Data Mining on Streams using Decision Trees

CS 536: Machine Learning
Instructor: Michael Littman
TA: Yihua Wu

Outline

- Introduction to Data Streams
  - Motivation
  - Models
- Summarizing data streams
  - Sampling-based
  - Sketch-based
- Overview of traditional DT learning Algs
- DT learning Algs on streams
  - One concept
  - Multiple concepts (concept drifts)

Puzzle: Find missing numbers

- Paul permutes numbers 1..n, and shows all but one to Carol, in the permuted order, one after the other.
- Carol must find the missing number.

Carol cannot remember all the numbers she has been shown.

Carol finds the missing number...

- Carol cumulates the sum of all the numbers she is being shown. At the end, she can subtract this sum from the total sum of the numbers 1..n.
- Uses $O(\log n)$ bits to store the partial sum.
- Performs one + each time Paul shows a number. Takes $O(\log n)$ time per number.
- At the end, computes the missing number with one subtraction. Takes $O(\log n)$ time for final computation.
Finding two missing numbers...

- What if Paul shows all but two numbers?
- Carol keeps the sum AND product of the numbers Paul shows her.
  - \( O(\log n) = O(n \log n) \) bits and time.
- Alternatively, Carol keeps the sum AND sum of squares of the numbers Paul shows her.
  - As before: \( O(\log n) \) storage, \( O(\log n) \) process time and \( O(\log n) \) compute time.

Desiderata

- \( \text{Polylog}(n) \) per item processing time
- \( \text{Polylog}(n) \) space stored
- \( \text{Polylog}(n) \) for computing functions on \( s \)
- Can only scan stream once!

<table>
<thead>
<tr>
<th></th>
<th>Traditional</th>
<th>Stream</th>
</tr>
</thead>
<tbody>
<tr>
<td>num of passes</td>
<td>multiple</td>
<td>single</td>
</tr>
<tr>
<td>time</td>
<td>unlimited</td>
<td>polylog</td>
</tr>
<tr>
<td>memory</td>
<td>unlimited</td>
<td>polylog</td>
</tr>
<tr>
<td>result</td>
<td>accurate</td>
<td>approximate</td>
</tr>
<tr>
<td>num of concepts</td>
<td>one</td>
<td>multiple</td>
</tr>
</tbody>
</table>

The Data Stream Phenomenon

- Highly detailed, automatic, rapid data feeds.
- 3 Billion Telephone Calls in US each day
- 30 Billion emails daily, 1 Billion SMS
- Scientific data: NASA’s observation satellites generate billions of readings each day.
- IP Network Traffic: up to 1 Billion packets per hour per router. Each ISP has many (hundreds) of routers!
- Compare to “human scale” data: “only” 1 billion worldwide credit card transactions per month.
- Need for near-real time analysis of data feeds. (classification; extreme events—heavy hitters, deltoids; etc.)

Models of Data Streams

- Signal \( s[1..n] \). \( n \) is universe size.
- Three models:
  - Time-series model: \( s(1), s(2),..., s(t),.... \)
  - Cash Register model: \( s_t(j) = s_{t-1}(j) + a_j(t). \)
    \( a_j(t) > 0 \). (insert only)
  - Turnstile model: \( s_t(j) = s_{t-1}(j) + u_j(t). \) (both insertion and deletion)
Summarizing Data Streams

**Synopsis**
- A small space representation of the data stream
- Can be rapidly updated on the fly
- Approx results: within error bounds with high probability

**Two approaches**
- Samples: use a subset to present all
- Sketches: all records observed, but store using much less space (e.g. sum)

---

What is a Decision Tree?

- **Decision Tree** is a classification model for finding patterns in data using tree structures. Can be used to explain data and make predictions from it.
  - **Internal Nodes**: tests on examples’ attribute values.
  - **Leaf Nodes**: class labels.
- Applies to **categorical outputs**.
- **ID3**: a DTLA, only categorical attributes.

---

Learning Process

**Input**
- A Large Data Set

**Decision Tree Learning System**

**Output**
- Decision Tree Structure
  - Predicted class label
  - Accuracy

**Decision Tree Structure**

**Accuracy**

---

Here is an Example...

```
<table>
<thead>
<tr>
<th>outlook</th>
<th>temp</th>
<th>humidity</th>
<th>windy</th>
<th>play</th>
</tr>
</thead>
<tbody>
<tr>
<td>sunny</td>
<td>hot</td>
<td>high</td>
<td>false</td>
<td>no</td>
</tr>
<tr>
<td>sunny</td>
<td>hot</td>
<td>high</td>
<td>true</td>
<td>no</td>
</tr>
<tr>
<td>overcast</td>
<td>hot</td>
<td>high</td>
<td>false</td>
<td>yes</td>
</tr>
<tr>
<td>rainy</td>
<td>mild</td>
<td>high</td>
<td>false</td>
<td>yes</td>
</tr>
<tr>
<td>rainy</td>
<td>cool</td>
<td>normal</td>
<td>false</td>
<td>yes</td>
</tr>
<tr>
<td>rainy</td>
<td>cool</td>
<td>normal</td>
<td>true</td>
<td>no</td>
</tr>
<tr>
<td>overcast</td>
<td>cool</td>
<td>normal</td>
<td>true</td>
<td>yes</td>
</tr>
<tr>
<td>sunny</td>
<td>mild</td>
<td>high</td>
<td>false</td>
<td>no</td>
</tr>
<tr>
<td>sunny</td>
<td>cool</td>
<td>normal</td>
<td>false</td>
<td>yes</td>
</tr>
<tr>
<td>rainy</td>
<td>mild</td>
<td>normal</td>
<td>false</td>
<td>yes</td>
</tr>
<tr>
<td>sunny</td>
<td>mild</td>
<td>normal</td>
<td>true</td>
<td>yes</td>
</tr>
<tr>
<td>overcast</td>
<td>mild</td>
<td>high</td>
<td>true</td>
<td>yes</td>
</tr>
<tr>
<td>overcast</td>
<td>hot</td>
<td>high</td>
<td>true</td>
<td>yes</td>
</tr>
<tr>
<td>rainy</td>
<td>mild</td>
<td>high</td>
<td>true</td>
<td>no</td>
</tr>
</tbody>
</table>
```

**Training Data**

**Test Data**
Building Decision Trees

- **Key Idea**
  - Evaluate splits for each attribute;
  - Pick the best attribute to test at root;
  - Divide the training data into subsets $D_i$ for each value the attribute can take on;
  - Recurse the tree construction for each $D_i$.

- **Attribute Selection**
  - Find an attribute that divides data into as pure subsets as possible.

Information Gain

- $C$ has $m$ possible values, and $A$ has $m$.
- For any dataset $D$,

$$\text{Split - Info}(A, D) = \sum_{i=1}^{m} w_i \cdot H \left( \frac{n_{i,1} - n_i}{n_i} \cdot \frac{n_{i,2} - n_i}{n_i} \ldots \frac{n_{i,m} - n_i}{n_i} \right)$$

$$\text{IG}(A, D) = \text{Info}(D) - \text{Split - Info}(A, D), \text{where } w_i = \frac{|D_i|}{|D|}$$

Sufficient Statistics

- Recall that...

$$\text{Split - Info}(A, D) = \sum_{i=1}^{m} w_i \cdot H \left( \frac{n_{i,1} - n_i}{n_i} \cdot \frac{n_{i,2} - n_i}{n_i} \ldots \frac{n_{i,m} - n_i}{n_i} \right)$$

$$= \sum_{i=1}^{m} \frac{|D_i|}{|D|} \cdot \sum_{k=1}^{l} \left( \frac{n_{i,k}}{n_i} \cdot \log \frac{n_{i,k}}{n_i} \right)$$

- **Sufficient Statistic:** $n_{ijk}$ is the number of examples whose $i^{th}$ attribute takes the $j^{th}$ value, and are classified to the $k^{th}$ class.
- Total # of $n_{ijk}$'s $= m^l \cdot m$. 
### Drawbacks

- One pass of data for each layer, multiple passes in total. (*stream: only one pass*)
- Once make a decision on a splitting attribute, never reconsider. (*stream: concept drifts*)

### Hoeffding Bound

- Consider a real-valued random variable \( r \) whose range is \( R \). Suppose we have \( n \) independent observations of this variable, and compute their mean \( \text{mean}(r) \). The hoeffding bound states that, with probability \( 1 - \delta \), the true mean of the variable is at least \( \text{mean}(r) - \varepsilon \), where

\[
\varepsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2n}}
\]

### Properties

- The hoeffding bound is independent of the probability distribution generating the observations.
- With high probability, the attribute chosen using \( n \) examples is the same that would be chosen using infinite examples.

### Hoeffding Tree Algorithm

*Inputs:* 
- \( S \) is a sequence of examples,
- \( X \) is a set of categorical attributes,
- \( G \) is a split evaluation function,
- \( \delta \) is one minus the desired probability of choosing the correct attribute at any given node.

*Output:* \( HT \) is a decision tree.
Hoeffding Tree Algorithm, cont’d

Procedure HoeffdingTree(S, X, G, δ)

Let HT be a tree with a single leaf l (the root).

For each class y:
   For each value x_{ij} of each attribute X_i in X:
      Let n_{yi}(l)=0.
      For each example (x, y) in S:
         Sort (x, y) into a leaf l using HT.
         For each x_{ij} in x such that x_{ij} in X_i:
            Increment n_{yi}(l).
            If the examples seen so far at l are not all of the same class, then:
               Compute G(X_i) for each attribute X_i in X_i using n_{yi}(l).
               Let X_{aa} be the attribute with highest G,
               Let X_{bb} be the attribute with second-highest G.
               Compute τ using Hoeffding bound.
               If G(X_{aa}) - G(X_{bb}) > τ, then:
                  Replace l by an internal node that splits on X_{aa}.
                  For each branch of the split:
                     Add a new leaf l_{mm}, and let X_{mm} = X_{aa}.
                     For each class y and each value x_{ij} of each attribute X_i in X_{mm}:
                        Let n_{yi}(l_{mm})=0.

Return HT.

Problems in Practice

- More than one attribute very close to the current best.
- How much time spent on a single example?
- Memory needed with the tree expansion?
- Number of candidate attributes at each node?

VFDT System

- It’s a Very Fast Decision Tree learner, based on the Hoeffding tree algorithm.

Refinements:
- Ties: If G_{X_i} < ε, where ε is a user-specified threshold, split on the current best attribute.
- G computation: Specify an n_{min} that must be accumulated at a leaf before G is recomputed.
- Memory: If the max available memory is reached, VFDT deactivates the least promising leaves (w/ the lowest p_eff) to make room for new ones. Can be reactivated if more promising later.
- Poor attributes: Memory is minimized by dropping early on attributes whose difference from the best attribute’s G becomes greater than ε.

VFDT Analysis

- Memory: O(ldvc)
   - l: the number of leaves in the tree
   - d: the number of attributes
   - v: the max number of values per attribute
   - c: the number of classes

It’s independent of the number of examples seen.

- Drawback: doesn’t take care of the time-changing data streams, because we never update the tree structure ever since we finish building the tree.
Brute Force Algorithm

- A sliding window + the VFDT
  - Reapply VFDT to a moving window of examples every time a new example arrives.
  
  \[ t \rightarrow t+1 \]

- From \( t \rightarrow t+1 \), only \( O(1) \) item in the sliding window changes, but we have to rescan \( O(w) \) items if reapply VFDT.

CVFDT

Concept-adapting Very Fast Decision Tree

Basic Ideas:
- An extension to VFDT.
- Maintains VFDT's speed and accuracy.
- Detects and responds to concept changes in \( O(1) \) per example.
- Stays current while making the most of old data by growing an alternative subtree whenever an old one becomes questionable.
- And replace the old with the new when the new becomes more accurate.

CVFDT Algorithm

Tree node (internal node & leaf node of HT and all alternate trees)
- maintain sufficient statistics \( n_{ijk} \)
- assigned a unique, monotonically increasing ID when created.

Sliding window
- the max ID of the leaves an example reaches is attached with the example in \( W \).

CVFDT Algorithm, cont'd

Observe a new example
- increase the sufficient statistics \( n_{ijk} \) along the way from the root to leaves.
- record the max ID of the leaves it reaches in HT and all alternate trees.

Forget the old
- decrease the sufficient statistics \( n_{ijk} \) of every node the example reaches whose ID \( \leq \) the stored ID.
CVFDT Algorithm, cont’d

- **Growth of alternate subtrees**
  - If \( G(X_a) - G(X_b) \leq \varepsilon \) and \( \varepsilon > \tau \), grow a subtree.
  - Check periodically, say every \( f \) examples.

- **Replacement with alternate subtrees**
  - The next coming \( m \) examples are used to compare the accuracy of the current subtree in HT with the accuracies of all of its alternate subtrees.
  - Replace if the most accurate alternate is more accurate than the current.
  - Prune alternate subtrees that are not making progress.
  - Check periodically.

CVFDT vs. VFDT

<table>
<thead>
<tr>
<th></th>
<th>VFDT</th>
<th>CVFDT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory</td>
<td>( O(lvdvc) )</td>
<td>( O(ldvc) )</td>
</tr>
<tr>
<td>Time</td>
<td>( O(ndvc) )</td>
<td>( O(l,dvc) )</td>
</tr>
</tbody>
</table>

1: the number of leaves in HT
2: the number of nodes in the main tree and all alternate subtrees
3: the number of attributes
4: the max number of values per attribute
5: the number of classes
6: the height of HT
7: the length of the longest path through HT times the number of alternate trees

What if …

- The number of attribute values is huge. e.g. # of IP addresses \( n = 2^{32} \).
- The flavor of sketch-based solution...
  - Hash functions and statistical techniques needed to guarantee accuracy and efficiency.
  - Why not approx \( n_{jk} \)'s? more time efficient!

References

- Mining High-Speed Data Streams, Pedro Domingos and Geoff Hulten, KDD 2000
- Mining Time-Changing Data Streams, Geoff Hulten, Laurie Spencer, and Pedro Domingos, KDD 2001
- A survey on Data Stream Algorithms. S. Muthukrishnan
Thanks!

Q & A ???