Tree learning

- Top-down recursive divide and conquer algorithm
  - Start with all examples at root
  - Select best attribute/feature
  - Partition examples by selected attribute
  - Recurse and repeat

- Other issues:
  - How to construct features
  - When to stop growing
  - Pruning irrelevant parts of the tree
Choosing an attribute/feature

- Idea: a good feature splits the examples into subsets that distinguish among the class labels as much as possible... ideally into pure sets of "all positive" or "all negative"

Information gain

- How much does a feature split decrease the entropy?

\[ \text{Gain}(S, A) = \text{Entropy}(S) - \sum_{v \in \text{values}(A)} \frac{|S_A|}{|S|} \frac{\text{Entropy}(S_A)}{\text{Entropy}(S)} \]

\[ \text{Entropy}(S) = -\frac{9}{14} \log \frac{9}{14} - \frac{5}{14} \log \frac{5}{14} = 0.94 \]

\[ \text{Entropy(student=yes)} = -\frac{6}{7} \log \frac{6}{7} - \frac{1}{7} \log \frac{1}{7} = 0.62 \]

\[ \text{Entropy(student=no)} = -\frac{3}{7} \log \frac{3}{7} - \frac{4}{7} \log \frac{4}{7} = 0.99 \]

\[ \text{Gain}(S, A) = 0.94 - \left( \frac{7}{14} \times 0.62 + \frac{7}{14} \times 0.99 \right) = 0.14 \]
Dealing with Continuous Attributes

*What can you do with numeric features?*

→ Easy, use threshold or ranges to get Boolean tests

*How should you determine the thresholds?*

**Problem:**
You should consider all split points \( c \) to define node test \( X_j > c \)

**Is there an easier way?**

Dealing with Continuous Attributes

Information gain is minimized when children maintain the same distribution over output labels as the parent node

- The split should change the proportion of labels in the children
- Each child should have a higher proportion of one of the labels (different)
Dealing with Continuous Attributes

If a threshold splits two examples of the same label (say, positive), and examples on one side of the threshold (say, left), have a higher proportion than the parent distribution then:

- Examples on the other side (right) will have a higher proportion of examples with the other label (negative)
- Moving the threshold in this direction (right) until we get to an example with different label, will keep increasing the proportion of that label (positive/negative) in the respective children
Dealing with Continuous Attributes

- The highest information gain split is between examples with different labels.
  - Simple approach: go over such split points, and find the one with highest information gain

- Question:
  - How many splits should you consider for each continuous attribute?

Overfitting in Decision Trees

- Learning a tree classifying the training data perfectly may not have the best generalization
  - Algorithm would fit to noise in the data
  - Sparse dataset

- Decision trees often overfit

A hypothesis \( h \) is said to overfit the training data if there is another hypothesis \( h' \), such that \( h \) has a smaller error than \( h' \) on the training data but \( h \) has larger error on the test data.
Overfitting in Decision Trees

Expressing Bias in Decision Tree Learning

- **Inductive Bias**: Expressing preference based on our knowledge (not the data)
  - We prefer shorter trees!
- How can we include this bias when learning DT?
  - **Search bias** (dynamic)
    - We bias the search (hill climbing) to prefer shorter trees
  - **Language bias** (static)
    - Restrict the length of the trees
Expressing Bias in Decision Tree Learning

- **Static**: Fix the depth of the tree
  - Only allow trees of size $K$
  - Tune $K$ using held-out validation set
  - *Decision stump* = a decision tree with only one level
- **Dynamic**: optimize while growing the tree
  - Grow tree on training data
  - Check performance on held-out data after adding a new node
- **Post Pruning**: Prune an existing tree
  - While accuracy on validation set decreases. Bottom up:
    - For each non leaf node:
      - Replace sub-tree under node by a majority vote
      - Test accuracy on validation set

Model Selection

- **What are the algorithm hyper-parameters?**
  - *Decision trees*:
    - Depth of the tree, Pruning strategy, Pruning decision
    - Attribute selection heuristic,
    - *Other choices?*
  - *KNN*
    - Value of $K$
    - Similarity metric
  - *Every learning algorithm has a set of hyper-parameters*
Model Selection

- **Think about selecting the best model as a secondary learning problem**
  - Split the data into: (1) train set (2) test set
  - Split the train data into: (1) train set (2) validation set
  - **Training**: train m models, with different parameters
    - E.g., Different ways to control the size of the tree
  - **Validation**: estimate the prediction error for each model
  - **Test**: use the model with the least validation error
- **The secondary learning problem**:
  - New hypothesis space: m different hypothesis to chose from
  - Pick the one that minimizes validation error

K-Fold Cross validation

- **The cross validation approach is “risky” (why?)**
  - Random selection of training examples for train/test/validation
    - You could be very unlucky
  - **You validation data may not reflect the same distribution as your test data**
    - Optimizing on the validation data will lead to worse performance
K-Fold Cross validation

- You could get really unlucky
  - But that’s not likely to happen too frequently!
- **K-Fold cross validation:**
  - repeat the process K times, and average the results.
- Randomly partition the data into K equal-size subsets $S_1..S_k$
  - For $i=1…K$
    - Train a hypothesis on $S_{1..i-1}S_{i+1..k}$
    - Evaluate on $S_i$ ($\text{Err}(S_i)$)
  - Return ($\Sigma_i \text{Err}(S_i)/K$)

Precision and Recall

- Given a dataset, we train a classifier that gets 99% accuracy
- **Did we do a good job?**
- Build a classifier for brain tumor:
  - 99.9% of brain scans do not show signs of tumor
  - *Did we do a good job?*
- By simply saying “NO” to all examples we reduce the error by a factor of 10!
  - Clearly Accuracy is not the best way to evaluate the learning system when the data is heavily skewed!
- **Intuition:** we need a measure that captures the class we care about!
  (rare)
Precision and Recall

• The learner can make two kinds of mistakes:
  • False Positive
  • False Negative

<table>
<thead>
<tr>
<th>Predicted</th>
<th>True Label</th>
<th>True Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>True Positive</td>
<td>False Positive</td>
</tr>
<tr>
<td>0</td>
<td>False Negative</td>
<td>True Negative</td>
</tr>
</tbody>
</table>

• **Precision:**
  “when we predicted the rare class, how often are we right?”

• **Recall**
  “Out of all the instances of the rare class, how many did we catch?”

\[
\text{Precision} = \frac{\text{True Pos}}{\text{Predicted Pos}}
\]

\[
\text{Recall} = \frac{\text{True Pos}}{\text{Actual Pos}}
\]

Precision and Recall give us two reference points to compare learning performance

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm 1</td>
<td>0.5</td>
<td>0.4</td>
</tr>
<tr>
<td>Algorithm 2</td>
<td>0.7</td>
<td>0.1</td>
</tr>
<tr>
<td>Algorithm 3</td>
<td>0.02</td>
<td>1</td>
</tr>
</tbody>
</table>

• Which algorithm is better?  
  *We need a single score*

• Option 1: **Average**  
  \[
  \frac{P + R}{2}
  \]

• Option 2: **F-Score**  
  \[
  2 \frac{PR}{P + R}
  \]

**Properties of f-score:**
• Ranges between 0-1
• Prefers precision and recall with similar values