Learning Model Trees From Evolving Data Streams (FIMT-DD)
outline

• Introduction
• Splitting-attribute selection
• Numerical attributes construct a tree
• Computing the linear models in the leaves
• Change detection and adaptation mechanisms
• Conclusion
Introduction

Data Stream

A splitting-attribute method and construct a model tree incrementally

Build a linear model in the leaves

Concept drift

Yes

No

continue......

Adaptation Mechanisms
Split criterion

There many split criterion strategies, in FIMT-DD the splitting criterion was used is SDR (Standard Deviation Reduction) which can be done incrementally.

For example: the dataset $S$ of size $N$, $hA$ in attribute $A$ will split the dataset into two subset $SL$ and $SR$ (i.e. $S = SL U SR$; $N = NL + NR$)

The formula for measuring split of SDR $hA$ is:

$$SDR(hA) = sd(S) - \frac{NL}{N}sd(S_L) - \frac{NR}{N}sd(S_R)$$

$$sd(S) = \sqrt{\frac{1}{N} \left( \sum_{i=1}^{N} (y_i - \bar{y})^2 \right)} = \sqrt{\frac{1}{N} \left( \sum_{i=1}^{N} y_i^2 - \frac{1}{N} \left( \sum_{i=1}^{N} y_i \right)^2 \right)}$$
If $h_A$ is the best split of attribute A and $h_B$ is the second best split of attribute B, further consider the ratio of the SDR values for the best two split ($h_A$ and $h_B$) as a real-valued random variable $r$. That is:

$$r = \frac{\text{SDR}(h_B)}{\text{SDR}(h_A)}$$

$r$ between 0 and 1, in the stream each value $r_1, r_2, \ldots, r_N$. Use the **Hoeffding inequality**: $P\left(\left|\bar{X} - E[\bar{X}]\right| \geq \varepsilon\right) \leq e^{-2N\varepsilon^2}$

confidence: $1 - \delta$  

$N$ random i.i.d in the range $R$ is within distance $\varepsilon$ of the true mean

$$N \geq -\frac{\log(\delta/2)}{2\varepsilon^2}$$
Why

\[ r^+ = r + \varepsilon \quad \text{and} \quad r^- = r - \varepsilon \quad \text{and} \quad r^- \leq r_{\text{true}} \leq r^+ \]

The best observed attribute over a portion of the data is really the best over the whole distribution. Therefore with confidence \(1 - \delta\) the split \( hA \) is deemed as the best one. In this case, the splitting criterion is satisfied and the split \( hA \) can be applied.
Numerical attributes (E-BST)

Selection of the distribution tree is dependent upon the number of points of divisions, traditional methods require initialization before the data is processed. In FIMT-DD, there use E-BST (Extended Binary Search Tree)

E-BST using two arrays of 3 elements:
Fig. 2 Illustrative example of a E-BST constructed from the given table of pair of values (right)
FindBestSplit

1. IF LeftChild exists
2. Call FindBestSplit(LeftChild)
3. Update the sums and counts for computing SDR of the split
4. IF maxSDR < ComputeSDR(current node)
5. Update maxSDR
6. IF RightChild exists
7. Call FindBestSplit(RightChild)
8. Update the sums and counts for computing SDR of the split
9. END
Prun

SDR\( (h_1) \): the greatest reduction in the standard deviation at the last evaluation
SDR\( (h_2) \): the second highest reduction over a different attribute.

\[
r = \frac{SDR(h_2)}{SDR(h_1)}
\]

for each split point with \( \frac{SDR(h_i)}{SDR(h_1)} < r - 2\epsilon \) is considered bad split and it will be pruned

When a node is removed from the tree no information is lost because its sums and counts are held by upper nodes
Linear models in leaves

Given a data stream $<x_i, y_i>$, the perceptron regression:

$$o = w_0 + \sum_{i=1}^{n} w_i x_i$$
The difference is the weights are updated when every data arrival.

\[ \omega_{i+1} = \omega_i + \eta(o - y)x_i \]

\( o \): predict value  \( y \): real value  \( \eta \): learning rate

The initial learning rate \( \eta_0 \) and the learning rate decay parameter \( \eta_d \) should be set to appropriately (e.g., \( \eta_0 = 0.1 \) and \( \eta_d = 0.005 \))

add: Each perceptron learning phase is done in parallel with the addition of nodes
Change detection

The detection mechanism that we propose is on-line and enables local change detection. Monitor the evolution of the error at every region of the instance space.

Update only the affected parts of the model.
PH(Page–Hinckley)

\[ m_t = \sum_{t=1}^{T} (x_t - \bar{x}_t - \alpha) \]

\[ \bar{x}_t = \frac{1}{t} \sum_{l=1}^{t} x_l \]

\[ M_T = \min \{ m_t, t = 1, \ldots, T \} \]

At every moment, the PH test monitors the difference between the minimum \( M_T \) and \( m_T \):

\[ PH_T = m_T - M_T \]

The parameter \( \alpha \) corresponds to the minimal absolute value of the amplitude of the jump to be detected.
When this difference is greater than a given threshold ($\lambda$), the threshold parameter $\lambda$ depends on the admissible false alarm rate. Increasing $\lambda$ entails fewer false alarms, but might miss some changes.

A general guideline, when $\alpha$ has a smaller value $\lambda$ should have a larger value: This is to reduce the number of false alarms.
TD AND BU

Top-Down (TD) method: If the error is computed from the node where the PH test in the node, the error will be passing from node to leaf.

Therefore, the loss will be monitored in the direction from the top towards the “bottom” of the tree.

Bottom-Up (BU) method: If the error is computed using the prediction in the leaf, the example must first reach the leaf. The computed difference at the leaf will be then back-propagated to the root node.

While back-propagating the error of the PH tests located in the internal nodes will monitor the evolution.
Adaption

There are three possible adaptation strategies for model trees:

1. **Further splitting and growing** new sub-trees to the old structure.

2. **Prune** the parts of the tree where concept drift was detected.

3. Build an **alternate sub-tree** for the region where drift is detected.
### Experiment

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>RE%</th>
<th>RRSE%</th>
<th>Leaves</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIRT</td>
<td>0.16</td>
<td>0.19</td>
<td>2452.23</td>
<td>24.75</td>
</tr>
<tr>
<td>CUBIST</td>
<td>0.13</td>
<td>–</td>
<td>37.50</td>
<td>104.79</td>
</tr>
<tr>
<td>FIMT_Const</td>
<td>0.11</td>
<td>0.14</td>
<td>2452.23</td>
<td>27.11</td>
</tr>
<tr>
<td>FIMT_Decay</td>
<td>0.11</td>
<td>0.14</td>
<td>2452.23</td>
<td>26.93</td>
</tr>
<tr>
<td>LR</td>
<td>0.46</td>
<td>0.51</td>
<td>1.00</td>
<td>2468.61</td>
</tr>
<tr>
<td>BatchRD</td>
<td>0.08</td>
<td>0.10</td>
<td>27286.30</td>
<td>5234.85</td>
</tr>
<tr>
<td>BatchRA</td>
<td>0.71</td>
<td>0.69</td>
<td>56.97</td>
<td>2316.03</td>
</tr>
<tr>
<td>OnlineRD</td>
<td>0.10</td>
<td>0.13</td>
<td>6579.50</td>
<td>3099.82</td>
</tr>
<tr>
<td>OnlineRA</td>
<td>0.70</td>
<td>0.68</td>
<td>57.77</td>
<td>2360.56</td>
</tr>
</tbody>
</table>
Local drift dataset: three points of abrupt change in the training dataset, the first one at 1/4 of the examples, the second one at 1/2 of the examples and the third at 3/4 of the examples
AltTree strategy on the local drift dataset

No Detection  TD AltTree  BU AltTree

RRSE

1  38  75  112  149  186  223  260  297  334  371  408  445  482  519  556  593  630  667  704  741  778  815  852  889  926  963
Adaptation evaluation on the local drift dataset

- No Detection
- TD/BU AltTree
- Rebuild
Global drift dataset: There are two points of concept drift, the first of which occurs at 1/2 of the examples and the second at 3/4 of the examples.
Conclusion

- Using tree structure to deal with incremental learning really is a good choice.

- Using another regression method to deal with concept drifting is an urgent job such as GBM (Gradient Boost Machines)
Thanks