COEN281 -- Introduction to Pattern Recognition and Data Mining

Lecture 11: Classification & Regression Trees

Instructor: Dr. Giovanni Seni

GSeni@scu.edu

Department of Computer Engineering
Santa Clara University

Fall/18

Syllabus

Week 1	Introduction; R (Ch.1)
Week 2	Bayesian Decision Theory (Ch.2; DHS: 2.1-2.6, 2.9) Parameter Estimation (DHS: 3.1-3.4)
Week 3	Linear Discriminant Functions (Ch.3&4; DHS: 3.8.2, 5.1-5.8) Regularization (Ch.6; SE: Ch.3)
Week 4	Neural Networks (DHS: 6.1-6.6, 6.8); Deep Learning
Week 5	Support Vector Machines (Ch.9)
Week 6	**Decision Trees** (Ch. 8.1; DHS: 8.3; Ch 2 SE)
Week 7	Ensemble Methods (Ch. 8.2; SE: Ch 4, 5)
Week 8	Clustering (Ch. 10; DHS: 10.6, 10.7) Clustering (DHS: 10.9); How many clusters are there? (DHS: 10.10)
Week 9	Non-metric: Association Rules Collaborative Filtering
Week 10	Text Retrieval; Other topics

Predictive Learning

Overview

- Motivation
- Predictive Learning Example
- Tree Induction
  - Growing
  - Pruning
- Key Features
- *Surrogate* splits
- Limitations

Motivation

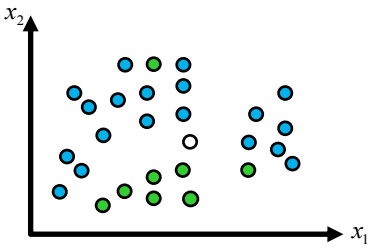
What's "tricky" with LDA, LogReg, NNets, SVMs?

- Handling of categorical features
- Sensitivity to variable scaling
- Sensitivity to bad x_j - distributions (e.g., outliers)
- Handling of missing values
- Interpretability (for NNets and SVMs)

Predictive Learning
Example

- A simple data set

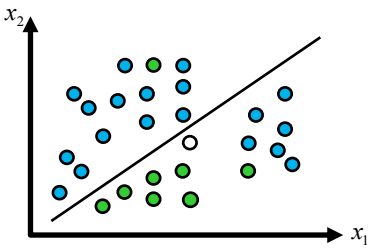
Attribute-1 (x_1)	Attribute-2 (x_2)	Class
1.0	2.0	blue
2.0	1.0	green
...	...	...
4.5	3.5	?



- What is the class of new point ○?
- Many approaches… no method is universally better; try several / use committee

Predictive Learning
Example (2)

- Ordinary Linear Regression (OLR)



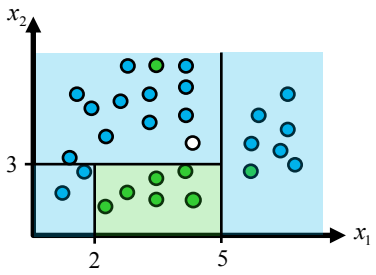
$$\hat{F}(\mathbf{x}) \geq 0 \begin{cases} \bullet \\ \text{else} \bullet \end{cases}$$

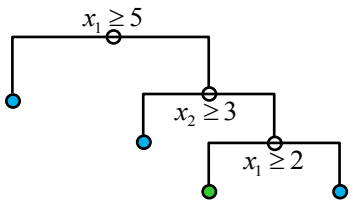
– Model: $\hat{F}(\mathbf{x}) = a_0 + \sum_{j=1}^{n} a_j x_j \qquad \forall (a_0, \mathbf{a})$ ⇒ Not flexible enough

– Accuracy (on training data): 74.07%

Decision Trees
Classification Example

- Discrete y



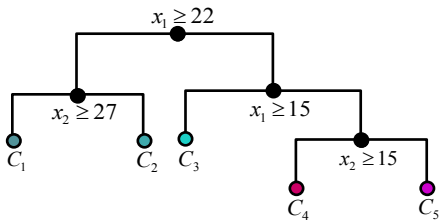


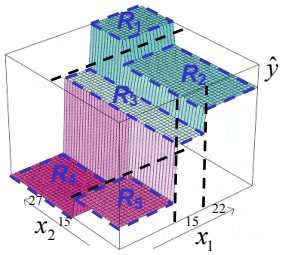
– Model: $\hat{F}(\mathbf{x}) = \sum_{m=1}^{M} \hat{c}_m I_{\hat{R}_m}(\mathbf{x})$ $\{\hat{R}_m\}_{m=1}^{M}$ = Subregions of input variable space

– Accuracy (on training data): 92.59%

Decision Trees
Regression Example

- Continuous y





- Model: $\hat{y} = T(\mathbf{x}) = \sum_{m=1}^{M} \hat{c}_m I_{\hat{R}_m}(\mathbf{x})$

where $I_A(\mathbf{x}) = 1$ if $\mathbf{x} \in A$, 0 otherwise

Tree Induction
Theory Overview

- Model: $\hat{F}(\mathbf{x}) = \sum_{m=1}^{M} \hat{c}_m I_{\hat{R}_m}(\mathbf{x})$

- Score criterion:
  - Regression – least squares – i.e., $L(y, \hat{y}) = (y - \hat{y})^2$

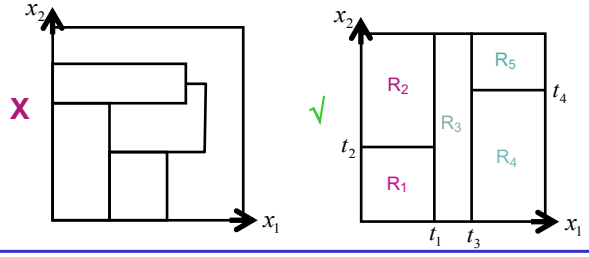
$$\{\hat{c}_m, \hat{R}_m\}_1^M = \arg\min_{\{c_m, R_m\}_1^M} \sum_{i=1}^{N} \left[y_i - \sum_{m=1}^{M} c_m I_{R_m}(\mathbf{x}_i) \right]^2$$

  - Classification – "0-1 loss" $\Rightarrow$ misclassification error

$$\{\hat{c}_m, \hat{R}_m\}_1^M = \arg\min_{\{c_m, R_m\}_1^M} \sum_{i=1}^{N} I\left(y_i \neq \sum_{m=1}^{M} c_m I_{R_m}(\mathbf{x}_i) \right)$$

Tree Induction
Theory Overview (2)

- Search Strategy
  - Unrestricted optimization with respect to $\{R_m\}_1^M$ is difficult!
  - Region restrictions
    - Disjoint
    - Cover input space
    - "Simple"



Tree Induction

Theory Overview (3)

- Search Strategy – "Simple" Regions
  - Let S_j be the set of all possible values of x_j
  - Define "splits" – $s_j \subseteq S_j$ – according to attribute type
    - Numeric (orderable): $I(x_j \in s_j) = I(x_j \le t_j)$
    - Categorical (nominal – unorderable): explicitly delineated

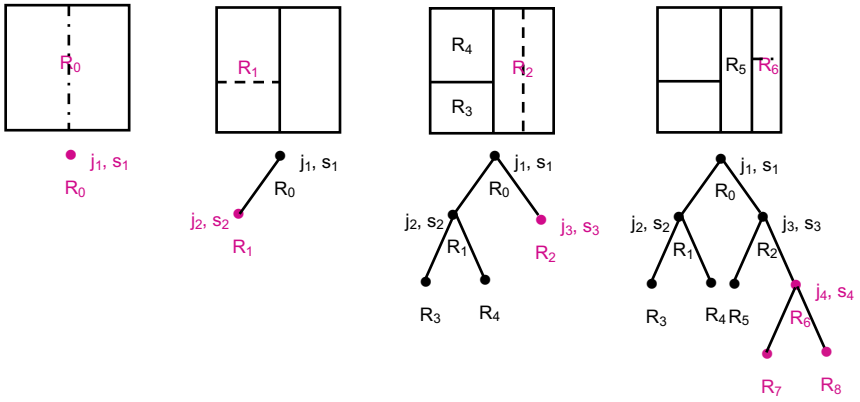
$$x_j = \text{occupation} \;\Rightarrow\; s_j = \{\text{manager}, \text{student}\}$$

- Regions as "conjunctive" rules: $x_1 \le t_1$ AND $x_2 \le t_2 \Rightarrow R_1$

$$R = \bigcap_{j=1}^{n} s_j \Rightarrow I(\mathbf{x} \in R) = \prod_{j=1}^{n} I(x_j \in s_j)$$

Tree Induction

Binary Tree Connection



- Internal nodes represent splits
- Terminal nodes represent final regions defining model

Tree Induction
Growing Algorithm

- Greedy Iterative procedure
  - Starting with a single region -- i.e., all given data
  - At the *m-th* iteration:

for each region R

  for each attribute x_j in R

    for each possible split s_j of x_j

      record change in score when we partition R into R^l and R^r

Choose (x_j , s_j) giving maximum improvement to fit

Replace R with R^l; add R^r

- When should we stop? i.e., how large should we grow the tree?

Tree Induction
Growing Algorithm (2)

- Split Scoring: *Regression*
  - We seek split-attribute j and split-point s that solve

$$\min_{j,s}\left[\min_{c_1}\sum_{\mathbf{x}_i\in R^l_{(j,s)}}(y_i-c_1)^2+\min_{c_2}\sum_{\mathbf{x}_i\in R^r_{(j,s)}}(y_i-c_2)^2\right]$$

where

$$R^l_{(j,s)}=\{\mathbf{x}\,|\,x_j\le s\}\quad\text{and}\quad R^r_{(j,s)}=\{\mathbf{x}\,|\,x_j>s\}$$

  - For any choice j and s, the inner minimization is solved by

$$\hat{c}_1=avg\{y_i|\boldsymbol{x}_i\in R^l_{(j,s)}\}\quad\text{and}\quad\hat{c}_2=avg\{y_i|\boldsymbol{x}_i\in R^r_{(j,s)}\}$$

Tree Induction
Growing Algorithm (3)

- Split Scoring: *Classification*
  - In a node m representing a region R_m with N_m points, let

$$\hat{p}_{m,k} = \frac{1}{N_m} \sum_{\mathbf{x}_i \in R_m} I(y_i = k)$$

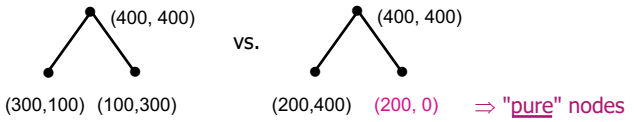
proportion of class k points in node m

  - We classify the points in node m to class $\hat{c}_m = \arg\max_k \hat{p}_{m,k}$ – i.e., *majority class*
    - $\hat{c}_m$ minimizes misclassification error:

$$\frac{1}{N_m} \sum_{\mathbf{x}_i \in R_m} I(y_i \neq \hat{c}_m)$$
$$= 1 - \hat{p}_{m,\hat{c}_m}$$

Tree Induction
Growing Algorithm (4)

- Split Scoring: Classification (Cont.)
  - Problem:



⇒ "pure" nodes are preferred

Both splits produce an error of 0.25!

- Surrogate score criterion: *"Gini" index of diversity*

$$G(\hat{p}_{m,1}, \hat{p}_{m,2}, \ldots, \hat{p}_{m,K}) = \sum_{k=1}^{K} \hat{p}_{m,k}(1 - \hat{p}_{m,k}) = 1 - \sum_{k=1}^{K} \hat{p}_{m,k}^2$$

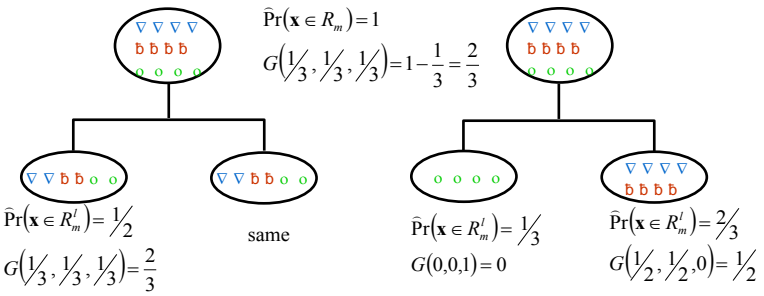
  - Same population solution!
  - Max diversity (min purity): $G(1/K, 1/K, \ldots, 1/K) = 1 - \frac{1}{K}$

Min diversity (max purity): $G(0,0,\ldots,1,0,\ldots,0) = 0$

Tree Induction
Growing Algorithm (5)

- Split Scoring: Classification (Cont.)
  - *Goodness of split:* difference between purity of parent and weighted sum of purity of daughters

$$\hat{I}_m(j, s_{jm}) = \hat{\Pr}(\mathbf{x} \in R_m) \cdot G(R_m) - \hat{\Pr}(\mathbf{x} \in R_m^l) \cdot G(R_m^l) - \hat{\Pr}(\mathbf{x} \in R_m^r) \cdot G(R_m^r)$$



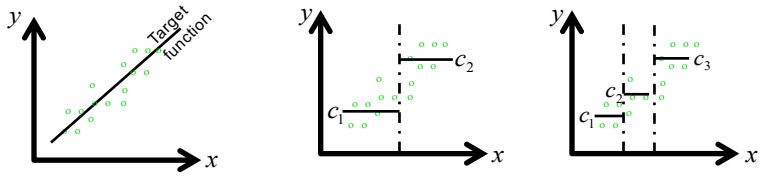
Tree Induction
Growing Algorithm (6)

- Split Scoring: Classification (Cont.)
  - Gini index not the only "diversity" measure
  - Second order entropy: $H^{(2)} = -\sum_{k=1}^{K} \hat{p}_k^2 (y = c_k \mid \mathbf{x} \in R)$
    - Has some bias towards "equal" size splits
  - Ordinary entropy: $H = -\sum_{k=1}^{K} \hat{p}_k \log \hat{p}_k$
    - Used in C4.5
    - Gets unhappy when $\hat{p}_k \approx 0$
  - Any super-linear function of $\{\hat{p}_k\}_{k=1}^{K}$ will work ⇒ Try them out!

Tree Induction

Pruning

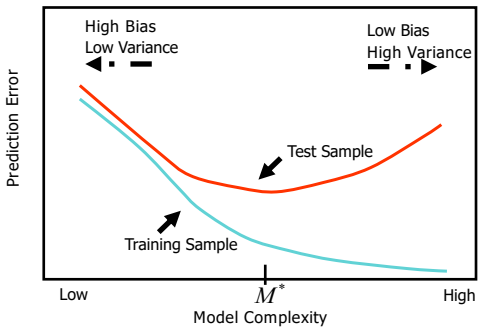
- Dilemma
  - If tree (# of regions) is too small, then piecewise constant approximation is too crude (*bias*) $\Rightarrow$ increased errors
  - If tree is too large, then it fits the training data too closely (overfitting, increased *variance*) $\Rightarrow$ increased errors



Tree Induction

Pruning (2)

- Bias-Variance tradeoff



  - Right sized tree, M^*, when *test* error is at a minimum
  - Error on the training is not a useful estimator!
    - If test set is not available, need alternative method (e.g., Cross-validation)

Tree Induction

Pruning (3)

- Two strategies
  - *Prepruning* - stop growing a branch when information becomes unreliable
    - #(R_m) – i.e., number of data points, too small

      ⇒ same bound everywhere in the tree
    - Next split not worthwhile

      ⇒ Not sufficient condition
  - *Postpruning* - take a fully-grown tree and discard unreliable parts (i.e., not supported by test data)
    - C4.5: pessimistic pruning
    - CART: cost-complexity pruning (more statistically grounded)

Key Features

- Ability to deal with irrelevant inputs
  - i.e., automatic variable subset selection
  - Measure anything you can measure
  - Score provided for selected variables ("importance")
- No data preprocessing needed
  - Naturally handle all types of variables
    - numeric, binary, categorical
  - Invariant under monotone transformations: $x_j = g_j(x_j)$
    - Variable scales are irrelevant
    - Immune to bad x_j - distributions (e.g., outliers)

Key Features (2)

- Computational scalability
  - Relatively fast: $O(nN\log N)$
- Missing value tolerant
  - Moderate loss of accuracy due to missing values
  - Handling via "surrogate" splits
- "Off-the-shelf" procedure
  - Few tunable parameters
- Interpretable model representation
  - Binary tree graphic

Surrogate Splits and Their Uses

Missing Predictor Values

- Suppose $\mathbf{x} = \{x_1, x_2, \ldots, x_l, \ldots, x_n\}$ (missing value)
- Assume we have a split $I(x_l \in S)$ in our tree

  What to do if $x_l = NA$?
  - Could delete cases $\Rightarrow$ data depletion
  - Don't go any deeper (do with what tree has learned so far for this case)
  - Allow case to follow majority (assumes all missing are somehow typical)
  - Allow missing to be a separate value (missing are indistinguishable)
  - Try to fill in (*impute*) value

Surrogate Splits and Their Uses

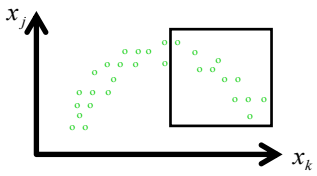
Missing Predictor Values (2)

- A more general approach
  - When considering predictor x_j for a split, only $\{\mathbf{x}_i \mid x_{ij} \neq \text{missing}\}$ are used
    - A penalty is added for number of missing
  - Suppose optimal, *primary*, split of $R_m = (x_{j(m)}, s_m)$
    - Idea: use other var's to predict this split
  - Consider "surrogate" response $\tilde{y} = I(x_{j(m)} \in s_m)$
  - For each var x_k (different from $x_{j(m)}$)
    - Compute $c_k = \max_s |corr[\tilde{y}, I(x_k \in s)]|$ ("agreement" score)
    - s_k maximizing value
    - Order $\{k \neq j(m)\}$ on decreasing $c_k \Rightarrow$ 1st surrogate, 2nd surrogate,...

Surrogate Splits and Their Uses

Missing Predictor Values (3)

- Surrogates exploit linear and nonlinear relationships among the input variables



⇒ Local correlations

i.e., Redundant var's can be helpful!

- Competitors vs. Surrogates

Class A	100
Class B	100
Class C	100

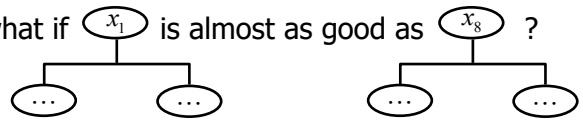
	Primary		Competitor		Surrogate	
	L	R	L	R	L	R
Class A	90	10	80	20	78	22
Class B	80	20	25	75	74	26
Class C	15	85	14	86	21	79

Surrogate Splits and Their Uses

Ranking Predictor Variables

- Which input variables are the most important?
  - i.e., relative influence or contribution in predicting the response
  - Higher importance variables are more likely to be of interest

- Variable masking
  - How to rank those vars that, while not giving the best split, may give the second, third, etc.?
  - i.e., what if x_1 is almost as good as x_8 ?



Surrogate Splits and Their Uses

Ranking Predictor Variables (2)

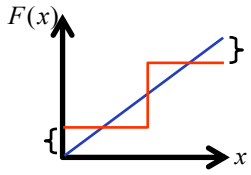
- Measure of relevance

$$\Im(x_l) = \sum_{t \in T} \hat{I}\left(v(t), \tilde{s}_{v(t)}\right) \cdot I\left(v(t) = l\right)$$

  - i.e., sum the "goodness of split" scores whenever x_l is used in a surrogate split
  - Sum is over all internal nodes in the tree
  - If x_l was used as the primary split in some node, then corresponding $\hat{I}(l, s_l)$ is used in the summation
  - We should have $\Im(x_1) \approx \Im(x_8)$ in previous scenario

Tree Limitations

- Discontinuous piecewise constant model



  - In order to have many splits you need to have a lot of data
    - In high-dimensions, you often run out of data after a few splits
  - Also note error is bigger near region boundaries

Tree Limitations (2)

- Not good for low interaction $F^*(\mathbf{x})$
  - e.g., $F^*(\mathbf{x}) = a_o + \sum_{j=1}^{n} a_j x_j$ is worst function for trees

    $= \sum_{j=1}^{n} f_j^*(x_j)$ (no interaction, additive)
  - In order for x_l to enter model, must split on it
    - Path from root to node is a product of indicators
- Not good for $F^*(\mathbf{x})$ that has dependence on many variables
  - Each split reduces training data for subsequent splits (data fragmentation)

Tree Limitations (3)

- High variance caused by greedy search strategy (local optima)
  - Errors in upper splits are propagated down to affect all splits below it

    ⇒ Small changes in data (sampling fluctuations) can cause big changes in tree

    - Very deep trees might be questionable
    - Pruning is important

What to do next?

- Live with problems
- Use other methods (when possible)
- Fix-up trees: use "ensembles“
  - Maintain advantages while dramatically increasing accuracy