Data Mining on Streams

Using Decision Trees

CS 536: Machine Learning
Instructor: Michael Littman

TA: Yihua Wu

Outline

• Introduction to data streams
• Overview of traditional DT learning ALG
• DT learning ALGs on streams
• Open issues of learning on streams
What is Data Mining

• A technique for finding and describing structural patterns in data;
• A tool for helping to explain data and make predictions from it.

E.G.
supermarket chain
stock market analysis

The Data Stream Phenomenon

• Highly detailed, automatic, rapid data feeds.
  – Radar: meteorological observations.
  – Satellite: geodetics, radiation,…
  – Astronomical surveys: optical, IR, radio,…
  – Internet: traffic logs, user queries, email, financial,
  – Sensor nodes: many more “observation points”.
• Need for near-real time analysis of data feeds.
  – Detect outliers, extreme events, fraud, intrusion, anomalous activity, complex correlations, classification,…
  – Monitoring.
Models of Data Streams

- Signal $s[1...n]$. $n$ is universe size.
- Three models:
  - Timeseries model: $s(1), s(2), ..., s(t), ...$
  - Cash Register model: $s(j) = s(j) + a(k)$. $a(k) > 0$. (insert only)
  - Turnstile model: $s(j) = s(j) + u(k)$. (both insert and delete)

Traditional vs. Stream Mining

<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>unlimit</td>
<td>strict</td>
</tr>
<tr>
<td>memory</td>
<td>unlimit</td>
<td>strict</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>
Desiderata

– Per item processing time
– Space stored
– Time for computing functions on $s$
– Must all be $\text{polylog}(n,||s||)$.

• And…
  – A one pass algorithm

Decision Tree

• **Decision tree** is a classification model. Its basic structure is a general tree structure.
  
  - internal node: test on example’s attribute values
  - leaf node: class labels

• **Input:** a set of examples (including both **attribute** and **class** variables)

• **Output:**
  
  - **training data:** a decision tree structure
  - **test data:** the predicted class
Building Decision Trees

- Key idea:
  1) pick an attribute to test at root;
  2) divide the training data into subsets $D_i$ for each value the attribute can take on;
  3) build the tree for each $D_i$ and splice it in under the appropriate branch at the root

- Pick an attribute:
  Find an attribute that divides data into as pure subsets as possible.
**Entropy Value**

- **Entropy value**
  
  Given $P_1, \ldots, P_m$, $0 \leq P_i \leq 1$, $1 \leq i \leq m$; \[
  \sum_{i=1}^{m} P_i = 1
  \]

  Entropy $(P_1, P_2, \ldots, P_m) = -\sum_{i=1}^{m} P_i \log P_i$

  ![Entropy Graph]

  The larger the entropy value, the less pure the data is.

**Gain Value**

- For each class $C_i$ and given data $D$, let $n_i = |D_{c=c_i}|$ (m is the # of classes)
  
  $\text{Data-Info}([n_1, n_2, \ldots, n_m]) = \text{entropy}(n_1/n, n_2/n, \ldots, n_m/n)$,
  where $n = |D|$.

- For feature $f_i$ with values $v_1, \ldots, v_{a_i}$
  
  $\text{Split-Info}(f_i, D) = \sum_{j=1}^{a_i} w_j \text{Data-Info}([D_{f_i=v_j}, c \neq c_i, \ldots, D_{f_i=v_j}, c = c_i])$
  where $w_j = |D_{f_i=v_j}|/|D|$.

- $\text{Gain}(f_i, D) = \text{Data-Info}([n_1, n_2, \ldots, n_m]) - \text{Split-Info}(f_i, D)$

- Pick $f_i$ with the largest $\text{Gain}(f_i, D)$. 
Sufficient Statistics

Split-Info \( f, D \) = \( \sum_{j=1}^{w} w_j \text{Data} - \text{Info}([D_{V, c = c_1}, ..., D_{V, c = c_w}]) \)

\[
= \sum_j \left| \frac{n_{ij}}{n_i} \right| \sum_k \left( -\frac{n_{ijk}}{n_{ij}} \log \frac{n_{ijk}}{n_{ij}} \right)
\]

Sufficient Statistics: \( n_{ijk} \), the number of examples whose \( i \)th attribute has the \( j \)th value, and are classified to the \( k \)th class.

Criteria
Drawbacks

• One pass of data for each internal node, multiple passes in total. (stream: only one pass)

• Once I make the decision on an 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 $\bar{r}$. The hoeffding bound states that, with probability $1-\delta$, the true mean of the variable is at least $\bar{r} - \varepsilon$, 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 (1)

- **Inputs:**
  - \( S \) is a sequence of examples,
  - \( X \) is a set of discrete 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 (2)

Procedure HoeffdingTree(S, X, G, δ)

Let HT be a tree with a single leaf l₁ (the root).

For each class yₖ

For each value xᵢ of each attribute Xᵢ ∈ X

Let nᵢₖ(l₁)=0.

For each example (x, yₖ) in S

Sort (x, y) into a leaf l using HT.

For each xᵢ in x such that Xᵢ ∈ Xᵢ

Increment nᵢₖ(l₁).

If the examples seen so far at l are not all of the same class, then

Compute Gᵢ(Xᵢ) for each attribute Xᵢ ∈ Xᵢ using nᵢₖ(l₁).

Let Xₐ be the attribute with highest Gᵢ.

Let X₉ be the attribute with second-highest Gᵢ.

Compute ε using hoeffding bound.

If Gᵢ(Xₐ) - Gᵢ(X₉) > ε, then

Replace l by an internal node that splits on Xₐ.

For each branch of the split

Add a new leaf lₘ, and let Xₘ = X - {Xₐ}.

For each class yₖ and each value xᵢ of each attribute Xᵢ ∈ Xₘ

Let nᵢₖ(lₘ)=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 the Very Fast Decision Tree learner, based on the hoeffding tree algorithm.
- **Refinements:**
  - **Ties.** If $\Delta G < \varepsilon < \tau$, where $\tau$ is a user-specified threshold, split on the current best attribute.
  - **G computation.** Specify an $n_{\text{min}}$ 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 $p_e$) 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 $\varepsilon$.

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.

- 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 (1)

- Tree nodes (internal nodes & leaf nodes of HT and all alternate trees)
  - maintain sufficient statistics \( n_{ijk} \)
  - assigned a unique, monotonically increasing ID when created.
- Sliding windows
  - the max ID of the leaves an example reaches is attached with the example in W.

CVFDT Algorithm (2)

- 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, whose ID \( \leq \) the stored ID, reaches.
CVFDT Algorithm (3)

• 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(ldvc) )</td>
<td>( O(ndvc) )</td>
</tr>
<tr>
<td>Time</td>
<td>( O(l_dvcw) )</td>
<td>( O(l_dvc) )</td>
</tr>
</tbody>
</table>

- \( l \): the number of leaves in HT
- \( n \): the number of nodes in the main tree and all alternate subtrees
- \( d \): the number of attributes
- \( v \): the max number of values per attribute
- \( c \): the number of classes
- \( l_H \): the height of HT
- \( l_d \): the length of the longest path through HT times the number of alternate trees
Open Issues in DT

• Space and Time complexity
• Other criteria to evaluate DTs?
• More strict bounds other than Hoeffding bound? (size of data to compute trees)
• When concepts change periodically, if some subtrees may become useful again, may we identify these situations and take advantage of them?
• What if coming examples have different weights?

Acknowledgements

• Michael Littman
  – Enjoy the class!
• S. Muthukrishnan
• Pedro Domingos
  – 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
The End

Thank you!