Incrementally Optimized Decision Tree
& Tree Explorer

This summary introduces an extension of decision tree algorithm for Massive Online Analysis (MOA), so called “incrementally optimized decision tree (iOVFDT)”.

Version 1.0

Hang Yang and Simon Fong
Software Engineering
University of Macau
For noisy big data, a new decision tree induction proposes to use a multi-objective function to balance prediction accuracy, tree size and learning speed. New methods of functional tree leaf improve accuracy. Besides, intuitive graph visualizes tree structure dynamically for massive data analysis.

How to extract meaningful information from big data has been a popular open problem. Decision tree, which has a high degree of knowledge interpretation, has been favored in many real world applications. However noisy values commonly exist in high-speed data streams, e.g. real-time online data feeds that are prone to interference. When processing big data, it is hard to implement pre-processing and sampling in full batches. To solve this trade-off, we propose a new decision tree so called incrementally optimized very fast decision tree (iOVFDT). Inheriting the use of Hoeffding bound in VFDT algorithm for node-splitting check, it contains four optional strategies of functional tree leaf, which improve the classifying accuracy. In addition, a multi-objective incremental optimization mechanism investigates a balance amongst accuracy, mode size and learning speed. iOVFDT is extension that can be integrated with the latest version of MOA. Besides, iOVFDT has a high extendibility that is able to combine with most VFDT-extended algorithms.

Incrementally Optimized Decision Tree (iOVFDT)
iOVFDT aims to train a decision tree with minimum error from big data, even if the data contain imperfect quality like noise and bias data. The incremental decision tree algorithm that inherits the use of Hoeffding bound in VFDT. Besides, four types of functional tree leaf are proposed in iOVFDT package, improving classifying accuracy. Suppose \( n_{ijk} \) is the sufficient statistic that reflects the number of attribute \( X_i \) with a value \( x_{ij} \) belonging to class \( y_k \). \( i, j, k \) are the index of attribute \( X \), value of attribute \( X \) and class \( y \) respectively.

Majority Class functional leaf:

\[
\text{arg max } k = \{ n_{i1j} \ldots n_{ijk} \ldots n_{ijk} \}
\]

Naïve Bayes functional leaf:

\[
p_{ijk} = \frac{p(x_{ij} | y_k) p(y_k)}{p(x_{ij})}, \text{ arg max } k = \{ p_{i1j} \ldots p_{ijk} \ldots p_{ijk} \}
\]

Weighted Naïve Bayes functional leaf:

\[
p_{ijk} = \omega_{ijk} \frac{p(x_{ij} | y_k) p(y_k)}{p(x_{ij})} \text{ where } \omega_{ijk} = \frac{n_{ijk}}{\sum_{k=1}^{K} n_{ijk}}
\]

Error-adaptive functional leaf:

\[
\text{arg min } F = \{ Err(F^{MC}, y_k), Err(F^{NB}, y_k), Err(F^{WNB}, y_k) \}
\]

\( F^{MC}, F^{NB} \) and \( F^{WNB} \) require memory proportional to \( O(N \cdot I \cdot J \cdot K) \), where \( N \) is the number of nodes in tree model, \( I \) the number of attributes, \( J \) is the maximum number of values per attribute; \( K \) is the number of classes. \( F^{NB} \) and \( F^{WNB} \) are converted from that of \( F^{MC} \). So we don’t require extra memory for \( F^{MC} \) respectively. When required, it can be converted from \( F^{MC} \).

Implementation Platform
Massive Online Analysis (MOA) is a framework for data stream mining. It includes a collection of machine learning algorithms (classification, regression, and clustering) and tools for evaluation. Related to the WEKA project, MOA is also written in Java, while scaling to more demanding problems (http://moa.cs.waikato.ac.nz/). In classification part, MOA has simulated decision tree algorithms that can be evaluated by built-in measurements. The well-defined experimental platform implements in two modes: graphic interface and command line.
Extension of MOA platform, iOVFDT package supports both GUI and command-line mode. What's more, this package adds new functions of ROC statistics and tree visualization to improve the experimental tasks.

iOVFDT applies a multi-objective optimization model to control the node-splitting. After normalizing those three dimensions, the area of this triangle model is defined as $\Phi(T_R_t)$, where $T_R_t$ is the decision tree structure at timestamp $t$. The range of $\Phi(T_R_t)$ is within a min-max model that ensures the variances of statistics mean and true mean isn’t too big to maintain, where $\text{Min.} \Phi(T_R_t) < \Phi(T_R_t) < \text{Max.} \Phi(T_R_t)$. If $\Phi(T_R_t)$ goes beyond this constraint, the existing model is not suitable to embrace new data that the algorithm should be updated. Therefore, the node-splitting condition is adaptively optimized that: $\Delta H(X_i) > H_B$ or $\Phi(T_R_t) > \text{Max.} \Phi(T_R_t)$ or $\Phi(T_R_t) < \text{Min.} \Phi(T_R_t)$.

iOVFDT Package Built-in MOA

After downloading iOVFDT package (iovfdt.jar) and required MOA packages (moa.jar and sizeofag.jar), GUI can be run by typing the command in console:

```
java -cp iovfdt.jar:moa.jar -javaagent:sizeofag.jar moa.gui.GUI
```

Three new components are included in iovfdt.jar:
- Family of iOVFDT algorithm (four types of optional functional tree leaf)
- Model Evaluation Method (with ROC statistics and tree structure buffer output)
- Decision Tree Visualization (by prefuse.jar open source visualization tool)

**Example 1a: evaluate model by GUI mode**

1. Configure the task as EvaluateModel_ROC;
2. Select iOVFDT as the learning method;
3. Select the training data nursery_train.arff and testing data nursery_test.arff;
4. Select the location to save tree model buffer to IOVFDT_SampleTree.txt;
5. Output the result to IOVFDT_MOA_2012.txt;
6. Press button “RUN”.

**Example 1b: evaluate model by command-line mode**

```
```
Another extension function is decision tree visualization. After saving a decision tree format in TXT file, this package provides an intuitive way to explore the tree. The graphically tree exploring can only be run under GUI mode.

**Example 2: visualize decision tree**

1. Configure the task as VisualizeTree;
2. Select the saved tree buffer `IOVFDT_SampleTree.txt`;
3. *Optionally: show dotty format converting;
4. Press button “RUN”.

**Integration With Other VFDT-extended Algorithms**

In source code part, we write comments for each place of modification based on `HoeffdingTree.java`. Generally, seven-part modifications are proposed in `iOVFDT.java`. In each of them, it includes some new class, variables and functions designed for iOVFDT algorithm. When you want to integrate it to other extension of decision tree that uses Hoeffding bound as node-splitting criteria, just add these seven modifications to appropriate places in source codes. It is very easy.

**Conclusion**

iOVFDT package is an extension stream mining algorithm which is implemented based on MOA platform. The current version supports the latest MOA release 201203. The main package contains iOVFDT with fours types of functional tree leaf, model evaluation with ROC statistics and decision tree visualization.

**Acknowledge**

The research is supported by data stream mining project in University of Macau. We are thanking the massive online analysis (MOA) project in University of Waikato.

---

**Converted Dotty Format**

digraph {node [label="att 8:health"] [label="att 5:housing"] [label="att 7:social"] [label="att 2:has_nurs"] [label="class:case"] [weight=0.43021]

```
converted Dotty format:
digraph {node [label="att 8:health"] [label="att 5:housing"] [label="att 7:social"] [label="att 2:has_nurs"] [label="class:case"] [weight=0.43021]
```

---

**About The Author**

Hang YANG (Henry)  
PhD Candidate  
Software Engineering  
University of Macau  
Website: [http://www.yhSky.cn](http://www.yhSky.cn)  
Email: henry.yh@gmail.com  

- Supervised by Dr. Simon Fong