COMS30035, Machine learning: Regresssion and Classification Trees

James Cussens

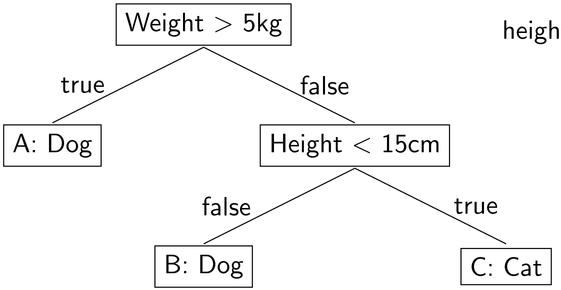
School of Computer Science
University of Bristol

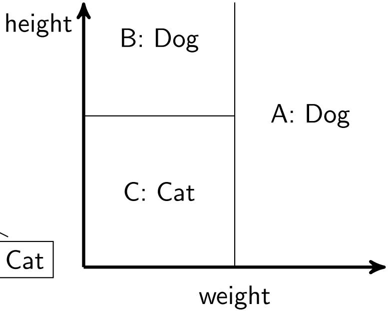
13th September 2024

Acknowledgement

- These slides are adapted from ones originally created by Edwin Simpson.

Decision Trees





Decision Trees as Partitioning Input Space

- One model is responsible for assigning a decision for each region of input space;
- The correct model for an input $\boldsymbol{x}$ is chosen by traversing the binary decision tree, following the path from the top to a leaf.
- Leaf node is responsible for assigning a decision, such as a:
  - Class label;
  - Probability distribution over class labels;
  - Scalar value (for regression tasks).

Learning the Tree Structure

- Which input variable to use at each node?
- What threshold to set for the split at each node?
- Classification and Regression Trees (CART): one of many possible learning algorithms
- Objective: greedily minimise the error
  - Regression: sum-of-squares
  - Classification: cross-entropy as used in neural networks or Gini impurity

Learning the Tree Structure

- Which input variable to use at each node?
- What threshold to set for the split at each node?
- Classification and Regression Trees (CART): one of many possible learning algorithms
- Objective: greedily minimise the error
  - Regression: sum-of-squares
  - Classification: cross-entropy as used in neural networks or Gini impurity

Learning the Tree Structure

- Which input variable to use at each node?
- What threshold to set for the split at each node?
- Classification and Regression Trees (CART): one of many possible learning algorithms
- Objective: greedily minimise the error
  - Regression: sum-of-squares
  - Classification: cross-entropy as used in neural networks or Gini impurity

Learning the Tree Structure

- Which input variable to use at each node?
- What threshold to set for the split at each node?
- Classification and Regression Trees (CART): one of many possible learning algorithms
- Objective: greedily minimise the error
  - Regression: sum-of-squares
  - Classification: cross-entropy as used in neural networks or Gini impurity

Learning the Tree Structure

- Number of possible solutions grows combinatorially with the number of input variables
- Greedy algorithm: add nodes one-at-a-time, choosing the best split at each point
  1. Start from the root node
  2. Run *exhaustive search* over each possible variable and threshold for a new node. For each variable and threshold:
     - Compute average of the target variable for each leaf of the proposed node
     - Compute the error if we stop adding nodes here
  3. Choose the variable & threshold that minimise the error
  4. Add a new node for the chosen variable and threshold.
  5. Repeat step 2 until there are only n data points associated with each leaf node.
  6. Prune back the tree to remove branches that do not reduce error by more than a small tolerance value, ϵ.

Learning the Tree Structure

- Number of possible solutions grows combinatorially with the number of input variables
- Greedy algorithm: add nodes one-at-a-time, choosing the best split at each point
  1. Start from the root node
  2. Run *exhaustive search* over each possible variable and threshold for a new node. For each variable and threshold:
     - Compute average of the target variable for each leaf of the proposed node
     - Compute the error if we stop adding nodes here
  3. Choose the variable & threshold that minimise the error
  4. Add a new node for the chosen variable and threshold.
  5. Repeat step 2 until there are only n data points associated with each leaf node.
  6. Prune back the tree to remove branches that do not reduce error by more than a small tolerance value, ϵ.

Learning the Tree Structure

- Number of possible solutions grows combinatorially with the number of input variables
- Greedy algorithm: add nodes one-at-a-time, choosing the best split at each point
  1. Start from the root node
  2. Run *exhaustive search* over each possible variable and threshold for a new node. For each variable and threshold:
     - Compute average of the target variable for each leaf of the proposed node
     - Compute the error if we stop adding nodes here
  3. Choose the variable & threshold that minimise the error
  4. Add a new node for the chosen variable and threshold.
  5. Repeat step 2 until there are only n data points associated with each leaf node.
  6. Prune back the tree to remove branches that do not reduce error by more than a small tolerance value, ϵ.

Learning the Tree Structure

- Number of possible solutions grows combinatorially with the number of input variables
- Greedy algorithm: add nodes one-at-a-time, choosing the best split at each point
  1. Start from the root node
  2. Run *exhaustive search* over each possible variable and threshold for a new node. For each variable and threshold:
     - Compute average of the target variable for each leaf of the proposed node
     - Compute the error if we stop adding nodes here
  3. Choose the variable & threshold that minimise the error
  4. Add a new node for the chosen variable and threshold.
  5. Repeat step 2 until there are only *n* data points associated with each leaf node.
  6. Prune back the tree to remove branches that do not reduce error by more than a small tolerance value, ϵ.

Learning the Tree Structure

- Number of possible solutions grows combinatorially with the number of input variables
- Greedy algorithm: add nodes one-at-a-time, choosing the best split at each point
  1. Start from the root node
  2. Run *exhaustive search* over each possible variable and threshold for a new node. For each variable and threshold:
     - Compute average of the target variable for each leaf of the proposed node
     - Compute the error if we stop adding nodes here
  3. Choose the variable & threshold that minimise the error
  4. Add a new node for the chosen variable and threshold.
  5. Repeat step 2 until there are only n data points associated with each leaf node.
  6. Prune back the tree to remove branches that do not reduce error by more than a small tolerance value, ϵ.

Learning the Tree Structure

- Number of possible solutions grows combinatorially with the number of input variables
- Greedy algorithm: add nodes one-at-a-time, choosing the best split at each point
  1. Start from the root node
  2. Run *exhaustive search* over each possible variable and threshold for a new node. For each variable and threshold:
     - Compute average of the target variable for each leaf of the proposed node
     - Compute the error if we stop adding nodes here
  3. Choose the variable & threshold that minimise the error
  4. Add a new node for the chosen variable and threshold.
  5. Repeat step 2 until there are only n data points associated with each leaf node.
  6. Prune back the tree to remove branches that do not reduce error by more than a small tolerance value, ϵ.

Learning the Tree Structure

- Number of possible solutions grows combinatorially with the number of input variables
- Greedy algorithm: add nodes one-at-a-time, choosing the best split at each point
  1. Start from the root node
  2. Run *exhaustive search* over each possible variable and threshold for a new node. For each variable and threshold:
     - Compute average of the target variable for each leaf of the proposed node
     - Compute the error if we stop adding nodes here
  3. Choose the variable & threshold that minimise the error
  4. Add a new node for the chosen variable and threshold.
  5. Repeat step 2 until there are only *n* data points associated with each leaf node.
  6. Prune back the tree to remove branches that do not reduce error by more than a small tolerance value, ϵ.

Pruning

- Balance residual training-set error against model complexity
- Start with a tree T_0
- Consider pruning each node in T_0 by combining the branches to obtain tree T
- Compute a criterion $C(T) = \sum_{\tau=1}^{|T|} e_\tau(T) + \lambda|T|$
- If $C(T) \leq C(T_0)$ keep the pruned tree, else reinstate the pruned node.

Pruning

- Balance residual training-set error against model complexity
- Start with a tree T_0
- Consider pruning each node in T_0 by combining the branches to obtain tree T
- Compute a criterion $C(T) = \sum_{\tau=1}^{|T|} e_\tau(T) + \lambda|T|$
- If $C(T) \leq C(T_0)$ keep the pruned tree, else reinstate the pruned node.

Pruning

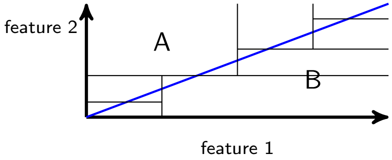
- Balance residual training-set error against model complexity
- Start with a tree T_0
- Consider pruning each node in T_0 by combining the branches to obtain tree T
- Compute a criterion $C(T) = \sum_{\tau=1}^{|T|} e_\tau(T) + \lambda |T|$
- If $C(T) \leq C(T_0)$ keep the pruned tree, else reinstate the pruned node.

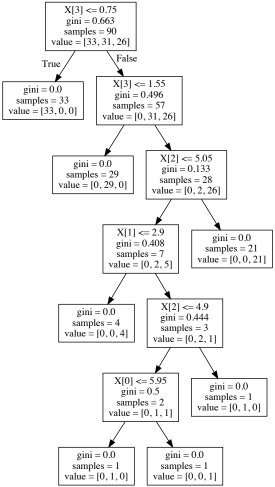
Pruning

- Balance residual training-set error against model complexity
- Start with a tree T_0
- Consider pruning each node in T_0 by combining the branches to obtain tree T
- Compute a criterion $C(T) = \sum_{\tau=1}^{|T|} e_\tau(T) + \lambda|T|$
- If $C(T) \leq C(T_0)$ keep the pruned tree, else reinstate the pruned node.

Interpretability

- The sequence of decisions is often easier to interpret than other methods (think of neural networks);
- However, sometimes small changes to the dataset cause big changes to the tree;
- If the optimal decision boundary is not aligned with the axes of an input variable, we need a lot of splits.





Reading

There is typo on p. 666 of Bishop where a minus sign has gone missing. Equation (14.32) should be:

$$Q_\tau(T) = -\sum_{k=1}^{K} p_{\tau k} \ln p_{\tau k}$$

- Bishop §14.4.
- Murphy §18.1

Problems and quizzes

- Bishop Exercise 14.10
- Bishop Exercise 14.11 (don't forget about the typo!)
- Quizzes:
  - Week 3: Trees