Classification
Lecture03: Handling Overfitting and Missing Attributes for Decision Tree

Instructor: Guo-Jun Qi
CAP 6676: Knowledge Representation
In the last lecture,

• Decision tree induction
  • Selecting attribute to expand the decision tree based on impurity measures (e.g., GINI index, entropy, classification error)
  • Splitting the training subset reaching at each node

• Stopping criterion

• Overfitting and Underfitting
  • Why should we stop growing the decision tree at a certain point?
Overfitting and Underfitting (Example)

- Two classes: 500 circular and 500 triangular data points

- Circular points:
  - A ring with radium between 0.5 and 1

- Triangular points:
  - Two parts
    - In a sphere with radium $\leq 0.5$
    - Outside the unit sphere
Overfitting and Underfitting

• Growing a binary tree by splitting training example with a threshold point alternatively along two dimensions

• Overfitting
  • Training error continues to decrease
    • The model fits well to the training set
  • Test error starts to increase
    • The model generalizes poorly to classify test data points

• We can always make a decision tree arbitrarily complex so that it can reduce the training error to zero

• But a too much complex model might not generalize well to new data points
Occam’s Razor

• Given two models with similar training errors, one should prefer the simpler model over the more complex one.

• For a complex model, there is a greater chance that it was fitted accidentally by noisy data points

• Therefore, one should penalize model complexity when evaluating the model.
Address overfitting Issue

• Pre-pruning (early stopping rule)
  • Stop the algorithm before it becomes a fully-grown tree
  • Typical stopping conditions for a node
    • Stop if all instances belong to the same class
    • Stop if all the attributes values are the same or very similar to each other
  • More restrictive conditions:
    • Stop if number of instances reaching at a node is fewer than a user-specified threshold
    • Stop if expanding the current node does not improve the impurity measure
    • Many others...
Address Overfitting

• Post-pruning
  
  • Step 1: Grow decision tree to its entirety (over-growing tree)
  
  • Step 2: Trim the nodes of the decision tree in a bottom-up fashion
  
  • Step 3: If generalization error reduces after trimming, replace sub-tree by a leaf node
    
    • Where to measure the generalization error? On a separate validation set
  
  • Step 4: The class label of the new leaf node is determined by majority voting among the instances
Handling Missing Attribute Values

• Missing values affect decision tree construction and testing in three different ways
  • How impurity measures are computed
  • How to split instances with missing values to child nodes
  • How a test instance with missing value is classified
Computing Impurity Measure

• Before splitting
  • Entropy(parent) = -0.3 \log(0.3)-0.7 \log(0.7)=0.8813
  • Missing attribute does not affect the computing of entropy before splitting

• After splitting on Refund:
  • Entropy (Refund=yes) = 0
  • Entropy (Refund =no) = -2/6 \log (2/6) – 4/6 \log(4/6) = 0.9183
  • Entropy (children) = 0.3 (0) + 0.6 (0.9183) = 0.551
  • Just computing on the instances with complete values

• Gain = 0.9 (0.8813 – 0.551) = 0.3303
  • Gain is weighted with the portion of completed instances, penalizing the choice of attributes with too many missing values
Splitting Instances

- Allow fractional instances
  - An instance with missing value is assigned with the weights to different branches in proportion to the prior distribution of the attribute value
- Prob that Refund = Yes is 3/9 (prior)
- Prob that Refund = No is 6/9 (prior)
- Assign the instance to left branch with weight 3/9
- Assign the instance to right branch with weight 6/9
Classify Instances

- Making probabilistic classification based on the different branches
  - With Prob. Of 3.67/6.67 (married), it is classified as “No”
  - With Prob. Of 3/6.67 (single/divorced), it is classified as “YES”
Expressiveness

• Decision tree provides expressive representation for learning discrete-valued function
  • However, they do not generalize well to certain types of Boolean functions
    • For example, parity function
      • Class = 1 if there are an even number of “TRUE” valued attributes
      • Class = 0 if there are an odd number of “TRUE” valued attributes
    • Must construct a complete tree (i.e., with all possible branching of attributes) to model this function
      • Bad generalization ability due to too complex tree

• Inefficient in modeling continuous attributes
  • Decision tree always divides the feature space into different classes along the axis of an attribute
  • More efficient in dividing the feature space with an arbitrary decision boundary.
Decision Boundary

• Border line between two neighboring regions of different classes is known as decision boundary

• Decision boundary is parallel to axes for decision tree because a test condition involves a single attribute one at a time
Oblique Decision Trees

• Test condition can involve multiple attributes
• More efficient decision-making
Next

• Directly learn an optimal linear classifier with hyperplane decision boundary in a high dimensional feature space

• Readings, Data Mining: the Textbook, Aggarwal.
  • Decision trees: Sec. 10.2, Sec. 10.3
  • Support Vector Machines: Sec. 10.6