LectureTrees
Learning Trees

Given this input:

```
@relation weather

@attribute outlook {sunny, overcast, rainy}
@attribute temperature real
@attribute humidity real
@attribute windy {TRUE, FALSE}
@attribute play {yes, no}

@data
sunny,85,85,FALSE,no
sunny,80,90,TRUE,no
overcast,83,86,FALSE,yes
rainy,70,96,FALSE,yes
rainy,68,80,FALSE,yes
rainy,65,70,TRUE,no
overcast,64,65,TRUE,yes
sunny,72,95,FALSE,no
sunny,69,70,FALSE,yes
rainy,75,80,FALSE,yes
sunny,75,70,TRUE,yes
overcast,72,90,TRUE,yes
overcast,81,75,FALSE,yes
rainy,71,91,TRUE,no
```

WEKA learns this decision tree:

```
outlook = sunny
|   humidity <= 75: yes
|   humidity > 75: no
outlook = overcast: yes
outlook = rainy
|   windy = TRUE: no
|   windy = FALSE: yes
```

Q: how?

A: "iterative dichotomization". This is a general algorithm defined by the operators *measure, split, recurse, stop, condense, prune*.

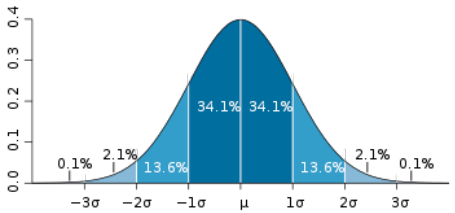
- Split things up
- Recurse on each split

How to learn a tree

- Given a bag of mixed-up stuff.
  - Need a measure of "mixed-up"
- Split: Find something that divides up the bag in two new sub-bags
  - And each sub-bag is less mixed-up;
  - Each split is the root of a sub-tree.
- Recurse: repeat for each sub-bag
  - i.e. on just the data that falls into each part of the split
    - Need a Stop rule
    - Condense the instances that fall into each sub-bag
- Prune back the generated tree.

Different tree learners result from different selections of

- CART: (regression trees)
  - measure: standard deviation
    - Three "normal" curves with different standard deviations
    - Expected values under the normal curve

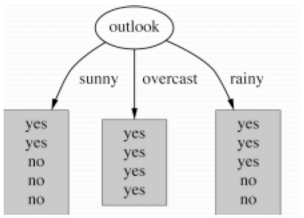


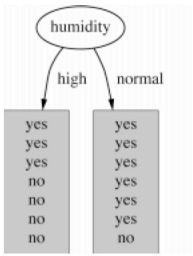
  - condense: report the average of the instances in each bag.
- M5prime: (model trees)
  - measure: standard deviation
  - condense: generate a linear model of the form a+b * x1 +c * x2 + d * x3 +...
- J48: (decision trees)
  - measure: "entropy"

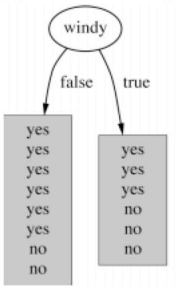
$$H(X) = \mathrm{E}(I(X)) = \sum_{i=1}^{n} p(x_i)\log_2\left(1/p(x_i)\right) = -\sum_{i=1}^{n} p(x_i)\log_2 p(x_i)$$

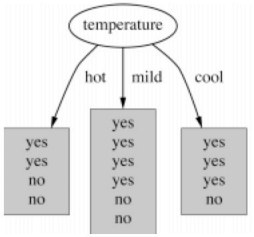
  - condense: report majority class

Example: C4.5 (a.k.a. J48)









Q: which attribute is the best to split on?

A: the one which will result in the smallest tree:

Heuristic: choose the attribute that produces the "purest" nodes (purity = not-mixed-up)

e.g. Outlook= sunny

- info((2,3))= entropy(2/5,3/5) = -2/5 * log(2/5) - 3/5 * log(3/5) = 0.971 bits

Outlook = overcast

- info((4,0)) = entropy(1,0) = -1 * log(1) - 0 * log(0) = 0 bits

Outlook = rainy

- info((3,2)) = entropy(3/5, 2/5) = -3/5 * log(3/5) - 2/5 * log(2/5) = 0.971 bits

Expected info for Outlook = Weighted sum of the above

- info((3,2),(4,0),(3,2)) = 5/14 * 0.971 + 4/14 * 0 + 5/14 * 0.971 = 0.693

Computing the information gain

- e.g. information before splitting minus information after splitting
- e.g. gain for attributes from weather data:
- gain("Outlook") = info(9,5) - info(2,3,4,0,3,2) = 0.940 - 0.963 = 0.247 bits
- gain("Temperature") = 0.247 bits
- gain("Humidity") = 0.152 bits
- gain("Windy") = 0.048 bits

Problem: Numeric Variables

No problem:

- use cliff learning to split the numerics
- Standard method proposed by Fayyad.

Other tree learning

Regression trees

- Differences to decision trees:
- Splitting criterion: minimizing intra-subset variation
- Pruning criterion: based on numeric error measure
- Leaf node predicts average class values of training instances reaching that node
- Can approximate piecewise constant functions
- Easy to interpret:

```
curb-weight <= 2660 :
|   curb-weight <= 2290 :
|   |   curb-weight <= 2090 :
|   |   |   length <= 161 : price=6220
|   |   |   length >  161 : price=7150
|   |   curb-weight >  2090 : price=8010
|   curb-weight >  2290 :
|   |   length <= 176 : price=9680
|   |   length >  176 :
|   |   |   normalized-losses <= 157 : price=10200
|   |   |   normalized-losses >  157 : price=15800
curb-weight >  2660 :
|   width <= 68.9 : price=16100
|   width >  68.9 : price=25500
```

- More sophisticated version: model trees

Model trees

- Regression trees with linear regression functions at each node

```
curb-weight <= 2660 :
|   curb-weight <= 2290 : LM1
|   curb-weight >  2290 :
|   |   length <= 176 : LM2
|   |   length >  176 : LM3
curb-weight >  2660 :
|   width <= 68.9 : LM4
|   width >  68.9 : LM5
.
LM1:  price = -5280 + 6.68 * normalized-losses
                    + 4.44 * curb-weight
                    + 22.1 * horsepower - 85.8 * city-mpg
                    + 98.6 * highway-mpg
LM2:  price = 9680
LM3:  price = -1100 + 91 * normalized-losses
LM4:  price = 9940 + 47.5 * horsepower
LM5:  price = -19000 + 13.2 * curb-weight
```

- Linear regression applied to instances that reach a node after full regression tree has been built
- Only a subset of the attributes is used for LR
- Attributes occurring in subtree (+maybe attributes occurring in path to the root)
- Fast: overhead for LR not large because usually only a small subset of attributes is used in tree

Building the tree

- Splitting criterion: standard deviation reduction into i bins
- SDR = sd(T) - sum((|Ti| / |T| * sd(Ti)))
  - where (|T| = number of instances in that tree).
- Termination criteria (important when building trees for numeric prediction):
- Standard deviation becomes smaller than certain fraction of sd for full training set (e.g. 5%)
- Too few instances remain (e.g. less than four)

Smoothing (Model Trees)

- Naive method for prediction outputs value of LR for corresponding leaf node
- Performance can be improved by smoothing predictions using internal LR models
- Predicted value is weighted average of LR models along path from root to leaf
- Smoothing formula: p' = (np+kq)/(n+k)
- p' is what gets passed up the tree
- p is what got passed from down the tree
- q is the value predicted by the linear models at this node
- n is the number of examples that fall down to here
- k magic smoothing constant; default=2