1)}$
  $\alpha_{k+1} \leftarrow \min_t g_k(t)$
  Visit the nodes in top-down order and prune whenever $g_k(t) = \alpha_{k+1}$ to obtain T_{k+1}
  $k \leftarrow k+1$
od

Example


t1
100
100
t2
90
60
t3
10
40
t4
80
0
t5
10
60
t6
10
0
t7
0
40
t8
0
60
t9
10
0