Classification with Decision Trees and Rules
Density Estimation – looking ahead

- Compare it against the two other major kinds of models:

  - **Classifier**
    - Prediction of **categorical** output or class
    - One of a few discrete values
  
  - **Density Estimator**
    - Probability
  
  - **Regressor**
    - Prediction of **real-valued** output
DECISION TREE LEARNING: OVERVIEW
Decision tree learning

Induction of decision trees
JR Quinlan - Machine learning, 1986 - Springer

... Several studies have been carried out to see how this modified procedure holds up under varying levels of noise (Quinlan 1983b, 1985a ... each experiment, the whole set of objects was artificially corrupted as described below and used as a training set to produce a decision tree. ...

Cited by 10584 - Related articles - Library Search - All 75 versions

Cited by 13602
A decision tree

Dependent variable: PLAY

OUTLOOK?

sunny

- Play: 2
- Don't Play: 3

HUMIDITY?

- <= 70
  - Play: 2
  - Don't Play: 0
- > 70
  - Play: 0
  - Don't Play: 3

overcast

- Play: 4
- Don't Play: 0

rain

WINDY?

- TRUE
  - Play: 0
  - Don't Play: 2
- FALSE
  - Play: 3
  - Don't Play: 0
Another format: a set of rules

if O=sunny and H<= 70 then PLAY
else if O=sunny and H>70 then DON’T_PLAY
else if O=overcast then PLAY
else if O=rain and windy then DON’T_PLAY
else if O=rain and !windy then PLAY

One rule per leaf in the tree

Simpler rule set

if O=sunny and H> 70 then DON’T_PLAY
else if O=rain and windy then DON’T_PLAY
else PLAY
A regression tree

Dependent variable: PLAY

- Play =~ 33
  - Outlook?:
    - Sunny:
      - Humidity?:
        - <= 70:
          - Play =~ 37
            - Play = 30m, 45min
        - > 70:
          - Play =~ 5
            - Play = 0m, 0m, 15m
    - Overcast:
      - Play =~ 48
        - Windy?:
          - False:
            - Play =~ 18
              - Play = 20m, 30m, 45m,
Motivations for trees and rules

- Often you can find a fairly accurate classifier which is small and easy to understand.
  - Sometimes this gives you useful insight into a problem, or helps you debug a feature set.
- Sometimes features interact in complicated ways
  - Trees can find interactions (e.g., “sunny and humid”). Again, sometimes this gives you some insight into the problem.
- Trees are very inexpensive at test time
  - You don’t always even need to compute all the features of an example.
  - You can even build classifiers that take this into account....
  - Sometimes that’s important (e.g., “bloodPressure<100” vs “MRIScan=normal” might have different costs to compute).
An example: “Is it the Onion”? 

Dataset: 200 Onion articles, ~500 Economist articles.

Accuracies: almost 100% with Naïve Bayes!

I used a rule-learning method called RIPPER
Fast Effective Rule Induction

Authors: William W Cohen
Publication date: 1995
Journal: Proceedings of the Twelfth International Conference on Machine Learning, Lake Tahoe, California
Total citations: Cited by 3040

Scholar articles:
- Fast Effective Rule Induction
  WW Cohen - Proceedings of the Twelfth International Conference on ..., 1995
  Cited by 3040 - Related articles - All 44 versions
Translation:

if “enlarge” is in the set-valued attribute wordsArticle
then class = fromOnion.  *this rule is correct 173 times, and never wrong*
...

if “added” is in the set-valued attribute wordsArticle
and “play” is in the set-valued attribute wordsArticle
then class = fromOnion.  *this rule is correct 6 times, and wrong once*
...
conference. "Many of them had been given nothing more than a pair of tube socks or men's briefs to wear."

Law enforcement officials continued clearing models from the compound reportedly grown too complacent to conduct its suicide mission, an attack on the San Onofre Nuclear Generating Station.

Three of the six terrorists spend an afternoon together watching an today."

Lieberman tells Hartford voters he'll be brief.
After cleaning ‘Enlarge Image’ lines

bash-3.2$ ripper clean-onion
Final hypothesis is:
fromOnion :- wordsInArticle ~ i, wordsInArticle ~ added, wordsInArticle ~ my (82/0).
fromOnion :- wordsInArticle ~ i, wordsInArticle ~ monday (42/0).
fromOnion :- wordsInArticle ~ said, wordsInArticle ~ tuesday (22/0).
fromOnion :- wordsInArticle ~ added, wordsInArticle ~ re (13/4).
fromOnion :- wordsInArticle ~ said, wordsInArticle ~ my, wordsInArticle ~ really (8/2).
fromOnion :- wordsInArticle ~ monday (5/1).
fromOnion :- wordsInArticle ~ ll, wordsInArticle ~ me (5/0).
fromOnion :- wordsInArticle ~ u, wordsInArticle ~ political (4/0).
fromOnion :- wordsInArticle ~ added, wordsInArticle ~ truly (3/0).
default fromEconomist (526/8).

========================================================================
Train error rate:  2.07% +/- 0.53% (725 datapoints)  <<
Hypothesis size:  9 rules, 28 conditions
Learning time:  3.98 sec

Also, estimated test error rate increases from 1.4% to 6%
Different Subcategories of Economist Articles

Final hypothesis is:
aboutInternational :- wordsInArticle ~ countries, wordsInArticle ~ nations (9/4).
aboutInternational :- wordsInArticle ~ soil (7/1).
aboutInternational :- wordsInArticle ~ based, wordsInArticle ~ authorities (6/4).
aboutNorthAmerica :- wordsInArticle ~ republican, wordsInArticle ~ barack (19/0).
aboutNorthAmerica :- wordsInArticle ~ barack (9/3).
aboutNorthAmerica :- wordsInArticle ~ republican, wordsInArticle ~ americans (13/1).
aboutNorthAmerica :- wordsInArticle ~ texas (7/2).
aboutNorthAmerica :- wordsInArticle ~ pricing (4/2).
aboutNorthAmerica :- wordsInArticle ~ huckabee (2/0).
aboutNorthAmerica :- wordsInArticle ~ miller (2/0).
aboutLatinAmerica :- wordsInArticle ~ n, wordsInArticle ~ president (23/2).
aboutLatinAmerica :- wordsInArticle ~ brazil (15/6).
aboutLatinAmerica :- wordsInArticle ~ latin (6/3).
aboutLatinAmerica :- wordsInArticle ~ fidel (8/0).
aboutLatinAmerica :- wordsInArticle ~ lvaro (3/0).
aboutLatinAmerica :- wordsInArticle ~ bolivia (5/0).
aboutLatinAmerica :- wordsInArticle ~ canadians (2/0).
aboutAfrica :- wordsInArticle ~ africa, wordsInArticle ~ president (32/4).
aboutAfrica :- wordsInArticle ~ al (15/7).
aboutAfrica :- wordsInArticle ~ lebanon (5/0).
aboutAfrica :- wordsInArticle ~ nigerian (3/1).
aboutAsia :- wordsInArticle ~ china (35/13).
aboutAsia :- wordsInArticle ~ india (12/3).
aboutAsia :- wordsInArticle ~ e09as761 (6/0).
aboutAsia :- wordsInArticle ~ interim (6/2).
aboutAsia :- wordsInArticle ~ park (4/2).
aboutBritain :- wordsInArticle ~ britain, wordsInArticle ~ british (34/7).
aboutBritain :- wordsInArticle ~ technology (11/2).
aboutBritain :- wordsInArticle ~ brown (17/3).
aboutBritain :- wordsInArticle ~ england (5/1).
aboutBritain :- wordsInArticle ~ craft (2/0).
default aboutEurope (91/42).

================================ summary ======================================
Train error rate: 21.50% +/- 1.78% (533 datapoints)    <<
Hypothesis size: 31 rules, 69 conditions
Learning time: 6.47 sec
aboutAfrica :~ wordsInArticle ~ africa, wordsInArticle ~ president (32/4).
aboutAfrica :~ wordsInArticle ~ al (15/7).
aboutAfrica :~ wordsInArticle ~ lebanon (5/0).
aboutAfrica :~ wordsInArticle ~ nigeria (3/1).
aboutAsia :~ wordsInArticle ~ china (35/13).
aboutAsia :~ wordsInArticle ~ india (12/3).
aboutAsia :~ wordsInArticle ~ e09as761 (6/0).
aboutAsia :~ wordsInArticle ~ interim (6/2).
aboutAsia :~ wordsInArticle ~ park (4/2).

bash-3.2$ grep -wi e09as761 economist/asia/*.*.txt
economist/asia/asia.14.txt:   [activity;src=1245986;type=econo981;cat=e09as761;ord=1?]
economist/asia/asia.18.txt:   [activity;src=1245986;type=econo981;cat=e09as761;ord=1?]
economist/asia/asia.64.txt:   [activity;src=1245986;type=econo981;cat=e09as761;ord=1?]
economist/asia/asia.78.txt:   [activity;src=1245986;type=econo981;cat=e09as761;ord=1?]
economist/asia/asia.8.txt:    [activity;src=1245986;type=econo981;cat=e09as761;ord=1?]
economist/asia/asia.83.txt:   [activity;src=1245986;type=econo981;cat=e09as761;ord=1?]
Motivations for trees and rules

• Often you can find a fairly accurate classifier which is small and easy to understand.
  – Sometimes this gives you useful insight into a problem, or helps you debug a feature set.

• Sometimes features interact in complicated ways
  – Trees can find interactions (e.g., “sunny and humid”) that linear classifiers can’t
  – Again, sometimes this gives you some insight into the problem.

• Trees are very inexpensive at test time
  – You don’t always even need to compute all the features of an example.

Rest of the class: the algorithms. But first…

decision tree learning algorithms are based on information gain heuristics.
BACKGROUND: ENTROPY AND OPTIMAL CODES
Problem: design an efficient coding scheme for leaf colors:
- green
- yellow
- gold
- red
- orange
- brown
$E[n\text{bits}] = \frac{1}{2} \times 1 + \frac{1}{8} \times 3 + \frac{1}{8} \times 3 + \frac{1}{16} \times 4 + \frac{1}{16} \times 4 + \frac{1}{8} \times 3 = 2.125$
yellow, green,...

encode

100 0 ...

decode

yellow, green,...
Entropy(P) = H(P) = - \sum_x p(x) \log_2 p(x)
\[ H(P) = -\sum_x p(x) \log_2 p(x) = -p \log p - (1-p) \log(1-p) \]
DECISION TREE LEARNING: THE ALGORITHM(S)
Most decision tree learning algorithms

1. Given dataset D:
   - return \( \text{leaf}(y) \) if all examples are in the same class \( y \) ... or nearly so
   - pick the best split, on the best attribute \( a \)
     - \( a=c_1 \) or \( a=c_2 \) or ...
     - \( a<\theta \) or \( a \geq \theta \)
     - \( a \) or not(\( a \))
     - \( a \) in \( \{c_1,\ldots,c_k\} \) or not
   - split the data into \( D_1, D_2, \ldots, D_k \) and recursively build trees for each subset
2. “Prune” the tree
Most decision tree learning algorithms

1. Given dataset D:
   - return leaf(y) if all examples are in the same class \( y \) ... or nearly so...
   - pick the best split, on the best attribute \( a \)
     - \( a = c_1 \) or \( a = c_2 \) or ...
     - \( a < \theta \) or \( a \geq \theta \)
     - \( a \) or not(\( a \))
     - \( a \ in \ \{c_1, ..., c_k\} \) or not
   - split the data into \( D_1, D_2, ..., D_k \) and recursively build trees for each subset

2. “Prune” the tree

\[
H(D) = \sum_k \Pr_D(Y = y_k) \log[\Pr_D(Y = y_k)]
\]
Most decision tree learning algorithms

- “Pruning” a tree
  - avoid overfitting by removing subtrees somehow
Most decision tree learning algorithms

1. Given dataset D:
   - return \( \text{leaf}(y) \) if all examples are in the same class \( y \) ... or nearly so..
   - pick the best split, on the best attribute \( a \)
     - \( a < \theta \) or \( a \geq \theta \)
     - \( a \) or \( \text{not}(a) \)
     - \( a = c_1 \) or \( a = c_2 \) or ...
     - \( a \) in \( \{c_1, ..., c_k\} \) or not
   - split the data into \( D_1, D_2, ... \) \( D_k \) and recursively build trees for each subset

2. “Prune” the tree
Another view of a decision tree

- **Sepal_length < 5.7**
- **Sepal_width > 2.8**
Another view of a decision tree
Another view of a decision tree
Overfitting and k-NN

- Small tree $\rightarrow$ a smooth decision boundary
- Large tree $\rightarrow$ a complicated shape
- What’s the best size decision tree?

Error/Loss on training set $\mathcal{D}$
Error/Loss on an unseen test set $\mathcal{D}_{\text{test}}$
DECISION TREE LEARNING: BREAKING IT DOWN
function prPos = classifyTree(T, x)
    if T is a leaf node with counts n,p
        prPos = (p + 1)/(p + n + 2)  -- Laplace smoothing
    else
        j = T.splitAttribute
        if x(j)==0 then prPos = classifyTree(T.leftSon, x)
        else prPos = classifyTree(T.rightSon, x)
Breaking down decision tree learning

• Reduced error pruning with information gain
  – Split the data $D$ $(2/3, 1/3)$ into $D_{\text{grow}}$ and $D_{\text{prune}}$
  – Build the tree recursively with $D_{\text{grow}}$
    $T = \text{growTree}(D_{\text{grow}})$
  – Prune the tree with $D_{\text{prune}}$
    $T' = \text{pruneTree}(D_{\text{prune}}, T)$
  – Return $T'$
Breaking down decision tree learning

- First: divide & conquer to build the tree with $D_{\text{grow}}$

```plaintext
function T = growTree(X,Y)
    if $|X|<10$ or allOneLabel(Y) then
        T = leafNode($|Y==0|, |Y==1|$)  \hfill -- counts for $n,p$
    else
        for $i = 1,...,n$  \hfill -- for each attribute $i$
            $a_i = X(:, i)$  \hfill -- column $i$ of $X$
            $\text{gain}(i) = \text{infoGain}( Y, Y(a_i==0), Y(a_i==1) )$
        $j = \text{argmax}(\text{gain});$  \hfill -- the best attribute
        $a_j = X(:, j)$
        $T = \text{splitNode}( \text{growTree}(X(a_j==0),Y(a_j==0)), \text{-- left son}$
            $\text{growTree}(X(a_j==1),Y(a_j==1)), \text{-- right son}$
        $j)$
```
Breaking down decision tree learning

function e = entropy(Y)
    n = |Y|;  p0 = |Y==0|/n;  p1 = |Y==1| /n;
    e = - p0*log(p0) - p1*log(p1)
Breaking down decision tree learning

• First: how to build the tree with $D_{\text{grow}}$

function g = infoGain(Y,leftY,rightY)
    n = |Y|; nLeft = |leftY|; nRight = |rightY|;
    g = entropy(Y)
        - (nLeft/n)*entropy(leftY) - (nRight/n)*entropy(rightY)

function e = entropy(Y)
    n = |Y|; p0 = |Y==0|/n; p1 = |Y==1| /n;
    e = - p1*log(p1) - p2*log(p2)
Breaking down decision tree learning

• Reduced error pruning with information gain
  – Split the data $D$ (2/3, 1/3) into $D_{\text{grow}}$ and $D_{\text{prune}}$
  – Build the tree recursively with $D_{\text{grow}}$
    \[ T = \text{growTree}(D_{\text{grow}}) \]
  – Prune the tree with $D_{\text{prune}}$
    \[ T' = \text{pruneTree}(D_{\text{prune}}) \]
  – Return $T'$
Next: how to prune the tree with $D_{\text{prune}}$

- Estimate the error rate of every subtree on $D_{\text{prune}}$
- Recursively traverse the tree:
  - Reduce error on the left, right subtrees of $T$
  - If $T$ would have lower error if it were converted to a leaf, convert $T$ to a leaf.
We’re using the fact that the examples for sibling subtrees are disjoint.
Breaking down decision tree learning

- To estimate error rates, classify the whole pruning set, and keep some counts

```matlab
function classifyPruneSet(T, X, Y)
    T.pruneN = |Y==0|; T.pruneP = |Y==1|
    if T is not a leaf then
        j = T.splitAttribute
        aj = X(:, j)
        classifyPruneSet( T.leftSon, X(aj==0), Y(aj==0) )
        classifyPruneSet( T.rightSon, X(aj==1), Y(aj==1) )
    end
end
```

```matlab
function e = errorsOnPruneSetAsLeaf(T):
    min(T.pruneN, T.pruneP)
end
```
Breaking down decision tree learning

• Next: how to prune the tree with $D_{\text{prune}}$
  – Estimate the error rate of every subtree on $D_{\text{prune}}$
  – Recursively traverse the tree:

```plaintext
function T1 = pruned(T)
  if T is a leaf then
    -- copy T, adding an error estimate \( T.\text{minErrors} \)
    T1 = leaf(T, errorsOnPruneSetAsLeaf(T))
  else
    e1 = errorsOnPruneSetAsLeaf(T)
    TLeft = pruned(T.leftSon); TRight = pruned(T.rightSon);
    e2 = TLeft.minErrors + TRight.minErrors;
    if e1 \leq e2 then T1 = leaf(T,e1) -- cp + add error estimate
    else T1 = splitNode(T, e2) -- cp + add error estimate
```
Decision trees: plus and minus

• Simple and fast to learn
• Arguably easy to understand (if compact)
• Very fast to *use*:
  — often you don’t even need to compute all attribute values
• Can find interactions between variables (play if it’s cool and sunny or ....) and hence non-linear decision boundaries
• Don’t need to worry about how numeric values are scaled
Decision trees: plus and minus

- Hard to prove things about
- Not well-suited to probabilistic extensions
- Sometimes fail badly on problems that seem easy
  - the IRIS dataset is an example
Fixing decision trees....

• Hard to prove things about
• Don’t (typically) improve over linear classifiers when you have lots of features
• Sometimes fail badly on problems that linear classifiers perform well on
  – One solution is to build *ensembles* of decision trees
  – more on this later
RULE LEARNING: OVERVIEW
Rules for Subcategories of Economist Articles

Final hypothesis is:

aboutInternational :- wordsInArticle ~ countries, wordsInArticle ~ nations (9/4).
aboutInternational :- wordsInArticle ~ soil (7/1).
aboutInternational :- wordsInArticle ~ based, wordsInArticle ~ authorities (6/4).
aboutNorthAmerica :- wordsInArticle ~ republican, wordsInArticle ~ barack (19/0).
aboutNorthAmerica :- wordsInArticle ~ barack (9/3).
aboutNorthAmerica :- wordsInArticle ~ republican, wordsInArticle ~ americans (13/1).
aboutNorthAmerica :- wordsInArticle ~ texas (7/2).
aboutNorthAmerica :- wordsInArticle ~ pricing (4/2).
aboutNorthAmerica :- wordsInArticle ~ huckabee (2/0).
aboutNorthAmerica :- wordsInArticle ~ miller (2/0).
aboutLatinAmerica :- wordsInArticle ~ n, wordsInArticle ~ president (23/2).
aboutLatinAmerica :- wordsInArticle ~ brazil (15/6).
aboutLatinAmerica :- wordsInArticle ~ latin (6/3).
aboutLatinAmerica :- wordsInArticle ~ fidel (8/0).
aboutLatinAmerica :- wordsInArticle ~ lvaro (3/0).
aboutLatinAmerica :- wordsInArticle ~ bolivia (5/0).
aboutLatinAmerica :- wordsInArticle ~ canadians (2/0).
aboutAfrica :- wordsInArticle ~ africa, wordsInArticle ~ president (32/4).
aboutAfrica :- wordsInArticle ~ al (15/7).
aboutAfrica :- wordsInArticle ~ lebanon (5/0).
aboutAfrica :- wordsInArticle ~ nigeria (3/1).
aboutAsia :- wordsInArticle ~ china (35/13).
aboutAsia :- wordsInArticle ~ india (12/3).
aboutAsia :- wordsInArticle ~ e09as761 (6/0).
aboutAsia :- wordsInArticle ~ interim (6/2).
aboutAsia :- wordsInArticle ~ park (4/2).
aboutBritain :- wordsInArticle ~ britain, wordsInArticle ~ british (34/7).
aboutBritain :- wordsInArticle ~ technology (11/2).
aboutBritain :- wordsInArticle ~ brown (17/3).
aboutBritain :- wordsInArticle ~ england (5/1).
aboutBritain :- wordsInArticle ~ craft (2/0).
default aboutEurope (91/42).

======================================== summary ===============================
Train error rate: 21.58% +/- 1.78% (533 datapoints)  <<
Hypothesis size: 31 rules, 69 conditions
Learning time: 6.47 sec
Trees vs Rules

• For every tree with $L$ leaves, there is a corresponding rule set with $L$ rules
  – So one way to learn rules is to extract them from trees.

• But:
  – Sometimes the extracted rules can be drastically simplified
  – For some rule sets, there is no tree that is nearly the same size
  – So rules are more expressive given a size constraint

• This motivated learning rules directly
Separate and conquer rule-learning

• Start with an empty rule set
• Iteratively do this
  
  – Find a rule that works well on the data
  
  On later iterations, the data is different
  
  – Remove the examples “covered by” the rule (they satisfy the “if” part) from the data

• Stop when all data is covered by some rule
• Possibly prune the rule set
Separate and conquer rule-learning

- Start with an empty rule set
- Iteratively do this
  - Find a rule that works well on the data
    - Start with an empty rule
    - Iteratively
      - Add a condition that is true for many positive and few negative examples
      - Stop when the rule covers no negative examples (or almost no negative examples)
    - Remove the examples “covered by” the rule
- Stop when all data is covered by some rule
Separate and conquer rule-learning

function Rules = separateAndConquer(X,Y)
    Rules = empty rule set
    while there are positive examples in X,Y not covered by rule R do
        R = empty list of conditions
        CoveredX = X; CoveredY = Y;
        -- specialize R until it covers only positive examples
        while CoveredY contains some negative examples
            -- compute the “gain” for each condition x(j)==1
            ....
            j = argmax(gain); aj=CoveredX(:,j);
            R = R conjoined with condition “x(j)==1” -- add best condition
            -- remove examples not covered by R from CoveredX,CoveredY
            CoveredX = X(aj); CoveredY = Y(aj);
        end while
        Rules = Rules plus new rule R
        -- remove examples covered by R from X,Y
    end while
    ...