Lecture 7. Data Stream Mining. Building decision trees

Ricard Gavaldà

MIRI Seminar on Data Streams, Spring 2015
Contents

1. Data Stream Mining

2. Decision Tree Learning
Data Stream Mining
Finding in what sense data is not random

For example: frequently repeated patterns, correlations among attributes, one attribute being predictable from others, . . .

Fix a description language a priori, and show that data has a description more concise than the data itself
Background: Learning and mining
Mining in Data Streams: What’s new?

- Make one pass on the data
- Use low memory
  - Certainly sublinear in data size
  - In practice, that fits in main memory – no disk accesses
- Use low processing time per item
- Data evolves over time - nonstationary distribution
Two main approaches

- Learner builds model, perhaps batch style
- When change detected, revise or rebuild from scratch
Two approaches

- Keep accurate statistics of recent / relevant data
- e.g. with “intelligent counters”
- Learner keeps model in sync with these statistics
Decision Tree Learning
Background: Decision Trees

- Powerful, intuitive classifiers
- Good induction algorithms since mid 80’s (C4.5, CART)
- Many many algorithms and variants now
Induct(Dataset $S$) returns Tree $T$

if (not expandable($S$)) then
    return Build_Learn($S$)
else
    choose “best” attribute $a \in A$
    split $S$ into $S_1, \ldots S_v$ using $a$
    for $i = 1..v$, $T_i = \text{Induct}(S_i)$
    return Tree(root=$a, T_1, \ldots, T_v$)
Top-down Induction, C4.5 Style

To have a full algorithm one must specify:

- `not_expandable(S)`
- `Build_Leaf(S)`
- notion of “best” attribute
  - usually, maximize some gain function $G(A, S)$
  - information gain, Gini index, . . .
  - relation of $A$ to the class attribute, $C$
- Postprocessing (pruning?)

Time, memory: reasonable polynomial in $|S|$, $|A|$, $|V|$

Extension to continuous attributes: choose attribute and cutpoints

- Very influential paper
- Very Fast induction of Decision Trees, a.k.a. Hoeffding trees
- Algorithm for inducing decision trees in data stream way
- Does not deal with time change
- Does not store examples - memory independent of data size
**Crucial observation [DH00]**

An almost-best attribute can be pinpointed quickly: Evaluate gain function \( G(A, S) \) on examples seen so far \( S \), then use Hoeffding bound.

**Criterion**

If \( A_i \) satisfies

\[
G(A_i, S) > G(A_j, S) + \varepsilon(|S|, \delta) \text{ for every } j \neq i
\]

conclude “\( A_i \) is best” with probability \( 1 - \delta \).
VFDT-like Algorithm

\( T := \text{Leaf with empty statistics}; \)
For \( t = 1, 2, \ldots \) do VFDT\_Grow\((T, x_t)\)

VFDT\_Grow (Tree \( T \), example \( x \))
run \( x \) from the root of \( T \) to a leaf \( L \)
update statistics on attribute values at \( L \) using \( x \)
evaluate \( G(A_i, S_L) \) for all \( i \) from statistics at \( L \)
if there is an \( i \) such that, for all \( j \),
\[
G(A_i, S_L) > G(A_j, S_L) + \varepsilon(S_L, \delta)
\]
then
turn leaf \( L \) to a node labelled with \( A_i \)
create children of \( L \) for all values of \( A_i \)
make each child a leaf with empty statistics
Extensions of VFDT

- **IADEM** [G. Ramos, J. del Campo, R. Morales-Bueno 2006]
  - Better splitting and expanding criteria
  - Margin-driven growth

- **VFDT\_c** [J. Gama, R. Fernandes, R. Rocha 2006],
  - UFFT [J. Gama, P. Medas 2005]
    - Continuous attributes
    - Naive Bayes at inner nodes and leaves
    - Short term memory window for detecting concept drift
    - Converts inner nodes back to leaves, fill them with window data
    - Different splitting and expanding criteria

- **CVFDT** [G. Hulten, L. Spencer, P. Domingos 2001]

- Concept-adapting VFDT
- Update statistics at leaves and inner nodes
- Main idea: when change is detected at a subtree, grow candidate subtree
- Eventually, either current subtree or candidate subtree is dropped
- Classification at leaves based on most frequent class in a window of examples
- Decisions at leaf use a window of recent examples
VFDT:
- No concept drift
- No example memory
- No parameters but $\delta$
- Rigorous performance guarantees

CVFDT:
- Concept drift
- Window of examples
- Several parameters besides $\delta$
- No performance guarantees
Parameters related to time-change [default]:

1. $W$: example window size [100,000]
2. $T_1$: time between checks of splitting attributes [20,000]
3. $T_2$: # examples to decide whether best splitting attribute is another [2,000]
4. $T_3$: time to build alternate trees [10,000]
5. $T_4$: # examples to decide if alternate tree better [1,000]
Enter ADWIN

[Bifet, G. 09] Adaptive Hoeffding Trees

Recall: the ADWIN algorithm
- detects change in the mean of a data stream of numbers
- keeps a window $W$ whose mean approximates current mean
- memory, time $O(\log W)$
Adaptive Hoeffding Trees

- Replace counters at nodes with ADWIN’s (AHT-EST), or
- Add an ADWIN to monitor the error of each subtree (AHT-DET)
- Also for alternate trees
- Drop the example memory window
AHT have no parameters!

- When to start growing alternate trees?
  - When ADWIN says “error rate is increasing”, or
  - When ADWIN for a counter says “attribute statistics are changing”

- How to start growing new tree?
  - Use accurate estimates from ADWIN’s at parent - no window

- When to tell alternate tree is better?
  - Use the estimation of error by ADWIN to decide

- How to answer at leaves?
  - Use accurate estimates from ADWIN’s at leaf - no window
CVFDT’s Memory is dominated by example window, if large

<table>
<thead>
<tr>
<th>CVFDT</th>
<th>AHT-Est</th>
<th>AHT-DET</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAVC + AW</td>
<td>TAVC log W</td>
<td>TAVC + T log W</td>
</tr>
</tbody>
</table>

\[
T = \text{Tree size} \quad A = \# \text{ attributes} \\
V = \text{Values per attribute} \quad W = \text{Size of example window} \\
C = \text{Number of classes}
Figure: Learning curve of SEA concepts using continuous attributes
Adaptive Hoeffding Trees: Summary

- No “magic” parameters. Self-adapts to change
- Always as accurate as CVFDT, and sometimes much better
- Less memory - no example window
- Moderate overhead in time (<50%). Working on it
- Rigorous guarantees possible
Exercise

Exercise 1

- Design a streaming version of the Naive Bayes classifier for stationary streams
- Now use ADWIN or some other change detection / tracking mechanism for making it work on evolving data streams

Recall that the NB classifier uses memory proportional to the product of \#attributes x \#number of values per attribute x \#classes. Expect an additional log factor in the adaptive version. Update time should be small (log, if not constant).