Big Data Analytics

Special Topics for Computer Science
CSE 4095-001  CSE 5095-005
Feb 17

Fei Wang
Associate Professor
Department of Computer Science and Engineering
fei_wang@uconn.edu
Prediction I

Slides adapted from the course notes of CSE 415 in UW.
http://courses.cs.washington.edu/courses/cse415/
Learning System
Training and Testing

Data acquisition

Training set (observed)

Universal set (unobserved)

Testing set (unobserved)

Practical usage
Settings

Supervised learning

Unsupervised learning

Semi-supervised learning
K-Nearest Neighbor

- Training method:
  - Save the training examples
- At prediction time:
  - Find the \( k \) training examples \((x_1,y_1),\ldots,(x_k,y_k)\) that are closest to the test example \( x \)
  - Predict the most frequent class among those \( y_i \)’s.
Decision Tree Hypothesis Space

- **Internal nodes** test the value of particular features $x_j$ and branch according to the results of the test.

- **Leaf nodes** specify the class $h(x)$.

[Diagram of decision tree]

Suppose the features are **Outlook** ($x_1$), **Temperature** ($x_2$), **Humidity** ($x_3$), and **Wind** ($x_4$). Then the feature vector $x = (\text{Sunny, Hot, High, Strong})$ will be classified as **No**. The **Temperature** feature is irrelevant.
Decision Tree Hypothesis Space

If the features are continuous, internal nodes may test the value of a feature against a threshold.
Decision Tree Decision Boundaries

Decision trees divide the feature space into axis-parallel rectangles, and label each rectangle with one of the $K$ classes.
Learning Algorithm for Decision Trees

The same basic learning algorithm has been discovered by many people independently:

\[
\text{GROWTree}(S)
\]

\[
\text{if } (y = 0 \text{ for all } \langle x, y \rangle \in S) \text{ return new leaf(0)}
\]

\[
\text{else if } (y = 1 \text{ for all } \langle x, y \rangle \in S) \text{ return new leaf(1)}
\]

\[
\text{else}
\]

\[
\text{choose best attribute } x_j
\]

\[
S_0 = \text{ all } \langle x, y \rangle \in S \text{ with } x_j = 0;
\]

\[
S_1 = \text{ all } \langle x, y \rangle \in S \text{ with } x_j = 1;
\]

\[
\text{return new node}(x_j, \text{GROWTree}(S_0), \text{GROWTree}(S_1))
\]

But how should we select the best attribute?
Attribute Selection

- **outlook**
  - sunny: yes, yes, no
  - overcast: yes, yes, yes
  - rainy: yes, yes, no

- **temperature**
  - hot: yes, yes, yes
  - mild: yes, yes, yes
  - cool: yes, yes, no

- **humidity**
  - high: yes, yes, yes
  - normal: yes, yes, no

- **windy**
  - false: yes, yes, yes
  - true: yes, yes, yes
Criterion for Attribute Selection

- Which is the best attribute?
  - The one which will result in the smallest tree
  - Heuristic: choose the attribute that produces the “purest” nodes

- Need a good measure of purity!
  - Maximal when?
  - Minimal when?
Information Gain

Split over whether Balance exceeds 50K

<table>
<thead>
<tr>
<th>Less or equal 50K</th>
<th>Over 50K</th>
</tr>
</thead>
</table>

Split over whether applicant is employed

<table>
<thead>
<tr>
<th>Unemployed</th>
<th>Employed</th>
</tr>
</thead>
</table>
Impurity/Entropy
Entropy

• Entropy = \( \sum_i - p_i \log_2 p_i \)
  
  \( p_i \) is the probability of class \( i \)
  
  Compute it as the proportion of class \( i \) in the set.

• Entropy comes from information theory. The higher the entropy the more the information content.

What does that mean for learning from examples?
Binary Case

• What is the entropy of a group in which all examples belong to the same class?
  – entropy = \(-1 \log_2 1 = 0\)
  not a good training set for learning

• What is the entropy of a group with 50% in either class?
  – entropy = 
    \[-0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1\]
  good training set for learning
Information Gain

**Information Gain** = \( \text{entropy(parent)} - \left[ \text{average entropy(children)} \right] \)

\[
\text{parent entropy} = - \left( \frac{14}{30} \cdot \log_2 \frac{14}{30} \right) - \left( \frac{16}{30} \cdot \log_2 \frac{16}{30} \right) = 0.996
\]

\[
\text{child entropy} = - \left( \frac{13}{17} \cdot \log_2 \frac{13}{17} \right) - \left( \frac{4}{17} \cdot \log_2 \frac{4}{17} \right) = 0.787
\]

\[
\text{child entropy} = - \left( \frac{1}{13} \cdot \log_2 \frac{1}{13} \right) - \left( \frac{12}{13} \cdot \log_2 \frac{12}{13} \right) = 0.391
\]

\[
\text{(Weighted) Average Entropy of Children} = \left( \frac{17}{30} \cdot 0.787 \right) + \left( \frac{13}{30} \cdot 0.391 \right) = 0.615
\]

\[
\text{Information Gain} = 0.996 - 0.615 = 0.38
\]
Entropy Based Decision Tree Construction

Training Set $S$

$\mathbf{x}_1 = (f_{11}, f_{12}, \ldots, f_{1m})$

$\mathbf{x}_2 = (f_{21}, f_{22}, \ldots, f_{2m})$

$\vdots$

$\mathbf{x}_n = (f_{n1}, f_{n2}, \ldots, f_{nm})$

What feature should be used?

What values?

Quinlan suggested information gain in his ID3 system and later the gain ratio, both based on entropy.
Using Information Gain for Constructing Decision Tree

Full Training Set $S$  

**Attribute A**

$v_1$  $v_2$  $v_k$

Construct child nodes for each value of $A$. Each has an associated subset of vectors in which $A$ has a particular value.

Set $S'$  

$S' = \{ s \in S \mid \text{value}(A) = v_1 \}$

Information gain has the disadvantage that it prefers attributes with large number of values that split the data into small, pure subsets. Quinlan’s gain ratio did some normalization to improve this.
Gain Ratio

Split Information: \(-(17/30)*\log(17/30)-(13/30)*\log(13/30)\)=0.987

Gain Ratio: \(0.38/0.987 = 0.385\)
Overfitting in Decision Trees

Consider adding a noisy training example:
*Sunny, Hot, Normal, Strong, PlayTennis=No*
What effect on tree?
Overfitting

Consider error of hypothesis $h$ over

- training data: $error_{train}(h)$
- entire distribution $\mathcal{D}$ of data: $error_{\mathcal{D}}(h)$

Hypothesis $h \in H$ overfits training data if there is an alternative hypothesis $h' \in H$ such that

$$error_{train}(h) < error_{train}(h')$$

and

$$error_{\mathcal{D}}(h) > error_{\mathcal{D}}(h')$$
Overfitting in Decision Tree Learning

![Graph showing accuracy of decision tree learning over the size of the tree. The graph compares accuracy on training data and test data. The accuracy on training data increases with the size of the tree, while the accuracy on test data plateaus and then decreases.]
Avoiding Overfitting

How can we avoid overfitting?

- Stop growing when data split not statistically significant
- Grow full tree, then post-prune

How to select “best” tree:

- Measure performance over training data
- Measure performance over separate validation data set
- Add complexity penalty to performance measure
Reduced-Error Pruning

Split data into *training* and *validation* set

Do until further pruning is harmful:

1. Evaluate impact on *validation* set of pruning each possible node (plus those below it)

2. Greedily remove the one that most improves *validation* set accuracy
The tree is pruned back to the red line where it gives more accurate results on the test data.
Rule Post-Pruning

1. Convert tree to equivalent set of rules
2. Prune each rule independently of others
3. Sort final rules into desired sequence for use

Perhaps most frequently used method (e.g., C4.5)
Converting A Tree to Rules

- **Outlook**
  - Sunny
  - Overcast
    - Yes
  - Rain
    - Wind
      - Strong
        - No
      - Weak
        - Yes
  - Humidity
    - High
    - Normal
      - No
      - Yes
IF $(Outlook = Sunny) \ AND \ (Humidity = High)$
THEN $PlayTennis = No$

IF $(Outlook = Sunny) \ AND \ (Humidity = Normal)$
THEN $PlayTennis = Yes$

...
Scaling Up

- ID3, C4.5, etc. assume data fits in main memory (OK for up to hundreds of thousands of examples)
- SPRINT, SLIQ: multiple sequential scans of data (OK for up to millions of examples)
- VFDT: at most one sequential scan (OK for up to billions of examples)