CS 229: Machine Learning

Christian Shelton

UC Riverside

Lecture 13a

Slides from Lecture 13a

- From UC Riverside
  - CS 229: Machine Learning
  - Professor Christian Shelton
- DO NOT REDISTRIBUTE
  - These slides contain copyrighted material (used with permission) from
    - Elements of Statistical Learning (Hastie, et al.)
    - Pattern Recognition and Machine Learning (Bishop)
    - Machine Learning: A Probabilistic Perspective (Murphy)
  - For use only by enrolled students in the course

Regression Trees

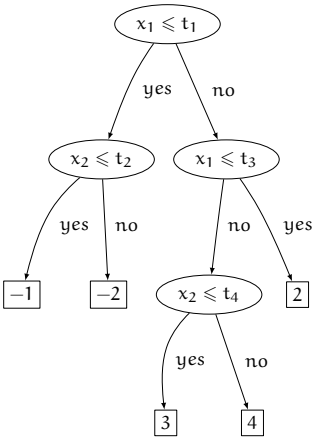
$f(x)$ represented by a (binary) tree.

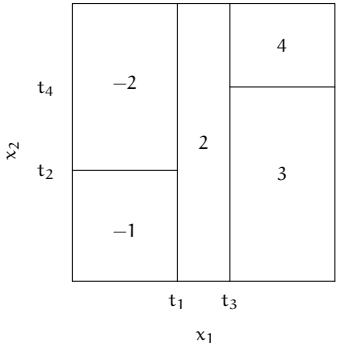
- Non-leaves are test
  - Usually uni-variate single-threshold tests
- Leaves are predicted values (real valued scalar)

Regression Trees

$f(x)$ represented by a (binary) tree.

- Non-leaves are test
  - Usually uni-variate single-threshold tests
- Leaves are predicted values (real valued scalar)

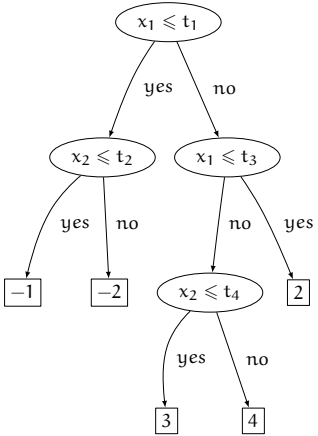


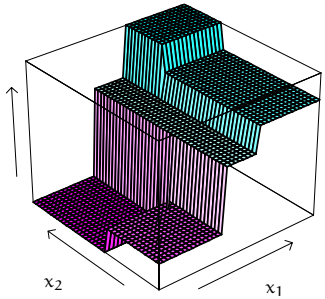


Regression Trees

$f(x)$ represented by a (binary) tree.

- Non-leaves are test
  - Usually uni-variate single-threshold tests
- Leaves are predicted values (real valued scalar)





Regression Trees

Given internal nodes, leaves can be selected easily:

$$f_{\mathcal{T}}(x) = \sum_{\text{leaf} \in \mathcal{T}} w_{\text{leaf}} \mathbf{1}(x \in \text{leaf})$$

$$L = \sum_i l\left(\sum_{\text{leaf} \in \mathcal{T}} w_{\text{leaf}} \mathbf{1}(x_i \in \text{leaf}), y_i \right)$$

$$= \sum_{\text{leaf} \in \mathcal{T}} \sum_{i | x_i \in \text{leaf}} l(w_{\text{leaf}}, y_i)$$

Regression Trees

Given internal nodes, leaves can be selected easily:

$$f_{\mathcal{T}}(x) = \sum_{\text{leaf} \in \mathcal{T}} w_{\text{leaf}} \mathbf{1}(x \in \text{leaf})$$

$$L = \sum_i l\left(\sum_{\text{leaf} \in \mathcal{T}} w_{\text{leaf}} \mathbf{1}(x_i \in \text{leaf}), y_i \right)$$

$$= \sum_{\text{leaf} \in \mathcal{T}} \sum_{i | x_i \in \text{leaf}} l(w_{\text{leaf}}, y_i)$$

How to pick internal nodes?
Generally computationally impossible (2^{2^n} trees if n tests).

Regression Trees

Learn by greedy search:

- Start at root
- Select test according to local greedy criterion
- Recurse on each subtree

Regression Trees

Learn by greedy search:

- Start at root
- Select test according to local greedy criterion
- Recurse on each subtree

Note that each test can be selected by exhaustive search:

- For each dimension
  - For each split point (finite number... why?)
    - Calculate loss if split there

Regression Trees

How large to grow?

- Grow until very few examples per leaf (5?).
- Then prune back to minimize

$$L + \alpha \|\mathcal{T}\|$$

Classification Trees

Same as a regression tree:

- Grow greedily using reduction in loss
- Prune back to minimize $\mathrm{L} + \alpha\|\mathcal{T}\|$

except, loss used in growing different from loss used in pruning:

- In growing, use impurity measure
- In pruning, use misclassification rate

Classification Trees

Impurity measures, as loss functions:

$$N_{\text{leaf}} = \text{number of training points in leaf leaf}$$

$$p_{\text{leaf},k} = \text{fraction of training points in leaf leaf of class } k$$

$$k^*_{\text{leaf}} = \arg\max_k p_{\text{leaf},k}$$

$$L = \sum_{\text{leaf} \in \mathcal{T}} N_{\text{leaf}} l_{\text{leaf}}$$

$$l_{\text{leaf}} = \frac{1}{N_{\text{leaf}}} \sum_{i | x_i \in \text{leaf}} \mathbf{1}(y_i \neq k^*_{\text{leaf}}) = 1 - p_{\text{leaf},k^*_{\text{leaf}}} \qquad \text{(misclassification rate)}$$

Classification Trees

Impurity measures, as loss functions:

$$N_{\text{leaf}} = \text{number of training points in leaf leaf}$$

$$p_{\text{leaf},k} = \text{fraction of training points in leaf leaf of class } k$$

$$k^*_{\text{leaf}} = \arg\max_k p_{\text{leaf},k}$$

$$L = \sum_{\text{leaf} \in \mathcal{T}} N_{\text{leaf}} l_{\text{leaf}}$$

$$l_{\text{leaf}} = \frac{1}{N_{\text{leaf}}} \sum_{i | x_i \in \text{leaf}} \mathbf{1}(y_i \neq k^*_{\text{leaf}}) = 1 - p_{\text{leaf},k^*_{\text{leaf}}} \qquad \text{(misclassification rate)}$$

$$l_{\text{leaf}} = \sum_{k \neq k'} p_{\text{leaf},k} p_{\text{leaf},k'} = \sum_k p_{\text{leaf},k}(1 - p_{\text{leaf},k}) \qquad \text{(Gini index)}$$

Classification Trees

Impurity measures, as loss functions:

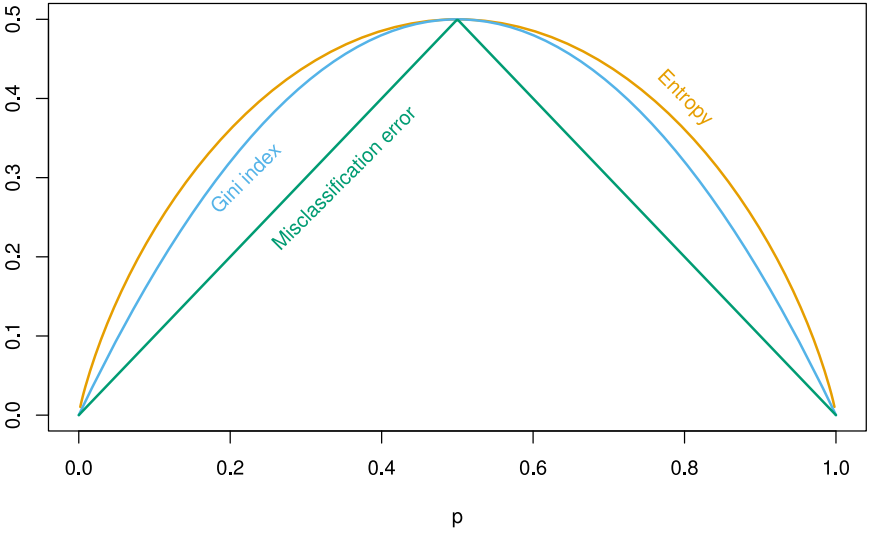
$$
\begin{aligned}
N_{\text{leaf}} &= \text{number of training points in leaf leaf} \\
p_{\text{leaf},k} &= \text{fraction of training points in leaf leaf of class } k \\
k^*_{\text{leaf}} &= \arg\max_k p_{\text{leaf},k} \\
L &= \sum_{\text{leaf} \in \mathcal{T}} N_{\text{leaf}} l_{\text{leaf}}
\end{aligned}
$$

$$
l_{\text{leaf}} = \frac{1}{N_{\text{leaf}}} \sum_{i | x_i \in \text{leaf}} \mathbf{1}(y_i \neq k^*_{\text{leaf}}) = 1 - p_{\text{leaf},k^*_{\text{leaf}}} \qquad \text{(misclassification rate)}
$$

$$
l_{\text{leaf}} = \sum_{k \neq k'} p_{\text{leaf},k} p_{\text{leaf},k'} = \sum_k p_{\text{leaf},k}(1 - p_{\text{leaf},k}) \qquad \text{(Gini index)}
$$

$$
l_{\text{leaf}} = -\sum_k p_{\text{leaf},k} \ln p_{\text{leaf},k} \qquad \text{(Cross-entropy)}
$$

Classification Trees



Decision Trees

Benefits:

- Interpretable (?)
- Missing values
- Categorical features

Problems:

- High variance
- Not smooth
- Cannot produce linear model

Variations

- Different loss weights
  - Binary: add extra weight to examples of one class:

$$p_{\text{leaf},k} = \underline{\text{weighted}} \text{ fraction of training points in leaf leaf of class } k$$

  - Multiclass: more complex

Variations

- Different loss weights
  - Binary: add extra weight to examples of one class:

$$p_{\text{leaf},k} = \underline{\text{weighted}} \text{ fraction of training points in leaf leaf of class } k$$

  - Multiclass: more complex
- Non-binary splits: not usually done (insufficient data)

Variations

- Different loss weights
  - Binary: add extra weight to examples of one class:

$$p_{\text{leaf},k} = \underline{\text{weighted}} \text{ fraction of training points in leaf leaf of class } k$$

  - Multiclass: more complex
- Non-binary splits: not usually done (insufficient data)
- Categorical predictors:
  If q values, there are $2^{q-1} - 1$ different binary splits.
  If two classes, can optimize for cross-entropy or Gini index:
  - Sort predictor values by fraction of class 1
  - Then split as if ordered predictor

Variations

- Different loss weights
  - Binary: add extra weight to examples of one class:

$$p_{\text{leaf},k} = \underline{\text{weighted}} \text{ fraction of training points in leaf leaf of class } k$$

  - Multiclass: more complex
- Non-binary splits: not usually done (insufficient data)
- Categorical predictors:
  If q values, there are $2^{q-1} - 1$ different binary splits.
  If two classes, can optimize for cross-entropy or Gini index:
  - Sort predictor values by fraction of class 1
  - Then split as if ordered predictor
- Missing values.
  - Use ternary splits (extra branch for "missing")
  - Use surrogate splits (that best mimic the best split that was chosen non-missing examples)

Pruning

Consider (recursively, bottom-up) replacing each subtree with a leaf

- Reduced error: Prune if doing so does not change training error

Pruning

Consider (recursively, bottom-up) replacing each subtree with a leaf

- Reduced error: Prune if doing so does not change training error
- Validation set: Prune if it does not degrade cross-validation error

Pruning

Consider (recursively, bottom-up) replacing each subtree with a leaf

- Reduced error: Prune if doing so does not change training error
- Validation set: Prune if it does not degrade cross-validation error
- Cost complexity criterion: Prune if it improves $\mathrm{L} + \alpha\|\mathcal{T}\|$ (α chosen by cross-validation)

Spam Classification Example

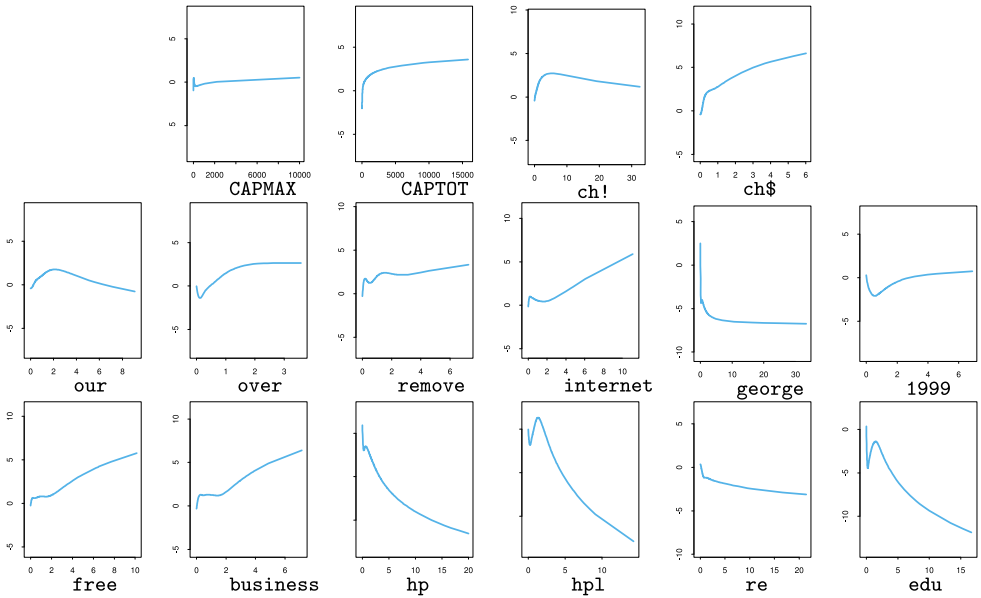
Features:

- percentage of words that match a given word (48)
- percentage of characters that match a given character (6)
- mean length of sequence of capital letters
- max length of sequence of capital letters
- total number of capital characters

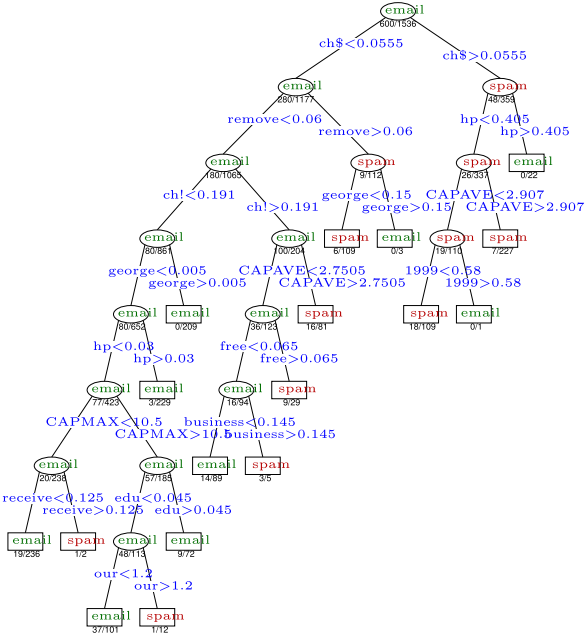
Method:

- 1536 for testing
- 3065 for training
- tried generalized linear model (cubic splines, regularization chosen from formula)
- tried classification tree

Spam Classification Example



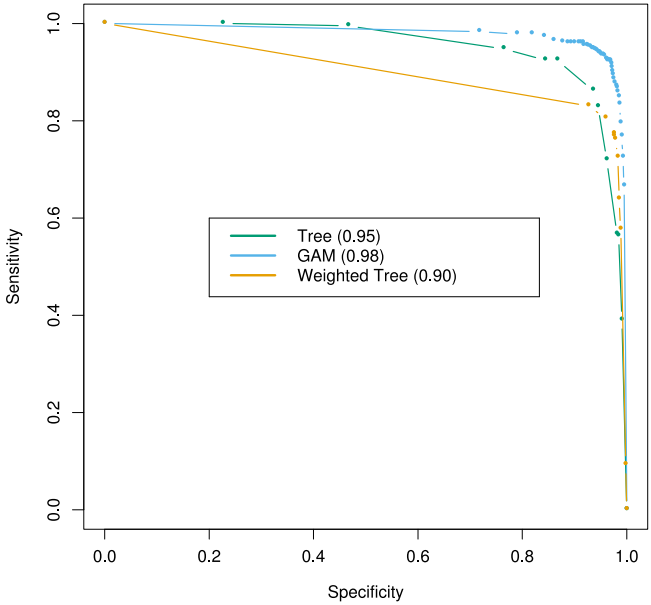
Spam Classification Example



Spam Classification Example

Specificity: True positive rate
Sensitivity: True negative rate

In this example,
Postive class: spam



Regression Tree Features

A regression tree is actually a linear model with carefully chosen features.

$$f(x) = \sum_{\text{leaf} \in \mathcal{T}} w_{\text{leaf}} \phi_{\text{leaf}}(x)$$

Regression Tree Features

A regression tree is actually a linear model with carefully chosen features.

$$f(x) = \sum_{\text{leaf} \in \mathcal{T}} w_{\text{leaf}} \phi_{\text{leaf}}(x)$$

$$\phi_{\text{leaf}}(x) = \mathbf{1}(x \in \text{leaf})$$

Regression Tree Features

A regression tree is actually a linear model with carefully chosen features.

$$f(x) = \sum_{\text{leaf} \in \mathcal{T}} w_{\text{leaf}} \phi_{\text{leaf}}(x)$$

$$\begin{aligned} \phi_{\text{leaf}}(x) &= \mathbf{1}(x \in \text{leaf}) \\ &= \prod_{(n,e) \in \text{path}(\text{leaf})} \mathbf{1}(n(x) = e) \end{aligned}$$

Growing Features

Thus, growing a regression tree is the same as

- Start with single feature: $\phi(x) = 1$
- Until tired
  - Remove one feature, $\phi(\cdot)$
  - Replace it with two new features:

$$\phi_1(x) = \phi(x)\mathbf{1}(x_i < t)$$
$$\phi_2(x) = \phi(x)\mathbf{1}(x_i \geqslant t)$$

    for some chosen i and t
  - Refit the weights (just on the two new features)

MARS

Thus, learning a MARS model is the same as

- Start with single feature: $\phi(x) = 1$
- Until tired
  - Consider one feature, $\phi(\cdot)$
  - Add two new features:

$$\phi_1(x) = \phi(x)(x_i - t)_+$$
$$\phi_2(x) = \phi(x)(t - x_i)_+$$

    for some chosen i and t
  - Refit the weights ~~(just on the two new features)~~