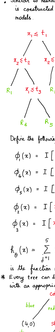
Decision Trees

- An adaptive basis function method
- Similar to neural networks in the sense that it is constructed by fitting simpler learning models



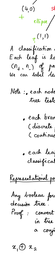
Define the following from figure:

$$\phi_1(x) = I[x_1 \le t_1 \text{ and } x_2 \le t_2]$$
$$\phi_2(x) = I[x_1 \le t_1 \text{ and } x_2 > t_2 \text{ and } x_2 \le t_3]$$
$$\phi_3(x) = I[x_1 \le t_1 \text{ and } x_2 > t_2 \text{ and } x_2 > t_3]$$
$$\phi_4(x) = I[x_1 > t_1 \text{ and } x_2 \le t_4]$$
$$\phi_5(x) = I[x_1 > t_1 \text{ and } x_2 > t_4]$$

$$f_\theta(x) = \sum_{j=1}^{5} \theta_j \phi_j(x) = \theta^T \phi(x)$$

is the function represented by the regression tree.

- Every tree can be converted into a linear combination with appropriately defined basis function.



A classification tree for a 2 class problem. Each leaf is labeled with a pair of counts (N_0, N_1) of positive and negative examples in our data. Leaf with majority class too.

Note: each node in a classification or regression tree leads to a simple feature

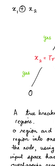
- each branch corresponds to a feature value (discrete feature) or a test on a value (continuous feature)
- each leaf node defines a value, or a classification or a pair of counts

Representational power of decision trees

Any function formula can be represented by a decision tree.

Proof: convert formula to DNF - every path in the tree is disjoint and the tree is a conjunction of disjuncts.

$x_1 \oplus x_2$



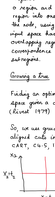
A tree breaks up the input space into regions. Each node is associated with a region and internal nodes break that region into two subregions for each split of the node, using an axis aligned cut. The input space has four disjoint cells and four leaf regions, with a one-to-one correspondence between leaf nodes and subregions.

Growing a tree

Finding an optimal partitioning of the input space given a data set D is NP-complete (Laurel 1976)

So, we use greedy algorithms to make axis aligned cuts in the input space.

CART, C4.5, ID3.



Optimization problem

Which feature j to split on and what is the value of that feature to split on?

CLASSIFICATION SETTING (discrete features)

We have 4 features $x_1, \ldots, x_4$ to select and 6 values to select from, for discrete features x_j.

Evaluate each split according to a cost function $C(A, B)$ computation

Evaluate a cost for each of the 24 splits

$$\text{cost}(j, t) = \frac{|D_{left}|}{|D|} \text{cost}(D_{left}) + \frac{|D_{right}|}{|D|} \text{cost}(D_{right})$$

For classification problems there are three choices for cost functions

(1) Misclassification rate.

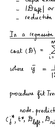
$$\text{cost}(B) = \frac{1}{|D|} \sum_{(x,y) \in D} I(y \neq \hat{y})$$

where $\hat{y}$ = majority label in B

(2) Entropy.

let p = fraction of positive examples in B

$$\text{cost}(B) = -p \log_2 p - (1-p) \log_2 (1-p)$$



(3) Gini index

$$\text{cost}(B) = 2p(1-p)$$

Greedy tree growing algorithm (classification)

procedure fit Tree (node, B, depth)

node prediction = majority label in B

$(j^*, t^*, B_{left}, B_{right}) = \text{split}(B)$ split (j^*, t^*) with highest reduction in cost

if node is not worth splitting
return node
else
node test $= \lambda x.\ x_{j^*} \le t^*$
node left = fit Tree (node, B_{left}, depth+1)
node right = fit Tree (node, B_{right}, depth+1)

Node is not worth splitting when

- node is pure
- depth exceeds max depth
- $|B_{left}|$ or $|B_{right}|$ is too small
- reduction in cost is too small

To a regression setting, we define

$$\text{cost}(B) = \sum_{i=1}^{N} (y^{(i)} - \bar{y})^2$$

where $\bar{y} = \frac{1}{|B|} \sum_{i=1}^{N} y^{(i)}$

procedure fit Tree (node, B, depth)

node prediction $= \frac{1}{|B|} \sum_{i=1}^{N} y^{(i)}$

$(j^*, t^*, B_{left}, B_{right}) = \text{split}(B)$ split (j^*, t^*) with highest reduction in cost

if node is not worth splitting
return node
else
node test $= \lambda x.\ x_{j^*} \le t^*$
node left = fit Tree (node, B_{left}, depth+1)
node right = fit Tree (node, B_{right}, depth+1)

Example

In's example is obvious.

Improved decision tree is quite complex as are the decision boundaries - very likely overfitted. Can stopping - stop growing the tree when the complexity of an additional level.

(1) Early stopping - stop growing the tree when the decrease in error is not sufficient to justify the complexity of an additional level.

(2) Post pruning - prune the full tree and then prune using a validation set to guide decision removal. Evaluate CV errors on each subtree and prune the subtree error is within 1 SE of minimum.

Example of calculation of cost reduction

Play tennis - 14 examples (9 positive, 5 negative). Attribute random consideration - wind with two values: strong, weak.

[9+, 5-]

wind = strong → [3+, 3-]

wind = weak → [6+, 2-]

$$\text{Cost}(B) = -\frac{3}{6} \log_2 \frac{3}{6} - \frac{3}{6} \log_2 \frac{3}{6} = 1$$
$$\text{Cost}(B_{left}) = -\frac{3}{8} \log_2 \frac{3}{8} - \frac{5}{8} \log_2 \frac{5}{8} = 0.94$$
$$\text{Cost}(B_{right}) = -\frac{6}{8} \log_2 \frac{6}{8} - \frac{2}{8} \log_2 \frac{2}{8} = 0.811$$

Reduction in cost due to wind:

$$0.94 - \frac{6}{14} \times 1 - \frac{8}{14} \times 0.811$$
$$= 0.048$$

Other attributes in this data set are outlook, humidity and temperature. Work as repeat task. Calculations for these attributes/features and pick the one with highest reduction in cost.

Pros and cons of trees

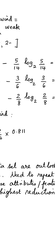
- easy to interpret
- handles continuous & discrete features
- insensitive to data scaling
- automatic feature selection
- robust to outliers

- not very accurate (axis aligned cuts)
- unstable (high variance estimators)

Random forest

Bagged ensemble of decision trees
We can reduce the variance by averaging many estimators

Train M estimators on different subsets of D, chosen randomly with replacement (bootstrap samples of D)



Ensemble average (regression)

$$h(x) = \sum_{k=1}^{M} \frac{1}{M} h^{(k)}(x)$$

Ensemble voting (classification)

$$h(x) = \underset{c}{\text{argmax}} \sum_{k=1}^{M} I(h^{(k)}(x) = c)$$

majority vote

In a random forest, we randomly choose M features to construct each decision tree so that the trees are as uncorrelated as possible.

If the trees make correlated errors, then averaging gives no advantage.