CPSC 340: Machine Learning and Data Mining

Supervised Learning and Decision Tree Learning

September 14, 2015
Admin

• Tutorials have started today:
  – 11am, 2pm, and 4pm in DMP 201.
  – 5pm in DMP 101.

• Office hours tomorrow:
  – 10am in ICICS X836
  – 4pm in ICICS 146

• Assignment 1 due Friday
  – Get further help on Piazza.
  – Q1 might be input as a UBC survey.
  – Setting up Handin for submission.
Motivating Example: Food Allergies

• You frequently start getting an upset stomach

• You suspect an adult-onset food allergy.
Motivating Example: Food Allergies

• To solve the mystery, you start a food journal:

<table>
<thead>
<tr>
<th>Egg</th>
<th>Milk</th>
<th>Fish</th>
<th>Wheat</th>
<th>Shellfish</th>
<th>Peanuts</th>
<th>...</th>
<th>Sick?</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.7</td>
<td>0</td>
<td>0.3</td>
<td>0</td>
<td>0</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>0.3</td>
<td>0.7</td>
<td>0</td>
<td>0.6</td>
<td>0</td>
<td>0.01</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.8</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0.3</td>
<td>0.7</td>
<td>1.2</td>
<td>0</td>
<td>0.10</td>
<td>0.01</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>0.3</td>
<td>0</td>
<td>1.2</td>
<td>0.3</td>
<td>0.10</td>
<td>0.01</td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

• But it’s hard to find the pattern:
  – You can’t isolate and only eat one food at a time.
  – You may be allergic to more than one food.
  – The quantity matters: a small amount may be ok.
  – You may be allergic to specific interactions.
## Supervised Learning

- We can formulate this as **supervised learning**:
  - Input is a set of continuous **features**.
    - the quantities of food eaten.
  - Output is a desired target **label**:
    - Whether or not we got sick.
  - Supervised learning: **learn map from features to labels**.
    - Given foods, map predicts whether you will get sick.
Supervised Learning

• Supervised learning general case:
  – Input: features and corresponding labels for objects.
  – Output: program that maps from features to labels.

• Most useful when:
  – You don’t know how to write a program to do task.
  – But have input/output examples.

• The most successful machine learning technique:
  – Spam filtering, Microsoft Kinect, speech recognition.

• Today we will learn about one approach:
  – Decision trees.
But first....

• Is this data IID? No!

• What cleaning/preprocessing steps?
  - Scaling (convert to grams)
  - Little data: common allergic ingredients (lactose, gluten, ...)
  - Look at different types of food (baker's bread vs. MacDonald's)
Decision Tree

- **milk > 0.5?**
  - no
  - **Oranges > 0.75?**
    - no
      - not sick
    - yes
      - sick
  - yes
    - **Sick**
Decision Tree

1. Start at root note.
2. Branch using splitting rule.
3. Leaf nodes are labeled.
   a) If leaf, return label.
   b) Otherwise, go to 2.
Decision Tree as a Program

- Think of this as a simple program:

    If (milk > 0.5)
    Return ‘sick’

    Else {
    If (oranges > 0.75)
    Return ‘sick’
    Else
    Return ‘not sick’

    }
Decision Tree Learning

• We could write decision program by hand.
• But might be hard:
  – Huge number of variables.
  – Sequences of rules might be hard to find.
• **Decision tree learning:**
  – Use data to automatically write the program.
• Usual ‘greedy’ procedure:
  – Start with all data and learn one simple rule.
  – Split data based on rule, recurse on subsets.
Learning A Decision Stump

- **Decision stump**: decision tree with one rule.

![Decision Stump Diagram](image)

- How do we find the variable and threshold?
  1. Define a ‘score’ for the rule.
  2. Search for the rule with the best score.
Decision Stump: Accuracy Score

• Most intuitive score: **classification accuracy**.
  – “If we use this rule, how many objects do we label correctly?”

• Computing classification accuracy:
  – Input is a rule like \((\text{eggs} > 2)\).
  – Go through all objects, and find out which class is more likely given given rule.
    • E.g., \((\text{eggs} > 2)\) implies ‘sick’ more often than ‘not sick’, and \((\text{eggs} \leq 2)\) implies ‘not sick’ more often than ‘sick’.
  – Go through all objects again, counting how many times the rule predicts the correct object label.
    • E.g., how many times did \((\text{eggs} > 2)\) actually gave ‘sick’, plus how many times did \((\text{eggs} \leq 2)\) actually gave ‘not sick’.
  – Output: sum of these counts divided by number of objects.

• Accuracy score of ‘1’: rule gives perfect prediction.
• Accuracy score of ‘0.50’: rule tells you nothing.
  (if you only have two classes)
Decision Stump: Rule Search

- Accuracy score evaluates how ‘good’ a rule is.
- To find the ‘best’ stump, find the ‘best’ rule.
- Attempt 1 (exhaustive search):

  Compute score of (eggs > 0)  Compute score of (milk > 0)  ...
  Compute score of (eggs > 0.01)  Compute score of (milk > 0.01)  ...
  Compute score of (eggs > 0.02)  Compute score of (milk > 0.02)  ...
  Compute score of (eggs > 0.03)  Compute score of (milk > 0.03)  ...
  ...
  ...
  Compute score of (eggs > 99.99)  Compute score of (milk > 0.99)  ...

- As you go, keep track of the highest score.
- Return rule with highest score.
Cost of Decision Stumps (Attempt 1)

• How much does this cost?

• Assume we have:
  – ‘n’ objects (days that we measured).
  – ‘d’ features (foods that we measured).
  – ‘t’ thresholds (>0, >0.01, >0.02,...)

• Computing the score costs $O(n)$:
  – We need to go through all ‘n’ examples.

• Total cost is $O(ndt)$:
  – Need to compute score for a total of $d*t$ rules.

• Can we do better?
  (if you are not familiar with “$O(n)$” see notes on webpage)
Cost of Decision Stumps (Attempt 1)

• We can ignore rules outside feature ranges:
  – E.g., we never have (eggs > 50) in our data.
  – These rules can never improve accuracy.
  – Restrict the thresholds to the range of each feature.

• Most of the thresholds give the same score.
  – E.g., if never have (eggs == 0.05) in the data, then
    (eggs > .04) and (eggs > 0.05) have the same score.’
  – Restrict thresholds to values of the features in data.
Decision Stump: Rule Search

- Accuracy score evaluates how ‘good’ a rule is.
- To learn ‘best’ stump, find the ‘best’ rule.
- Attempt 2 (search over features in data):

Compute score of (eggs > 0)     Compute score of (milk > 0.5)     ...
Compute score of (eggs > 1)     Compute score of (milk > 1)     ...
Compute score of (eggs > 2)     Compute score of (milk > 1.5)     ...
Compute score of (eggs > 3)     Compute score of (milk > 2)     ...
Compute score of (eggs > 4)     Compute score of (milk > 2)     ...
Compute score of (eggs > 5)     Compute score of (milk > 2)     ...

- Now at most ‘n’ thresholds for each feature.
- So we now consider only $O(nd)$ rules.
- Total cost changes from $O(ndt)$ to $O(n^2d)$. 
Decision Stump: Rule Search

• Do we have to compute score from scratch?
  – Rule (eggs > 1) and (eggs >2) have same score, except when (eggs == 2).
  – Sort the examples based on ‘eggs’.
  – Go through the rules in order, updating score.

• Sorting costs $O(n \log n)$ per feature.
• You do at most $O(n)$ score updates per feature.
• Total is down from $O(n^2d)$ to $O(nd \log n)$.
• This is a good runtime:
  – $O(nd)$ is size of data, this is only slightly bigger.
Greedily Making Trees From Stumps

[all data] ← fit stump
Greedily Making Trees From Stumps

1. All data
2. Fit stump
3. Milk > 0.5
   - No
   - Not Sick
   - Yes
   - Sick
Greedily Making Trees From Stumps

- All data
  - fit stump
    - milk > 0.5
      - no
        - not sick
      - yes
        - sick
          - take data where milk > 0.5, fit stump
          - take data where milk > 0.5, fit stump
Greedily Making Trees From Stumps

- $m_{12} > 0.5$
  - no
  - $\text{oranges} > 0.5$
    - no
      - not sick
    - yes
      - sick
  - yes
    - lactase $> 0$
      - no
      - sick
      - not sick
Greedily Making Trees From Stumps

Stop when:
- only have one label left.
- reach user-defined maximum depth.
Issues with Decision Trees

• Advantages:
  – Interpretable.
  – Fast to learn.
  – Very fast to classify

• Disadvantages:
  – Hard to find optimal set of rules.
  – Rules are very simple.
  – Not the most accurate.

• Issues:
  – Can you revisit a feature?
  – More complicated rules?
  – Is accuracy the best score?
  – What depth?
Can you re-visit a feature?

• Yes.

Knowing you had ice cream makes small milk quantities more relevant.
Can you have more complicated rules?

• Yes:

  \[ \text{milk + ice cream} > 0.5 \]

  \begin{align*}
  \text{no} & \quad \text{sick} \\
  \text{not sick} & \quad \text{yes}
  \end{align*}

• But searching for best rule is more expensive.
Which Score Function?

• Shouldn’t we just use accuracy score?
  – For leafs: yes, just maximize accuracy.
  – For internal nodes: maybe not.
    • There may be no simple rule like (eggs > 0.5) that improves accuracy.

• Most common score in practice: information gain.
  – How much does entropy (“randomness”) of labels decrease if I use this rule to split the data?
  – Hope is that later rules on ‘less random’ data will be able to improve accuracy.
Summary

• Supervised learning: using data to build program that outputs labels from input features.

• Decision trees: making a decision via a sequence of simple rules.

• Decision stumps: very simple decision trees that we can very efficiently fit.

• We can greedily construct decision trees from a sequence of decision stumps.
  – Fast/interpretable, but may not be very accurate.