Lecture Slides for

INTRODUCTION TO

Machine Learning

By ETHEM ALPAYDIN
Modified by Zehra Cataltepe
Illisturative example (from: http://l2r.cs.uiuc.edu/~danr/Teaching/CS446-12/)
 is modified by Yusuf Yaslan

alpaydin@boun.edu.tr
http://www.cmpe.boun.edu.tr/~ethem/i2ml

CHAPTER 9:

Decision Trees

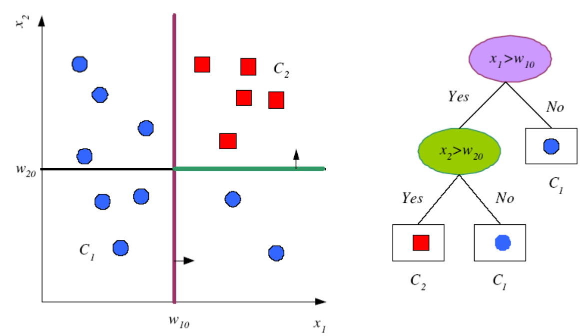
Decision Tree

Parametric Estimation: Assume a model over the whole input space

Nonparametric Estimation: Assume local models, find the local model based on the neighbors. Downside: costly, O(N).

Decision Tree: A hierarchical model for supervised learning where the local region is identified in a sequence of recursive splits.

Tree Uses Nodes, and Leaves



Divide and Conquer

- Internal decision nodes
  - Univariate: Uses a single attribute, x_i
    - Numeric x_i : Binary split : $x_i > w_m$
    - Discrete x_i : n-way split for n possible values
  - Multivariate: Uses all attributes, $\boldsymbol{x}$
- Leaves
  - Classification: Class labels, or proportions
  - Regression: Numeric; r average, or local fit
- Learning is greedy; find the best split recursively (Breiman et al, 1984; Quinlan, 1986, 1993)

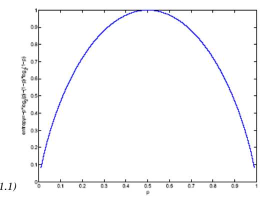
Classification Trees (ID3, CART, C4.5)

- For node m, N_m instances reach m, N^i_m belong to C_i

$$\hat{P}(C_i \mid \mathbf{x}, m) \equiv p^i_m = \frac{N^i_m}{N_m}$$

- Node m is pure if p^i_m is 0 or 1
- One measure of impurity is entropy

$$\mathcal{I}_m = -\sum_{i=1}^{K} p^i_m \log_2 p^i_m$$

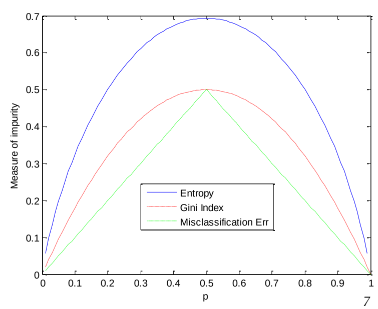


Measures of Impurity of a Split

1. Entropy (defined above)

2. Gini Index:
2p*(1-p)

3. Misclassification err:
1-max(p,1-p)



Best Split

- If node m is pure, generate a leaf and stop, otherwise split and continue recursively
- Impurity after split: N_{mj} of N_m take branch j. N^i_{mj} belong to C_i

$$\hat{P}(C_i \mid \mathbf{x}, m, j) \equiv p^i_{mj} = \frac{N^i_{mj}}{N_{mj}}$$

$$\mathcal{I}'_m = -\sum_{j=1}^{n} \frac{N_{mj}}{N_m} \sum_{i=1}^{K} p^i_{mj} \log_2 p^i_{mj}$$

- Find the variable and split that min impurity (among all variables -- and split positions for numeric variables)

```
GenerateTree(𝒳)
   If NodeEntropy(𝒳)< θ_I /* eq. 9.3
      Create leaf labelled by majority class in 𝒳
      Return
   i ← SplitAttribute(𝒳)
   For each branch of x_i
      Find 𝒳_i falling in branch
      GenerateTree(𝒳_i)
SplitAttribute(𝒳)
   MinEnt← MAX
   For all attributes i = 1,...,d
      If x_i is discrete with n values
         Split 𝒳 into 𝒳_1,...,𝒳_n by x_i
         e ← SplitEntropy(𝒳_1,...,𝒳_n) /* eq. 9.8 */
         If e<MinEnt MinEnt ← e; bestf ← i
      Else /* x_i is numeric */
         For all possible splits
            Split 𝒳 into 𝒳_1,𝒳_2 on x_i
            e←SplitEntropy(𝒳_1,𝒳_2)
            If e<MinEnt MinEnt ← e; bestf ← i
   Return bestf
```

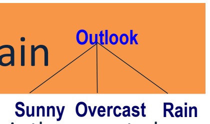

Classification Tree Notes:

Branching factor and tree height both define the complexity of a tree.

When there is noise, stop when impurity is less than a certain threshold.

At the leaves, store not only the labels, but also the ratio for other classes. This helps with calculating risks for examples.

Information Gain



- The information gain of an attribute a is the expected reduction in entropy caused by partitioning on this attribute

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

where S_v is the subset of S for which attribute a has value v and the entropy of partitioning the data is calculated by weighing the entropy of each partition by its size relative to the original set

- Partitions of low entropy (imbalanced splits) lead to high gain
- Go back to check which of the A, B splits is better

An Illustrative Example

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

An Illustrative Example (II)

$Entropy(S) = -\frac{9}{14}\log(\frac{9}{14}) - \frac{5}{14}\log(\frac{5}{14}) = 0.94$

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

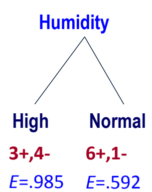
9+,5-

An Illustrative Example (II)

Humidity	Wind	Play Tennis
High	Weak	No
High	Strong	No
High	Weak	Yes
High	Weak	Yes
Normal	Weak	Yes
Normal	Strong	No
Normal	Strong	Yes
High	Weak	No
Normal	Weak	Yes
Normal	Weak	Yes
Normal	Strong	Yes
High	Strong	Yes
Normal	Weak	Yes
High	Strong	No

9+,5-
E=.94

An Illustrative Example (II)

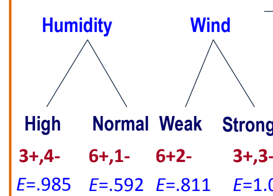


Humidity	Wind	Play Tennis
High	Weak	No
High	Strong	No
High	Weak	Yes
High	Weak	Yes
Normal	Weak	Yes
Normal	Strong	No
Normal	Strong	Yes
High	Weak	No
Normal	Weak	Yes
Normal	Weak	Yes
Normal	Strong	Yes
High	Strong	Yes
Normal	Weak	Yes
High	Strong	No

9+,5-
E=.94

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

An Illustrative Example (II)



Humidity	Wind	Play Tennis
High	Weak	No
High	Strong	No
High	Weak	Yes
High	Weak	Yes
Normal	Weak	Yes
Normal	Strong	No
Normal	Strong	Yes
High	Weak	No
Normal	Weak	Yes
Normal	Weak	Yes
Normal	Strong	Yes
High	Strong	Yes
Normal	Weak	Yes
High	Strong	No

9+,5-
E=.94

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

An Illustrative Example (II)

Humidity: High 3+,4- E=.985 | Normal 6+,1- E=.592

Wind: Weak 6+2- E=.811 | Strong 3+,3- E=1.0

Gain(S,Humidity)=
.94 - 7/14 0.985
- 7/14 0.592=
0.151

Humidity	Wind	Play Tennis
High	Weak	No
High	Strong	No
High	Weak	Yes
High	Weak	Yes
Normal	Weak	Yes
Normal	Strong	No
Normal	Strong	Yes
High	Weak	No
Normal	Weak	Yes
Normal	Weak	Yes
Normal	Strong	Yes
High	Strong	Yes
Normal	Weak	Yes
High	Strong	No

9+,5-
E=.94

$$\mathrm{Gain}(S,a) = \mathrm{Entropy}(S) - \sum_{v \in values(a)} \frac{|S_v|}{|S|}\mathrm{Entropy}(S_v)$$

An Illustrative Example (II)

Humidity: High 3+,4- E=.985 | Normal 6+,1- E=.592

Wind: Weak 6+2- E=.811 | Strong 3+,3- E=1.0

Gain(S,Humidity)=
.94 - 7/14 0.985
- 7/14 0.592=
0.151

Gain(S,Wind)=
.94 - 8/14 0.811
- 6/14 1.0 =
0.048

Humidity	Wind	Play Tennis
High	Weak	No
High	Strong	No
High	Weak	Yes
High	Weak	Yes
Normal	Weak	Yes
Normal	Strong	No
Normal	Strong	Yes
High	Weak	No
Normal	Weak	Yes
Normal	Weak	Yes
Normal	Strong	Yes
High	Strong	Yes
Normal	Weak	Yes
High	Strong	No

9+,5-
E=.94

$$\mathrm{Gain}(S,a) = \mathrm{Entropy}(S) - \sum_{v \in values(a)} \frac{|S_v|}{|S|}\mathrm{Entropy}(S_v)$$

An Illustrative Example (III)

Outlook

Gain(S,Humidity)=0.151
Gain(S,Wind) = 0.048
Gain(S,Temperature) = 0.029
Gain(S,Outlook) = 0.246

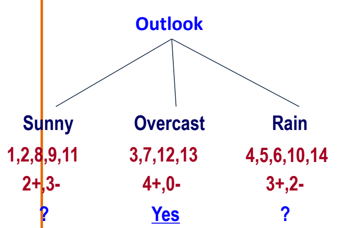
An Illustrative Example (III)

Outlook

Sunny	Overcast	Rain
1,2,8,9,11	3,7,12,13	4,5,6,10,14
2+,3-	4+,0-	3+,2-
?	Yes	?

Day	Outlook	PlayTennis
1	Sunny	No
2	Sunny	No
3	Overcast	Yes
4	Rain	Yes
5	Rain	Yes
6	Rain	No
7	Overcast	Yes
8	Sunny	No
9	Sunny	Yes
10	Rain	Yes
11	Sunny	Yes
12	Overcast	Yes
13	Overcast	Yes
14	Rain	No

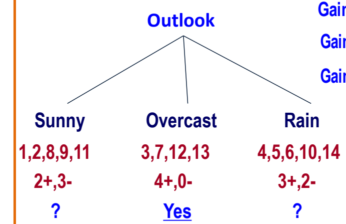
An Illustrative Example (III)



Day	Outlook	PlayTennis
1	Sunny	No
2	Sunny	No
3	Overcast	Yes
4	Rain	Yes
5	Rain	Yes
6	Rain	No
7	Overcast	Yes
8	Sunny	No
9	Sunny	Yes
10	Rain	Yes
11	Sunny	Yes
12	Overcast	Yes
13	Overcast	Yes
14	Rain	No

Continue until:
- Every attribute is included in path, or,
- All examples in the leaf have same label

An Illustrative Example (IV)



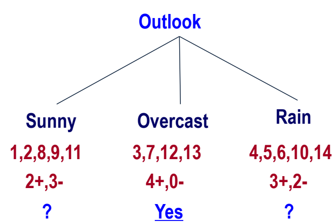
$Gain(S_{sunny}, Humidity) = .97-(3/5)\ 0-(2/5)\ 0 = .97$

$Gain(S_{sunny}, Temp) = .97-\ 0-(2/5)\ 1 = .57$

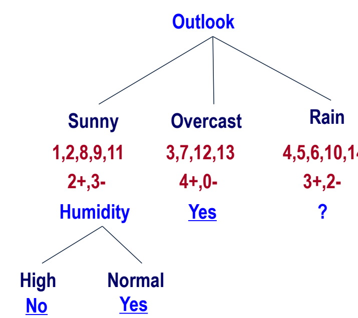
$Gain(S_{sunny}, Wind) = .97-(2/5)\ 1 - (3/5)\ .92 = .02$

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

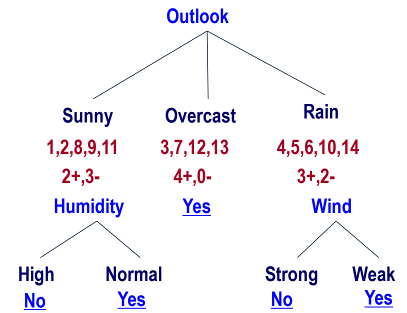
An Illustrative Example (V)



An Illustrative Example (V)



An Illustrative Example (VI)



Summary: ID3 (Examples, Attributes, Label)

- Let S be the set of Examples
  Label is the target attribute (the prediction)
  Attributes is the set of measured attributes
- Create a Root node for tree
- If all examples are labeled the same return a single node tree with Label
- Otherwise Begin
- A = attribute in Attributes that *best* classifies S
- for each possible value v of A
- Add a new tree branch corresponding to A=v
- Let *Sv* be the subset of examples in S with A=v
- if *Sv* is empty: add leaf node with the common value of Label in S
- Else: below this branch add the subtree
- ID3(*Sv*, Attributes - {a}, Label)

End

Return Root

Regression Trees

- Error at node *m*:

$$b_m(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} \in \mathcal{X}_m : \mathbf{x} \text{ reaches node } m \\ 0 & \text{otherwise} \end{cases}$$

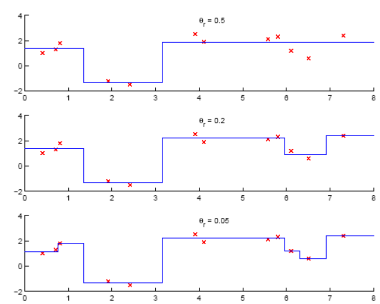
$$E_m = \frac{1}{N_m}\sum_t \left(r^t - g_m\right)^2 b_m\left(\mathbf{x}^t\right) \qquad g_m = \frac{\sum_t b_m\left(\mathbf{x}^t\right) r^t}{\sum_t b_m\left(\mathbf{x}^t\right)}$$

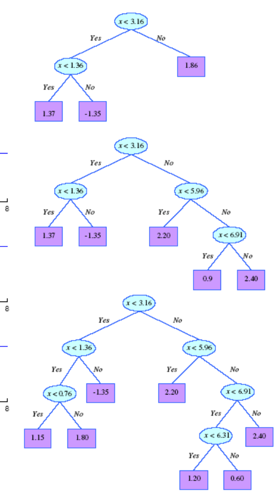
- After splitting:

$$b_{mj}(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} \in \mathcal{X}_{mj} : \mathbf{x} \text{ reaches node } m \text{ and branch } j \\ 0 & \text{otherwise} \end{cases}$$

$$E'_m = \frac{1}{N_m}\sum_j \sum_t \left(r^t - g_{mj}\right)^2 b_{mj}\left(\mathbf{x}^t\right) \qquad g_{mj} = \frac{\sum_t b_{mj}\left(\mathbf{x}^t\right) r^t}{\sum_t b_{mj}\left(\mathbf{x}^t\right)}$$

Model Selection in Trees:





if $E_m < \theta_r$ make the node a leaf node

Regression Trees

Another error function is maximum possible error.

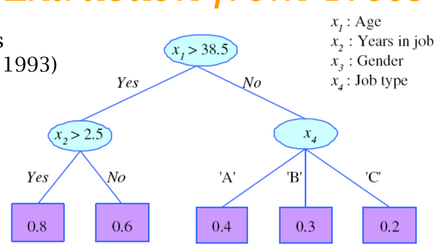
Similar to running mean and running line for non parametric estimation, instead of mean of training data at a leaf, a linear interpolation to them could be used. Could minimize error faster and hence create smaller trees, at the expense of storing the linear interpolation weights and the linear interpolation computation.

Pruning Trees

- Remove subtrees for better generalization (decrease variance)
  - Prepruning: Early stopping
  - Postpruning: Grow the whole tree then prune subtrees which overfit on the pruning set
- Prepruning is faster, postpruning is more accurate (requires a separate pruning set, which is separate from validation set.)

Rule Extraction from Trees

C4.5Rules
(Quinlan, 1993)



R1: IF (age>38.5) AND (years-in-job>2.5) THEN y =0.8
R2: IF (age>38.5) AND (years-in-job≤2.5) THEN y =0.6
R3: IF (age≤38.5) AND (job-type='A') THEN y =0.4
R4: IF (age≤38.5) AND (job-type='B') THEN y =0.3
R5: IF (age≤38.5) AND (job-type='C') THEN y =0.2

Learning Rules

- Rule induction is similar to tree induction but
  - tree induction is breadth-first,
  - rule induction is depth-first; one rule at a time
- Rule set contains rules; rules are conjunctions of terms
- Rule covers an example if all terms of the rule evaluate to true for the example
- Sequential covering: Generate rules one at a time until all positive examples are covered
- Outer loop to add one rule at a time, inner rule to add one condition at a time.
- IREP (Fürnkrantz and Widmer, 1994), Ripper (Cohen, 1995)

```
Ripper(Pos,Neg,k)
  RuleSet ← LearnRuleSet(Pos,Neg)
  For k times
    RuleSet ← OptimizeRuleSet(RuleSet,Pos,Neg)
LearnRuleSet(Pos,Neg)
  RuleSet ← ∅
  DL ← DescLen(RuleSet,Pos,Neg)
  Repeat
    Rule ← LearnRule(Pos,Neg)
    Add Rule to RuleSet
    DL' ← DescLen(RuleSet,Pos,Neg)
    If DL'>DL+64
      PruneRuleSet(RuleSet,Pos,Neg)
      Return RuleSet
    If DL'<DL DL ← DL'
      Delete instances covered from Pos and Neg
  Until Pos = ∅
  Return RuleSet
```


Ripper (Cohen 95, Earlier algo: Irep)

Conditions added to the rule to maximize an information gain measure used in Quinlan's Foil algo (1990).

R: current rule, R' candidate rule after adding cond:

$$Gain(R,R') = s\left(\log_2 \frac{N'_+}{N'} - \log_2 \frac{N_+}{N}\right)$$

N:no of instances covered by R, N+:no of true positives

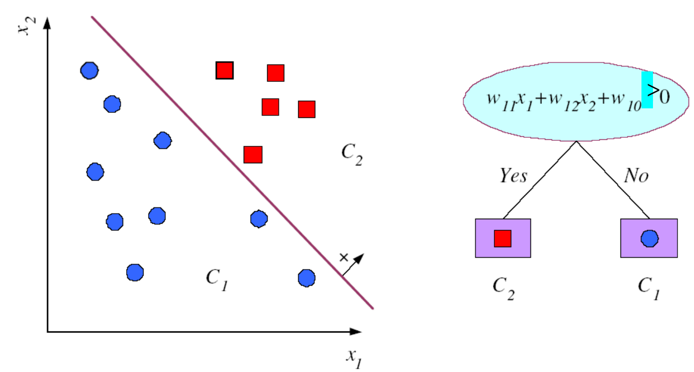
s:no of true positives in R, which are still true positives in R', after adding the condition.

Conditions added to a rule until it covers no negative examples.

Once a rule is grown, it is pruned back by deleting conditions in reverse order, maximizing rule value metric: rvm(R)=(p-n)/(p+n)


```
PruneRuleSet(RuleSet,Pos,Neg)
  For each Rule ∈ RuleSet in reverse order
    DL ← DescLen(RuleSet,Pos,Neg)
    DL' ← DescLen(RuleSet-Rule,Pos,Neg)
    IF DL'<DL Delete Rule from RuleSet
  Return RuleSet
OptimizeRuleSet(RuleSet,Pos,Neg)
  For each Rule ∈ RuleSet
      DL0 ← DescLen(RuleSet,Pos,Neg)
      DL1 ← DescLen(RuleSet-Rule+
        ReplaceRule(RuleSet,Pos,Neg),Pos,Neg)
      DL2 ← DescLen(RuleSet-Rule+
        ReviseRule(RuleSet,Rule,Pos,Neg),Pos,Neg)
      If DL1=min(DL0,DL1,DL2)
        Delete Rule from RuleSet and
          add ReplaceRule(RuleSet,Pos,Neg)
      Else If DL2=min(DL0,DL1,DL2)
        Delete Rule from RuleSet and
          add ReviseRule(RuleSet,Rule,Pos,Neg)
  Return RuleSet
```


Multivariate Trees



Software

Matlab: classregtree()
Weka: Various decision tree algorithms in java (J48)
See 5.0 (Ross Quinlan' s rule extraction program)