MACHINE LEARNING FOR DATA STREAMS

DECISION TREES

Note: A number between parenthesis corresponds to a certain survey:

1. **Mining High-Speed Data Streams (2000)_VFDT_NOT READ YET: No aparece en el survey (6)**
   - (1) Pages 3,4
     - This method essentially subsamples the data in order to achieve scalability in the construction of the decision tree. The idea is to show that the **entire decision tree constructed would be the same as the one built on sub-sampled data with high probability**.
     - The idea is to determine a random sample of sufficient size so that the tree constructed on the sample is the same as that constructed on the entire data set. The **Hoeffding bound** is used to show that the decision tree on the sub-sampled tree would make the same split as on the full stream with high probability. This approach can be used with a variety of criteria such as the gini-index, or information gain.
     - The number of examples required to produce the same split as the original data (with high probability) is determined. The Hoeffding bound is used to determine the number of relevant examples, so that this probabilistic guarantee may be achieved. If all splits in the decision tree are the same, then the same decisión tree will be created.
     - The Hoeffding tree can also be **applied to data streams**, by building the tree incrementally, as more examples stream in, from the higher levels to the lower levels. At any given node, one needs to wait until **enough tuples are available in order to make decisions about lower levels**. The memory requirements are modest, because only the counts of the different discrete values of the attributes (over different classes) need to be maintained in order to make Split decisions.
     - The **VFDT algorithm** is also based on the Hoeffding tree algorithm, though it makes a number of modifications. Specifically, it is **more aggressive about making choices in the tie breaking** of different attributes for splits. It also allows the deactivation of less promising leaf nodes. It is generally more memory efficient, because of these optimizations.
     - The original VFDT method is not designed for cases where the stream is evolving over time.

- http://www.otnira.com/2013/03/28/hoeffding-tree-for-streaming-classification/
    - Hoeffding bound gives certain level of confidence on the best attribute to split the tree, hence we can build the model based on certain number of instances that we have seen.
    - Hoeffding-tree, which is a new decision-tree learning method for streaming that solves these following challenges:
        - Uncertainty in learning time. Learning in Hoeffding tree is **constant time per example** (instance) and this means Hoeffding tree is suitable for mining data streaming.
        - The resulting trees are nearly identical with trees built by conventional batch learner, given enough example to train the and build the Hoeffding tres
    - To achieve the streaming classification characteristics, the authors introduce Hoeffding bound to decide **how many examples of instances needed to achieve certain level of confidence** (i.e. the chosen instance attribute using the bound is the close to the attribute chosen when infinite examples are presented into the classifier).
    - What makes Hoeffding bound attractive is its ability to give the same results **regardless the probability distribution generating the observations**. However, the number of observations needed to reach certain values of \delta and \epsilon are different across probability distributions.
    - With probability 1-\delta , one attribute is superior compared to others when observed difference of information gain is greater than \epsilon.
    - The authors implemented the Hoeffding tree algorithm into Very Fast Decision Tree learner (VFDT) which includes some enhancements for practical use, such as node-limiting strategy, introduction of as tie breaking parameter, grace period of bound calculation, poor attributes removal, fast initialization by using conventional RAM-based learner and ability to rescan previously-seen examples when data rate is slow.
- HT cautiously works toward the asymptotic batch tree, ignoring, and thus not benefiting from potential improvements on the current state of the tree, until it is sufficiently confident that they will not need to be subsequently revised.

RESUMEN DEL PAPER “Strict Very Fast Decision Tree: a memory conservative algorithm for data stream mining (2018)”: Among them, the Very Fast Decision Tree (VFDT) algorithm (Domingos and Hulten, 2000) is one of the most well-known for stream classification, being capable of constructing a decision tree in an online fashion by taking advantage of a statistical property called Hoeffding Bound (HB). By doing so, the VFDT obtains a predictive performance simila to conventional decision tree induction algorithms applied to static datasets. Although VFDT is somewhat memory-friendly, learning from data streams can lead to unnecessary tree growth, increasing memory usage and even compromising its application on memory-scarce scenarios.

Explains the VFDT proposal in an aceptable way.

2. **Mining Time-Changing Data Streams (2001)_CVFDT_NOT READ YET: No aparece en los surveys (5), (6) y (7)**
    - (1) Pages 4,5
        - The original VFDT method is not designed for cases where the stream is evolving over time. The work in [47] extends this method to the case of concept-drifting data streams. This method is referred to as **CVFDT**. CVFDT incorporates two main ideas in order to address the additional challenges of drift:
            - A sliding window of training items is used to limit the impact of historical behavior.
            - Alternate subtrees at each internal node i are constructed.
        - Because of the sliding window approach, the main issue here is the update of the attribute frequency statistics at the nodes, as the sliding window moves forward. For the incoming items, their statistics are added to the attribute frequencies in the current window, and the statistics of the expiring items at the other end of the window are decreased. Therefore, **when these statistics are updated, some nodes may no longer meet the Hoeffding bound, and somehow need to be replaced**.

RESUMEN PAPER “Learning Decision Trees from Data Streams with Concept Drift (2016)”: CVFDT is able to adapt to concept-drift in streams, first growing alternate subtrees at each node of the decision tree, and then replacing the current subtree with the alternate whenever the latter becomes more accurate. This is achieved by maintaining sufficient statistics on a time-window moving over the data stream.

RESUMEN DEL PAPER “Extremely Fast Decision Tree (2018)”: Hulten et al [18] follow up on the Hoeffding Tree work with a procedure for drift adaptation (Concept-adapting Very Fast Decision Tree, CVFDT). CVFDT has a moving window that diminishes statistics recorded at a node due to an example that has fallen out of a window at a given time step. The example statistics at each internal node change as the window moves, and existing splits are replaced if the split attribute is no longer the winning attribute and one of a set of alternate subtrees grown by splitting on winning attributes registers greater accuracy.

3. **Efficient Decision Tree Construction on Streaming Data (2003)_NOT READ YET: No aparece en los surveys (2), (3), (4), (5), (6) y (7)**
    - (1) Page 4
        - One of the major challenges of the VFDT family of methods is that it is naturally suited to categorical data. This is a natural derivative of the fact that decision tree splits implicitly asume categorical

data. Furthermore, discretization of numerical to categorical data is often done offline in order to ensure good distribution of records into the discretized intervals. Of course, it is always possible to test all possible split points, while dealing with numerical data, but the number of posible split points may be very large, when the data is numerical. The bounds used for ensuring that the split is the same for the sampled and the original data may also not apply in this case. The technique in [48] uses **numerical interval pruning in order to reduce the number of possible split points**, and thereby make the approach more effective.

- Furthermore, the work uses the properties of the gain function on the entropy in order to achieve the **same bound as the VFDT method** with the use of a smaller number of samples.

RESUMEN DEL PAPER "Extremely Fast Decision Tree (2018)": There is a sizable literature that adapts HT in sometimes substantial ways [12, 19, 23] that do not, to the best of our knowledge, lead to the same fundamental change in learning premise as does HATT. Substitutes the Hoeffding Test with the "Normal" test.

4. Accurate Decision Trees for Mining High-speed Data Streams (2003)_VDTc_NOT READ YET

RESUMEN DEL PAPER "Learning Decision Trees from Data Streams with Concept Drift (2016)": Another extension of VFDT is VFDTc, which is able to deal with numerical attributes, i.e. not only categorical ones [12]. The algorithm in each leaf stores counters for numerical values. Additionally, to improve performance authors proposed to add a local model in the leaves (i.e. naïve Bayes).

RESUMEN DEL PAPER "Extremely Fast Decision Tree (2018)": Adds support for Naive Bayes at leaves.

RESUMEN DEL PAPER "Strict Very Fast Decision Tree: a memory conservative algorithm for data stream mining (2018)": Later, to increase VFDT predictive performance, instead of using a traditional most common (MC) prediction at the leaves, a Naive Bayes (NB) or Adaptive Naive Bayes (ANB) algorithm can be employed (Gama et al., 2003).

5. Handling numeric attributes in hoeffding trees (2008)_CHAPTER

- Not in the proposals.
- Referenced in the paper "Strict Very Fast Decision Tree: a memory conservative algorithm for data stream mining (2018)": The first version of the VFDT only handled nominal features. Afterwards, many estimators for continuous features were proposed. Pfahringer et al. (2008) reviewed these estimators and observed that the **Gaussian estimator** is the least sensitive to hyperparameter value and induced the most accurate models,

- becoming the default estimator in recent works.

6. **Adaptive Learning from Evolving Data Streams (2009)_HWT_HAT_NOT READ YET.**

   RESUMEN Y COMPARACIÓN DEL PAPER "Learning Decision Trees from Data Streams with Concept Drift (2016)": Since development of the Hoeffding bounds, a number of modifications have been proposed. Bifet and Gavalda proposed the Hoeffding Window Tree (HWT) and the Hoeffding Adaptive Tree (HAT). HWT differs from CVFDT since it creates subtrees without waiting for a fixed number of instances (faster reaction for drift) and updates a subtree as soon as there is a benefit from building the new one., instead of using fixed-size sliding window to detect changes, HAT employs an adaptive window at each internal node.

   RESUMEN DEL PAPER "Extremely Fast Decision Tree (2018)": This method builds a tree that grows alternate subtrees if a subtree is observed to have poorer prequential accuracy on more recent examples, and substitutes an alternate when it has better accuracy than the original subtree. HAT uses an error estimator, such as ADWIN [5] at each node to determine whether the prediction error due to a recent sequence of examples is significantly greater than the prediction error from a longer historical sequence so it can respond to drift.

7. **Fast Perceptron Decision Tree Learning from Evolving Data Streams (2010)_NOT READ YET:** **No aparece en los surveys (2), (4), (5), (6) y (7)**
   - (1) Page 5
     - Bifet et al [19] proposed a method, that shares similarities with the work in [40]. However, the work in [19] replaces the naive Bayes with perceptron classifiers. The idea is to gain greater efficiency, while maintaining competitive accuracy.

   RESUMEN DEL PAPER "Cost-Sensitive Perceptron Decision Trees for Imbalanced Drifting Data Streams (2017)": We propose to build our learning algorithm for imbalanced and drifting data streams on top of the Fast Perceptron Decision Tree [1], as it provides both high accuracy and update speed, making it highly suitable for the task at hand. Its main advantage lies in using a linear perceptron at each leaf. This allows to speed-up the decision making process, as well as improve the overall accuracy. This hybrid solution combines the advantages of trees and neural models, allowing for efficient processing of data streams.

   Original implementation of Fast Perceptron Decision Tree used Hoeffding inequality to determine the amount of instances needed for conducting a split [1]. However, recent study discussed flaws in the Hoeffding bound (paper "Decision Trees for Mining Data Streams Based on the McDiarmid's Bound (2013)".

8. **Decision Trees for Mining Data Streams Based on the McDiarmid's Bound (2013)_NOT READ YET:**

RESUMEN DEL PAPER “Learning Decision Trees from Data Streams with Concept Drift (2016)”: There are also other bounds – like the McDiarmid. In [21] authors proved that the Hoeffding’s inequality is not suitable for soving the underlying problem.

RESUMEN DEL PAPER “Extremely Fast Decision Tree (2018)”: There is a sizable literature that adapts HT in sometimes substantial ways [12, 19, 23] that do not, to the best of our knowledge, lead to the same fundamental change in learning premise as does HATT. Substitutes the Hoeffding Test with McDiarmid’s test.

9. **Hellinger Distance Trees for Imbalanced Streams (2014)_NOT READ YET:**

   RESUMEN DEL PAPER “Cost-Sensitive Perceptron Decision Trees for Imbalanced Drifting Data Streams (2017)” : Combination of Hoeffding decisión tree with Hellinger distance splitting criterion.
   While decisión trees are popular both in static imbalanced or balanced streaming data mining areas [13], for online skewed data there exists only a modification of Hoeffding Tree using Hellinger distance for conducting splits [5]. This metric, although skew insensitive, may still fail for difficult imbalanced datasets with complex class structures. On the other hand, it imposes minimal additional computational cost on the classifier - a highly desirable property in data stream mining. In non-stationary scenarios using data preprocessing is challenging and may lead to a prohibitively increased computational complexity. Therefore, algorithm-level solutions are worth pursuing and we will concentrate on them in this paper.
   Another decision tree algorithm for imbalanced data streams.

10. **Learning Decision Trees from Data Streams with Concept Drift (2016)_CEVOT: No aparece en los surveys (1), (2), (3), (4), (5), (6) y (7)**
    - The proposed algorithm, named **Concept-adapting Evolutionary Algorithm For Decision Tree** does not require any knowledge of the environment such as numbers and rates of drifts. The novelty of the approach is **combining tree learner and evolutionary algorithm**, where the decision tree is learned incrementally and all information is stored in an internal structure of the trees’ population. The proposed algorithm is experimentally compared with state-of-the-art stream methods on several real live and synthetic datasets.
    - There is lack of adaptive approaches for evolutionary tree learning.
    - Proposed algorithm is extension of our batch (offline) algorithm EVO-Tree

- CEVOT learns from the sliding windows without making any assumption about the nature or type of a drift nor on presence or lack of new concept classes. The novelty of the approach is **combining tree learner and evolutionary algorithm, where the decision tree is learned incrementally and all information (knowledge) is stored in the internal structure of the population of trees.**
- This method has other advantages:
  - A natural variable-length encoding structure is used, because the optimal size of a tree for a given data set is not known a priori;
  - The initial algorithm allows for random generation of unbalanced trees of different sizes;
  - The fitness function allows for simultaneous optimization of both, the accuracy and the tree size;
  - All crossover and mutation operators are designed in such a way that as a result only correct individuals, i.e. decision trees, are created.
- CEVOT inherits from EVO-Tree an evolutionary computation to process population of trees. The difference is its ability to handle data streams and gather knowledge.
- CEVOT uses **fixed size sliding window** that takes a chunk of data of size w and retrains the model with the last w examples. Because nonstationary environment is considered, CEVOT algorithm will evolve with data.
- Algorithm starts by random generation of an initial population. Nonetheless, this action takes place only when the first data chunk is received. For each subsequent chunks, CEVOT starts with population which remained from the previous run – this is a key feature which maintains “memory”. The idea is that the algorithm adapts itself to data. With every incoming chunk, population of trees shall converge to a current state (i.e. concept) and improve their accuracy.
- The main limitation of the blind approaches is slow reaction to the concept drift in data. CEVOT forgets old concepts at a constant speed, independently of whether changes are happening or not (individuals selection process). To discard old information we implemented a special destructive mutation mechanism. It works as follows. Randomly selected internal nodes are converted to leaves and the sub-trees rooted at those nodes are pruned.
- Comparison with Hoeffding Adaptive Tree (HAT), Hoeffding Tree (HT), Naïve Bayes (NB), Accuracy Updated Ensemble (AUE) (based on HT), Accuracy Weighted Ensemble (AWE) (based on HT):
  - CEVOT creates the smallest model for prediction (a single tree classifier). This result is achieved through the optimal tree structure encoding and minimizing size of the tree in evolutionary algorithm.

    Tthe fastest classifiers were NB, HT and HAT, respectively. Unfortunately, these algorithms also have the lowest prediction performance. This fact can be explained by the simplicity of their models.

        - -Ensemble models received not much worse times than single classifier models (NB, HT, HAT) but AUE predicts much better than AWE. The details are given in Table 4. CEVOT turned out to be the slowest classifier. Evolutionary computation are very time-consuming processes. The positive aspect is the fact that most of real live datasets processing time was about five second

11. **Cost-Sensitive Perceptron Decision Trees for Imbalanced Drifting Data Streams (2017): No aparece en los surveys (1), (2), (3), (4), (5), (6) y (7)**
    - They propose an efficient and fast **cost-sensitive decisión tree learning scheme for handling online class imbalance**. In each leaf of the tree they train a perceptron with output adaptation to compensate for skewed class distributions, while McDiarmid's bound is used for controlling the splitting attribute selection. The cost matrix automatically adapts itself to the current imbalance ratio in the stream, allowing for a smooth compensation of evolving class relationships. Furthermore, they analyze characteristics of minority class instances and incorpórate this information during the model update process. It allows their classifier to focus on most difficult instances, while a sliding window keeps track of changes in class structures.
    - In this paper they propose a novel decision tree learning approach for handling imbalanced and drifting data streams. As a base for our model, they use **fast perceptron trees** and improve them to become skew-insensitive by using a **moving threshold** solution. It aims at re-balancing the supports for each class during the decision making step, thus alleviating the skew bias with almost no additional computational cost. This is achieved by weighting support functions for each class according to a specified cost function. In our solution the cost matrix evolves over time and adapts to the current state of the stream. This allows them to propose an adaptive cost-sensitive solution that is able to learn from both binary and multi-class imbalanced data streams. We augment it with drift detection and use McDiarmids bound for controlling the splitting attribute selection. Additionally, they show how to analyze the structure of minority classes in an online manner by using a sliding window. This allows us to estimate the **difficulty of incoming minority class instances**, giving an additional insight into the current state of the stream. They propose an efficient method of **incorporating this background information into the update process of the proposed decision tree in order to better capture the minority class characteristics**.
    - They propose to build their learning algorithm for imbalanced and drifting data streams on top of the Fast Perceptron Decision Tree (paper Fast perceptron decision tree learning from evolving data streams).
    - They use an online perceptron approach with sigmoid activation function (as suggested by Bifet et al. [1]) with squared error optimization.
    - Bifet et al. [1] proposed to use sigmoid activation function instead of a traditional threshold and they follow this approach.

- As they deal with online learning, a stochastic gradient descent is being used with weights updated after each instance [1]. A single perceptron is trained per each class, making it suitable for both binary and multi-class problems. To obtain a final prediction regarding the class of new instance they select the highest value of support functions returned by each perceptron.
- They propose to take an advantage of **using perceptrons in leafs of the decisión tree and enhance them with cost-sensitive approach**. This will be achieved by modifying the output of each perceptron, instead of changing the structure of the training data or the training algorithm.
- This solution is highly compatible with data stream mining requirements, as it does not impose significant additional computational needs, do not rely on data preprocessing and can be easily included in the proposed decision tree learning scheme, taking the advantage of McDiarmid's inequality. Additionally, it is easily applicable for both binary and multi-class data streams, making it a versatile approach.
- They propose a simple, yet effective approach of monitoring the current imbalance ratio among classes and setting the cost according to local pairwise imbalance ratios. This will allow for an easy modeling of multi-minority and multimajority cases.
- As the stream evolved over time one cannot keep all of previous information regarding class relationships. Therefore, we propose to use a **fixed time threshold**, as well as **time stamps with each recorded label** and use them to remove outdated cases from imbalance ratio counting. This allows for dynamically adapting our cost matrix to changes and drifts in the data stream.
- In order to efficiently learn from imbalanced and drifting data streams, we require tools that will be able to monitor the imbalance ratio and the appearance of concept drifts. We propose **to combine their Cost-Sensitive Perceptron Decision Tree with Drift Detection Method for Online Class Imbalance (DDM-OCI)**.
- DDM-OCI was proposed for binary online imbalance, but can be easily extended to multi-class cases. Here, we monitor the averaged recall over all of minority classes.
- Imbalance ratio among classes is not the sole source of learning difficulty. The underlying class structures, overlapping and noisy instances have significant impact on the decision boundaries being estimated. Therefore, one may asume that minority class instances may pose a different level of difficulty to the learning procedure. In this work, they propose to analyze the difficulty of incoming minority class objects, while taking into account the evolving structure of classes.
- They propose six levels of difficulty λ that can be assigned to each new minority instance based on how contaminated is its neighborhood. This is measured by parameter ρ that states how many of k neighbors belong to the same minority class. They propose to label each new minority class instance based on this analysis.

- As they deal with an online scenario, they cannot nor want to keep the entire stream in the memory. Therefore, they propose to analyze the types of minority instances using a small **sliding window** that will keep only the most recent instances, allowing for a fast neighborhood search within it. Additionally, they will incorporate the information from the drift detector.
- Another issue lies in how to utilize this information regarding minority class structure during the online learning process. Each new minority instance will be presented to the cost-sensitive perceptron tree λ times during online learning, where λ is the difficulty level associated with this instance. This will shift their classifier towards concentrating on difficult instances, which in turn should lead to a better predictive performance.

- Comparison with Fast Perceptron Decision Tree (PDT) and Hellinger Hoeffding Decision Tree (HHT):
    - Original implementation of Fast Perceptron Decision Tree used Hoeffding inequality to determine the amount of instances needed for conducting a Split [1]. However, recent study discussed flaws in the Hoeffding bound [8]. In this work, they propose to modify the underlying base of the original Fast Perceptron Decision Tree and use a McDiarmid's inequality for controlling the splitting criteria. It is a generalization of the Hoeffding's inequality, being applicable to both numerical and non-numerical data, as well as better describing the Split measures.
    - While decisión trees are popular both in static imbalanced or balanced streaming data mining areas [13], for online skewed data there exists only a modification of Hoeffding Tree using Hellinger distance for conducting splits [5].
    - Their algorithm is directly applicable for both binary and multi-class data streams, while for multi-class cases they modify HHT to conduct binary decompositions in each split in an identical fashion as its static versión. Furthermore, they evaluate the performance of the proposed cost-sensitive algorithm without taking into account the types of minority instances (CSPT) and with this extension included (CSPT+).
    - Binary Imbalanced Data Streams: Standard PDT cannot tackle skewed distributions and becomes easily biased towards the majority class. HHT performs much better. yet CSPT and CSPT+ outperform it on 10 out of 12 data streams. This can be explained by Hellinger split criterion not being enough to counter severe class imbalance and difficult minority class structures.
    - Binary Imbalanced Data Streams: In most cases CSPT+ returns the superior performance, showing that the proposed online analysis of minority instances difficulty can be beneficial to the learning process (most of datasets). They may conclude that difficult minority instances are bound to happen in online learning scenarios, especially when the minority class structure is

constantly evolving. Therefore, it is worthwhile to incorpórate such information during online classifier updating.

  - Binary Imbalanced Data Streams: When taking into account both time and memory resources being used, one can see that perceptron-based solutions are faster than HHT. CSPT displays almost identical resource usage as the native PDT, proving that the proposed cost-sensitive modification and adaptive cost matrix does not impose any significant additional costs. CSPT+ displays slightly higher computational requirements, which was to be expected as for each new minority instance it analyzes its type and needs to store instances in a sliding window. However, this search is conducted only for minority instances, leading only to a slight increase in overall resource consumption which is far from being prohibitive.
  - Multi-class Imbalanced Data Streams: PDT fails to deliver satisfactory performance. However, we can see much bigger discrepancies between HHT and CSPT/CSPT+. As Hellinger distance is a binary metric, to adapt it for multiclass problems one must use a binary decomposition at each node and ten average the metric results when conducting splits. Our experiments show that this fails for multi-class imbalanced data streams. CSPT+ always returns the superior performance.
  - When analyzing the resource usage, we can see that perceptron-based solutions increased their costs. This is due to higher number of perceptrons being trained at each leaf. Additionally, CSPT+ needs to store more instances in the sliding window and conduct more instance difficulty analyses, as minority instances may arrive from multiple classes. However, the displayed complexity does is not prohibitive and shows that CSPT+ can be used in real-life scenarios with multi-class imbalanced data streams.

12. **Strict Very Fast Decision Tree: a memory conservative algorithm for data stream mining (2018): No aparece en los surveys (1), (2), (3), (4), (5), (6) y (7)**
    - This study proposes the Strict VFDT (SVFDT), a novel algorithm based on the VFDT. The SVFDT algorithm minimises unnecessary tree growth, substantially reducing memory usage and keeping competitive predictive performance.
    - In order to deal with memory cost restrictions, keeping the predictive performance, we propose a new base learner, called Strict Very Fast Decision Tree (SVFDT).
    - Two SVFDT versions were proposed, one designed to consume less memory and training time (SVFDT-I) and another (SVFDT-II) with a higher predictive performance.
    - Comparison with VFDT, OVFDT, OVFDT with functional tree leaf - majority class, naive bayes and adaptive hybrid integrations and Tie-breaking in Hoeffding Trees:

        - Since it creates much more shallow trees than VFDT, SVFDT can achieve a shorter processing time.
        - In the last years, (Holmes et al., 2005; Yang and Fong, 2011b, 2013) proposed a series of modifications to increase the predictive performance of the VFDT algorithm. However, this came with a substantial increase in the memory cost.
        - SVFDT is faster than the VFDT in some cases.
        - SVFDT uses significantly less memory in comparison to VFDT, reaching similar predictive performance

13. **Extremely Fast Decision Tree (2018)_HAT_: No aparece en los surveys (1), (2), (3), (4), (5), (6) y (7)**
    - Novel incremental decision tree learning algorithm, Hoeffding Anytime Tree, that is statistically **more efficient than current state-of-the-art, Hoeffding Tree**.
    - Hoeffding Anytime Tree produces the asymptotic batch tree in the limit, is naturally resilient to concept drift, and can be used as a higher accuracy replacement for Hoeffding Tree in most scenarios, at a small additional computational cost.
    - In practice, if no split attribute exists at a node, rather than splitting only when the top candidate split attribute outperforms the second-best candidate, HATT will split when the information gain due to the top candidate split is non-zero with the required level of confidence. At later stages, HATT will split when the difference in information gain between the current top attribute and the current split attribute is non-zero, assuming this is better than having no split.
    - On nominal with data with d attributes, v values per attribute, and c classes, HATT requires O(dvc) memory to store node statistics at each node, as does HT.
    - There are two primary operations associated with learning for HT: (i) incorporating a training example by incrementing leaf statistics and (ii) evaluating potential splits at the leaf reached by an example. The same operations are associated with HATT, but we also increment internal node statistics and evaluate potential splits at internal nodes on the path to the relevant leaf.

      Comparison with VFDT, CVFDT and HAT:
        - Our implementation of the Hoeffding Anytime Tree algorithm, the Extremely Fast Decision Tree (EFDT), achieves higher prequential accuracy than the Hoeffding Tree implementation Very Fast Decision Tree (VFDT) on many standard benchmark tasks.
        - HT constructs a tree incrementally, delaying the selection of a split at a node until it is confident it has identified the best split, and never revisiting that decision. In contrast, HATT seeks to select and deploy a split as soon as it is confident the split is useful, and

then revisits that decision, replacing the split if it subsequently becomes evident that a better split is available.

- The HT strategy is more efficient computationally, but HATT is more efficient statistically, learning more rapidly from a stationary distribution and eventually learning the asymptotic batch tree if the distribution from which the data are drawn is stationary. Further, false acceptances are inevitable, and since HT never revisits decisions, increasingly greater divergence from the asymptotic batch learner results as the tree size increases.
- They observe VFDT taking longer and longer to learn progressively more difficult concepts obtained by increasing the number of classes. EFDT learns all of the concepts very quickly. and keeps adjusting for potential overfitting as fresh examples are observed.
- In scenarios where information distribution among attributes is skewed, with some attributes containing more information than others, such a policy (building structure that improves on the current state but making subsequent corrections when further alternatives are found to be even better) can be highly effective because of the limited cost of rebuilding the tree when replacing a higher-level attribute with a highly informative one. However, where information is more uniformly distributed among attributes, Hoeffding Tree will struggle to split and might have to resort to using a tie-breaking threshold that depends on the number of random variables, while HATT will pick an attribute to begin with and switch when necessary, leading to faster learning.
- The idea common to both CVFDT and HATT is that of split reevaluation. However, the circumstances, objectives, and methods are entirely different. CVFDT is explicitly designed for a drifting scenario; HATT for a stationary one. CVFDT's goal is to reduce prequential error for the current window in the expectation that this is the best way to respond to drift; HATT's goal is reduce prequential error overall for a stationary stream so that it asymptotically approaches that of a batch learner. CVFDT builds and substitutes alternate subtrees; HATT does not. CVFDT deliberately employs a range of forgetting mechanisms; HATT only forgets as a side effect of replacing splits—when a subtree is discarded, so too are all the historical distributions recorded therein. CVFDT always compares the top attributes, while HATT compares with either the current split attribute or the null split. However, CVFDT is not incompatible with the core idea of Hoeffding Anytime Tree; it would be interesting to examine whether the idea of comparing with the null split or the current split attribute when applied to CVFDT will boost its performance on concept drifting streams.
- Hoeffding Adaptive Tree (HAT) builds a tree that grows alternate subtrees if a subtree is observed to have poorer prequential

accuracy on more recent examples, and substitutes an alternate when it has better accuracy than the original subtree. HAT uses an error estimator, such as ADWIN [5] at each node to determine whether the prediction error due to a recent sequence of examples is significantly greater than the prediction error from a longer historical sequence so it can respond to drift. HATT, on the other hand, does not rely on prediction results or error, and does not aim to deliberately replace splits in response to drift. HATT has some inbuilt tolerance to concept drift, though it is not specifically designed as a learner for drift. It is easy to conceive of ensemble, forgetting, decay, or subtree replacement approaches built upon HATT to deal with concept drift, along the lines of approaches that have been proposed for HT.

- Thus, they expect HATT to have an advantage over HT in situations where HT considerably delays splits at each level—such as when the difference in information gain between the top atributes at a node is low enough to require a large number of examples in order to overcome the Hoeffding bound, though the information gains themselves happen to be significant. This would lead to a potentially useful split in HT being delayed, and poor performance in the interim.
- Conversely, when the differences in information gain between top attributes as well as the information gains themselves are low, it is possible that HATT chooses a split that would require a large number of examples to readjust. However, since we expect this to keep up with VFDT on the whole, the main source of underperformance for EFDT is likely to be an overfitted model making low-level adjustments.
- Hoeffding AnyTime Tree makes a simple change to the current de facto standard for incremental tree learning. The current state-of-the-art Hoeffding Tree aims to only split at a node when it has identified the best possible split and then to never revisit that decision. In contrast HATT aims to split as soon as a useful split is identified, and then to replace that split as soon as a better alternative is identified. Their results demonstrate that this strategy is highly effectively on benchmark datasets.
- HT cautiously works toward the asymptotic batch tree, ignoring, and thus not benefiting from potential improvements on the current state of the tree, until it is sufficiently confident that they will not need to be subsequently revised.