Reproducing Paper
Extremely Fast Decision Tree

Qi Li [1] **Jialong Wu** [1] **Yishujie Zhao** [1] **Xinhao Xu** [1]

Abstract

The de facto standard for learning decision trees from streaming data is *Hoeffding Tree(HT)*, which can converge to conventional batch tree given enough samples. However, *HT* assume the distribution does not change over time and may stuck into ties problem if there are similar attributes. In the paper *Extremely Fast Decision Tree*, the authors proposed a variant called *Hoeffding Anytime Tree(HATT)* and claimed that it solves the above two problems and has superior performance over *HT*. In this project, we reimplemented both the *Hoeffding Tree* and *Hoeffding Anytime Tree* in the same codebase, reran the experiments on UCI classification data sets and verified that the conclusions from paper *Extremely Fast Decision Tree* are reproducible.

1. Introduction and Related Work

A majority of Machine Learning algorithms are batch data oriented, i.e., they are trained by a certain volume of data at once and then deployed for usage. However, those algorithms could fail in many present-day data mining applications, where the data is continuously generated by different sources, called *streaming data* (Amazon, 2019). The stream data features in real-time and high-volume. For instance, every day retail chains record millions of calls, large banks process millions of ATM and credit card operations, and popular Web sites log millions of hits. As the expansion of Internet continues, we can expect that such kind of data will be the rule rather than the exception. In such scenarios, we would like to have stream processing techniques that can operate continuously and indefinitely without access to all of the data.

The de facto standard for learning decision trees from streaming data is Hoeffding Tree (HT) (Domingos & Hulten, 2000), which has been used as a base for many state-of-the-art drift learners. (d. Barros et al., 2016; Bifet et al., 2009; Hoeglinger & Pears, 2007; Hulten et al., 2001). The Hoeffding trees can be learned in constant time per example and is shown to be identical to the trees learned by conventional batch learner, given enough examples. It doesn't need to store any offline examples in main memory and is an anytime algorithm in the sense that the model is available at any time after some warm up training.

However, despite the superior of Hoeffding Tree, it has two disadvantages. Firstly, it doesn't consider concept drift (Gama et al., 2014), i.e., the relation between the input data and the target variable changes over time, which is a common phenomenon in real time applications. Since Hoeffding tree never revisit the split branches, there is no recourse to alter the tree when data distribution changes. Secondly, the Hoeffding tree uses Hoeffding bound to guarantee that one attribute is statistically better than other attributes. However, there are cases where attributes could be very similar and potentially many examples would be required to distinguish them with high confidence, leading to *ties* problem.

To ease both two issues mentioned above, Hoeffding Anytime Tree (Manapragada et al., 2018) is proposed with minor modifications to the Hoeffding Tree. Hoeffding Anytime Tree is equivalent to Hoeffding tree except that it uses Hoeffding bound to determine whether the merit of splitting on the best attribute exceeds that of not having a split, or that of current split attribute. By those adjustment, Hoeffding Anytime Tree is naturally resilient to concept drift and ties, and will still converge to batch tree asymptotically.

In this project, we reimplement the Hoeffding Tree and Hoeffding Anytime Tree in the same codebase, reran the experiments on UCI classification data sets and verified the conclusions of (Manapragada et al., 2018).

2. Problem and Methods

2.1. Problem Description

Online Machine Learning. As opposed to batch learning techniques where the predictor are learned on the entire training data set at once. In this project we focus on online machine learning (Hazan, 2016), in which the data becomes available in a sequential order and is used to update the

[1]School of Software, Tsinghua University. Correspondence to: Qi Li <liqi17@mails.tsinghua.edu.cn>.

predictor at each step. Specifically, in our experiment the data is fed one by one into predictor.

Concept Drift. A common property of streaming data is *Concept Drift*, i.e., the distribution of data changes over time in unforeseen ways. The model could be less effective over time if no adaption is made for this phenomenon. In some datasets, the data is sorted by the target variable, so the concept drift occurs naturally. However, we can easily get i.i.d. data by simply shuffling them before training. We evaluate both two scenarios in our experiments. We also generate simulated datasets with concept drift. See Section 4.1 for details.

Prequential Accuracy. To evaluate the performance of online learning, we use the prequential (abbv. for predictive sequential) accuracy. Specifically, for each incoming instance, we firstly test it using our model, and then use it as training sample. A window of size N is set to compute accumulative accuracy, i.e., the last N samples are considered to get accuracy at each timestamp.

2.2. Method Description

Hoeffding Bound. The core statistical result used by both Hoeffding Tree and Hoeffding Anytime Tree is Hoeffding Bound. Consider a real-valued random variable r whose range is R. Suppose we made n independent observations of variable and computer their mean $\bar{r}$. The Hoeffding bound states that, with probability $1-\delta$, the true mean of the variable is at least $\bar{r}-\epsilon$, where

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

The most important property of Hoeffding bound is that it is *independent* of the probability distribution of data.

Hoeffding Tree. Also referred to as *Very Fast Decision Tree(VFDT)*, Hoeffding Tree solve the hard problem of deciding when to split using *Hoeffding bound*. The target is to make sure the split attribute chosen using n samples would be the same as that chosen using infinite samples with high probability. Concretely, suppose G is to be maximized and X_a and X_b are the attributes with best and second-best $\bar{G}$ after seeing n samples, Hoeffding Tree decide to split when $\bar{G}(X_a) - \bar{G}(X_b) \geq \varepsilon$, which guarantees that X_a would be the best attribute with probability $1-\delta$. Under realistic assumptions Hoeffding Tree would be asymptotically arbitrarily close to the ones produced by a batch learner, for proof please refer to (Domingos & Hulten, 2000).

To keep track of overall statistics without storing past data. Hoeffding Tree use counts n_{ijk} to compute heuristic measures, which means the count of class y_k given $X_i = j$. The pseudo code of Hoeffding Tree algorithm is shown in Algorithm 1 according to (Domingos & Hulten, 2000). The pseudo code only handle discrete attributes for simplicity, and the extension to numeric ones is straight-forward, by tests of the form $X_i \leq x_{ij}$.

Algorithm 1 The Hoeffding Tree algorithm

Input: S is a sequence of examples,
$\mathbf{X}$ is a set of discrete attributes,
$G(.)$ is a split evaluation function,
δ is one minus the desired probability of choosing the correct attribute at any given node.
Output: HT is a decision tree
Procedure HoeffdingTree (S, X, G, δ)
Let HT be a tree with a signle leaf l_1 (the root.)
Let $X_1 = X \cup \{X_\emptyset\}$
Let $\bar{G}_1(X_\emptyset)$ be the $\bar{G}$ obtained by predicting the most frequent class in S.
for each class in y_k **do**
  for each value x_{ij} of each attribute $X_i \in \mathbf{X}$ **do**
    Let $n_{ijk}(l_1) = 0$.
for each example $(\mathbf{x}, y_k)$ in S **do**
  Sort $(\mathbf{x}, y_k)$ into a leaf l using HT.
  for each x_{ij} in $\mathbf{x}$ such that X_i in $\mathbf{X}_l$ **do**
    Increment $n_{ijk}(l)$.
  Label l with the majority class among the examples seen so far at l.
  if the examples seen so far at l are not all of the same class, **then**
    Compute $\bar{G}_l(\mathbf{X}_i)$ For each attribute $X_i \in \mathbf{X}_l - \{X_\emptyset\}$ using the counts $n_{ijk}(l)$.
    Let X_a be the attribute with highest $\bar{G}_l$.
    Let X_b be the attribute with second-highest $\bar{G}_l$.
    Compute ϵ using Equation 1.
    if $\bar{G}_l(X_a) - \bar{G}_l(X_b) > \epsilon$ and $X_a \neq X_\emptyset$ **then**
      Replace l by an internal node that splits on X_a.
      for each branch of the split **do**
        Add a new leaf l_m, and let $\mathbf{X_m} = \mathbf{X} - \{X_a\}$
        Let $\bar{G}_m(\mathbf{X}_\emptyset)$ be the $\bar{G}$ obtained by predicting the most frequent class at l_m.
        for each class y_k and each value x_{ij} of each attribute $X_i \in \mathbf{X_m} - \{X_\emptyset\}$ **do**
          Let $n_{ijk}(l_m) = 0$.

Hoeffding Anytime Tree. Also referred to as *Extremely Fast Decision Tree(EFDT)*, Hoeffding Anytime Tree is a variant of Hoeffding Tree, which seeks to select and deploy a split as soon as it is confident the split is useful, and revisits that decision, substituting it if it becomes evident that a better split is available. The pesudo code is shown in Algorithm 2 according to (Manapragada et al., 2018).

Algorithm 2 The Hoeffding Anytime Tree algorithm

Input: S, a sequence of examples. At time t, the observed sequence is $S^t = ((\overrightarrow{x}_1, y_1), (\overrightarrow{x}_2, y_2), \cdots, (\overrightarrow{x}_t, y_t))$
$\mathbf{X} = \{X_1, X_2, ..., X_m\}$, a set of m attributes.
δ, the acceptable probability of choosing the wrong split attribute at a given node.
$G(.)$, a split evaluation function.
Output: $HATT^t$, the model at time t constructed from having observed sequence S^t.
Procedure HoeffdingAnyTimeTree (S, X, G, δ)
Let HATT be a tree with a signle leaf, the root.
Let $X_1 = X \cup X_\emptyset$
Let $G_1(X_\emptyset)$ be the G obtained by predicting the most frequent class in S.
for each class in y_k **do**
  for each value x_{ij} of each attribute $X_i \in X$ **do**
    Set counter $n_{ijk}(root) = 0$
for each example $(\overrightarrow{x}, y)$ in S **do**
  Sort $(\overrightarrow{x}, y)$ into a leaf using HATT
  for each node in path $(root...l)$ **do**
    for each x_{ij} in $\overrightarrow{x}$ such that $X_i \in X_{node}$ **do**
      Increment $n_{ijk}(node)$
      if node = l **then**
        AttemptToSplit(l)
      else
        ReEvaluateBestSplit($node$)

Algorithm 3 AttemptToSplit(leafNode l)

Label l with the majority class at l.
if all examples at l are not of the same class **then**
  Compute $\bar{G}_l(X_i)$ for each attribute $X_l - \{X_\emptyset\}$ using the counts $n_{ijk}(l)$
  Let X_a be the attribute with the highest $\bar{G}_l$
  Let $X_b = X_\emptyset$
  Compute ε using equation (1).
  if $\bar{G}_l(X_a) - \bar{G}_l(X_b) > \varepsilon$ and $X_a \neq X_\emptyset$ **then**
    Replace l by an internal node that splits on X_a
    for each branch of the split **do**
      Add a new leaf l_m and let $\mathbf{X}_m = \mathbf{X} - X_a$
      Let $\bar{G}_m(X_\emptyset)$ be the G obtained by predicting the most frequent class at l_m
      for each class y_k and each value x_{ij} of each attribute $X_i \in X_m - \{X_\emptyset\}$ **do**
        Let $n_{ijk}(l_m) = 0$.

Algorithm 4 ReEvaluateBestSplit(internalNode int)

Compute $\bar{G}_{int}(X_i)$ for each attribute $\mathbf{X}_{int} - \{\mathbf{X}_\emptyset\}$ using the counts $n_{ijk}(int)$
Let X_a be the attribute with the highest $\bar{G}_{int}$
Let $X_{current}$ be the *current* split attribute
Compute ε using equation (1).
if $\bar{G}_l(X_a) - \bar{G}_l(X_{current}) > \varepsilon$ **then**
  if $X_a = X_\emptyset$ **then**
    Replace internal node *int* with a leaf (kills subtree)
  else
    Replace *int* with an internal node that splits on X_a
    for each branch of the split **do**
      Add a new leaf l_m and let $\mathbf{X}_m = \mathbf{X} - X_a$
      Let $\bar{G}_m(X_\emptyset)$ be the G obtained by predicting the most frequent class at l_m.
      for each class y_k and each value x_{ij} of each attribute $\mathbf{X}_i \in \mathbf{X}_m - \{X_\emptyset\}$ **do**
        Let $n_{ijk}(l_m) = 0$

3. Implementation

3.1. Dependency

We use ***Python*** for reimplementation. The main package dependencies required for algorithm implementation are ***Numpy*** and ***Pandas***.

3.2. Implementation Details

Histogram Observer. We reimplement the Hoeffding Tree and Hoeffding Anytime Tree primarily based on Algorithm 1 and Algorithm 2. The pesudo codes only handle the categorical attributes. For numeric attributes, we didn't follow the common convention of tests of form $X_i < x_{ij}$ and compute $\bar{G}$ for each. Instead, we simulate the categorical attributes by building histograms of numeric attributes. Specifically, we divide numeric range into intervals according to seen samples. (e.g., intervals may be like $(-\infty, -10), [-10, 0), [0, 10), [10, +\infty)$), and then numeric value can be seen as if they were categorical.

Ties Threshold. The δ in both Hoeffding Tree and Hoeffding Anytime Tree is 10^{-7} in our experiments if not specified. We alse use ties breaking trick as mentioned in (Domingos & Hulten, 2000) and $\tau = 0.05$. For full details and instructions to reproduce please refer to `https://github.com/liqi17thu/incremental_decision_tree`.

4. Experiments

4.1. Dataset Description

Gas. (Huerta et al., 2016) Gas sensors for home activity monitoring Data Set. 10 real attributes are provided: 8 MOX gas sensors, and a temperature and humidity sensor. The task is 3-way classification: banana, wine, background.

Poker. (Lichman, 2013) Poker Hand Data Set. A poker hand with five playing cards. 10 categorical attributes are provided. The task is 10-way classification.

Skin. (Bhatt & Dhall, 2012) Skin Segmentation Data Set. 3 real attributes are provided: B,G,R values from face images. The task is binary classification: skin/non-skin samples.

Forest. (Blackard & Dean, 1999) Covertype Data Set. 10 quantitative attributes and 44 qualitative attributes are provided. The task is 7-way classification(7 types of forest cover).

Activity Prediction. (Kwapisz et al., 2011) WISDM's activity prediction DataSet. 3 real attributes are provided: Acceleration in three directions, x, y and z. The task is 6-way classification: Walking, Jogging, Sitting, Standing, Upstairs, Downstairs.

Activity Recognition. (Stisen et al., 2015) Heterogeneity Dataset for Human Activity Recognition. 5 real attributes are provided: The values provided by the sensor for the three axes, x,y,z, the type of sensors and the model the sample originates from. The task is 6-way classification: Biking, Sitting, Standing, Walking, Stair Up and Stair down.

MOA dataset. To evaluate how VFDT and EFDT are affected by the complexity of the learning task and *Concept Drift*, we generate synthetic datasets using MOA RandomTreeGenerator (Bifet et al., 2010). The tree generator can generate a stream based on a randomly generated tree, which are specified in our experiments to have 5 nominal attributes, 5 values per attribute and 2, 3, 4 or 5 classes (refered as *moa2*, *moa3*, *moa4* and *moa5* respectively). To model a concept drift, we naively make a change to a different hidden tree at middle point in the stream.

4.2. Experiment Results

The UCI classification results are shown from Fig 1 to Fig 6. EFDT indeed attains higher prequential accuracy and converges faster on most of the streams, whether shuffled or unshuffled. Especially, EFDT performs significantly better than VFDT in Forest dataset (Fig 2), Poker dataset (Fig 3) and Gas dataset (Fig 4), which is also aligned with statement in (Manapragada et al., 2018). In most cases EFDT run longer than VFDT, but it rarely requires more than double of the time, and sometimes even quicker (see unshuffled Skin dataset cases in Fig 1.)

However, in the simulated dataset MOA, we find out that *EFDT* adjust slower than *VFDT* when an abrupt change of the distribution occurs, which doesn't align with the conclusion that *EFDT* is more resilient to concept drift. We conjecture that it is because the *EFDT* make splits too aggressively, while the differences in information gain between top attributes as well as the information gains themselves are low, making a lot of samples are needed for *EFDT* to readjust.

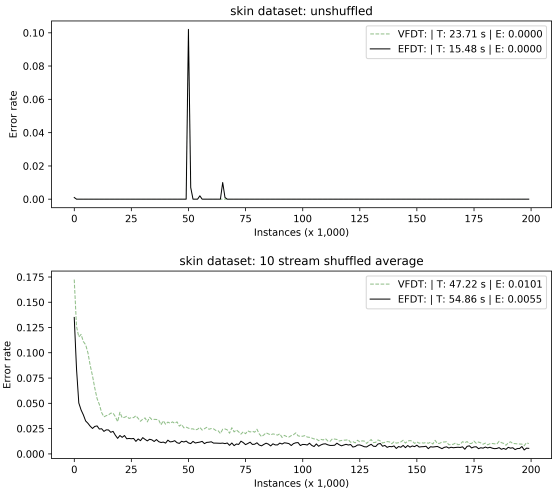

Figure 1. Results on Skin dataset. The legend includes CPU time (T) and the error rate over last few samples of the stream (E).

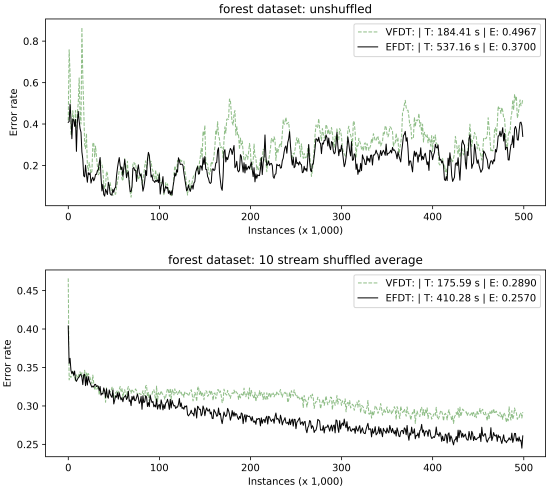

Figure 2. Results on Forest dataset

5. Conclusion

In this project, we reimplemented both the *HT* and *HATT* in the same codebase and reran the experiments on UCI classification data sets. In most of the data streams *HATT* indeed obtains notably better performance than *HT* with tolerably extra time costs, which verify the conclusion of (Manapragada et al., 2018). However, in the simulated dataset MOA, we instead find *HATT* is more vulnerable to concept drift, which doesn't align with our expectation.

Acknowledgements

We would like to thank Prof. Long and TAs for their devotion to ML Course.

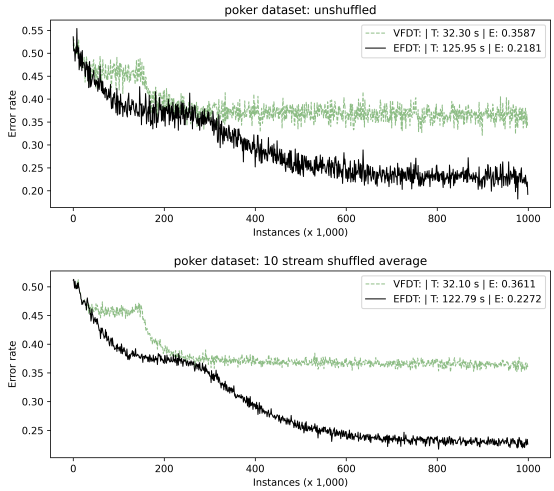

Figure 3. Results on Poker dataset

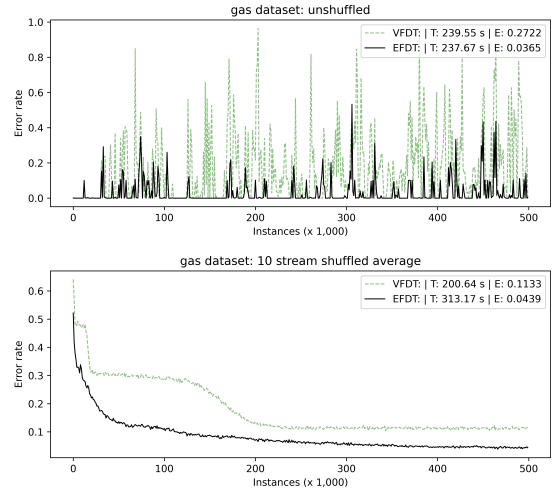

Figure 4. Results on Gas dataset

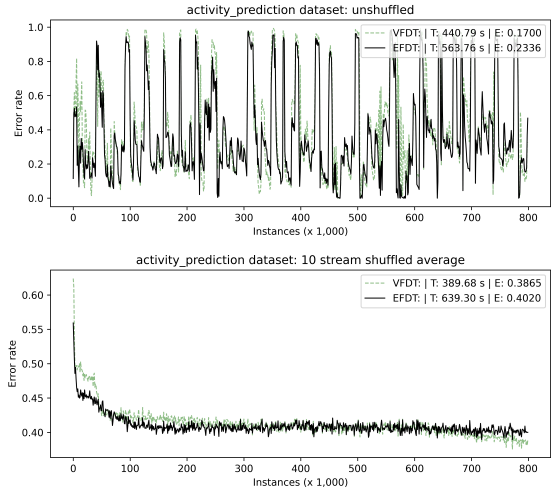

Figure 5. Results on Activity Prediction dataset

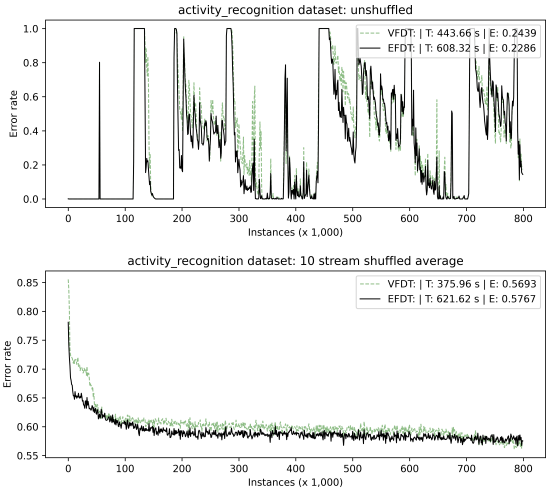

Figure 6. Results on Activity Recognition dataset

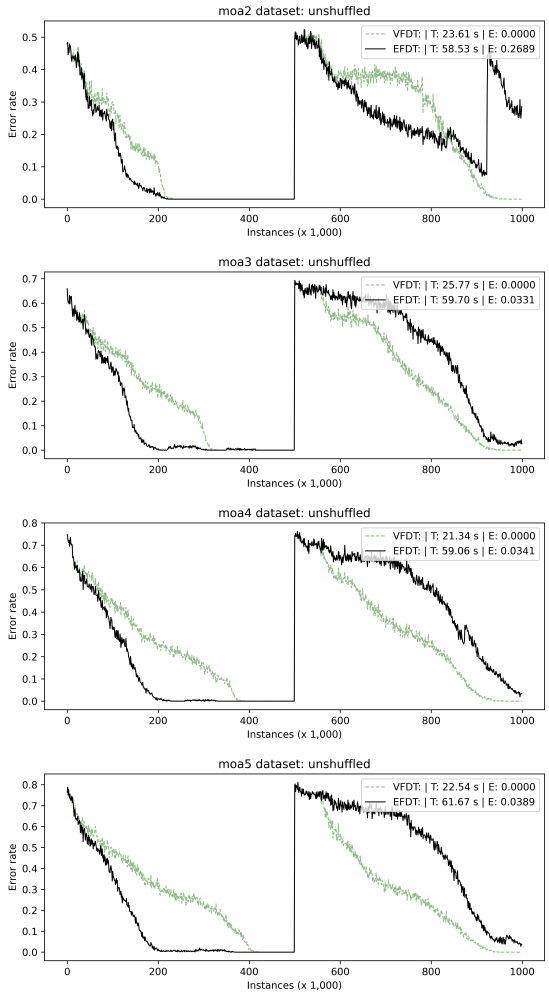

Figure 7. Results on MOA dataset

References

Amazon. What is streaming data? Technical report, Amazon Kinesis, New Brunswick, MA, 2019.

Bhatt, R. and Dhall, A. Skin segmentation dataset: Uci machine learning repository, 2012.

Bifet, A., Holmes, G., Pfahringer, B., Kirkby, R., and Gavaldà, R. New ensemble methods for evolving data streams. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pp. 139–148, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605584959. doi: 10.1145/1557019.1557041. URL https://doi.org/10.1145/1557019.1557041.

Bifet, A., Holmes, G., Pfahringer, B., Kranen, P., Kremer, H., Jansen, T., and Seidl, T. Moa: Massive online analysis, a framework for stream classification and clustering. In *Proceedings of the First Workshop on Applications of Pattern Analysis*, pp. 44–50. PMLR, 2010.

Blackard, J. A. and Dean, D. J. Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. *Computers and electronics in agriculture*, 24(3):131–151, 1999.

d. Barros, R. S. M., Garrido T. de Carvalho Santos, S., and Gonçalves Júnior, P. M. A boosting-like online learning ensemble. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pp. 1871–1878, 2016. doi: 10.1109/IJCNN.2016.7727427.

Domingos, P. and Hulten, G. Mining high-speed data streams. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '00, pp. 71–80, New York, NY, USA, 2000. Association for Computing Machinery. ISBN 1581132336. doi: 10.1145/347090.347107. URL https://doi.org/10.1145/347090.347107.

Gama, J. a., Žliobaitundefined, I., Bifet, A., Pechenizkiy, M., and Bouchachia, A. A survey on concept drift adaptation. *ACM Comput. Surv.*, 46(4), March 2014. ISSN 0360-0300. doi: 10.1145/2523813. URL https://doi.org/10.1145/2523813.

Hazan, E. Introduction to online convex optimization. *Found. Trends Optim.*, 2(3–4):157–325, August 2016. ISSN 2167-3888. doi: 10.1561/2400000013. URL https://doi.org/10.1561/2400000013.

Hoeglinger, S. and Pears, R. Use of hoeffding trees in concept based data stream mining. In *2007 Third International Conference on Information and Automation for Sustainability*, pp. 57–62, 2007. doi: 10.1109/ICIAFS.2007.4544780.

Huerta, R., Mosqueiro, T., Fonollosa, J., Rulkov, N. F., and Rodriguez-Lujan, I. Online decorrelation of humidity and temperature in chemical sensors for continuous monitoring. *Chemometrics and Intelligent Laboratory Systems*, 157:169–176, 2016.

Hulten, G., Spencer, L., and Domingos, P. Mining time-changing data streams. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, pp. 97–106, New York, NY, USA, 2001. Association for Computing Machinery. ISBN 158113391X. doi: 10.1145/502512.502529. URL https://doi.org/10.1145/502512.502529.

Kwapisz, J. R., Weiss, G. M., and Moore, S. A. Activity recognition using cell phone accelerometers. *ACM SigKDD Explorations Newsletter*, 12(2):74–82, 2011.

Lichman, M. UCI machine learning repository. http://archive.ics.uci.edu/ml, 2013.

Manapragada, C., Webb, G. I., and Salehi, M. Extremely fast decision tree. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining*, KDD '18, pp. 1953–1962, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450355520. doi: 10.1145/3219819.3220005. URL https://doi.org/10.1145/3219819.3220005.

Stisen, A., Blunck, H., Bhattacharya, S., Prentow, T. S., Kjærgaard, M. B., Dey, A., Sonne, T., and Jensen, M. M. Smart devices are different: Assessing and mitigatingmobile sensing heterogeneities for activity recognition. In *Proceedings of the 13th ACM conference on embedded networked sensor systems*, pp. 127–140, 2015.

A. Contribution

Qi Li: VFDT implementation, code optimization and report writing.
Jialong Wu: EFDT implementation, evaluation implementation and code optimization.
Yishujie Zhao: Preparing and experiment on datasets (MOA, etc.).
Xinhao Xu: Preparing and experiment on datasets (WISDM, activity recognition, etc.).