

Efficient Online Decision Tree Learning with Active Feature Acquisition

Arman Rahbar¹, Ziyu Ye², Yuxin Chen² and Morteza Haghir Chehreghani¹

¹Chalmers University of Technology

²University of Chicago

armanr@chalmers.se, ziyuye@uchicago.edu, chenyuxin@uchicago.edu,
morteza.chehreghani@chalmers.se

Abstract

Constructing decision trees online is a classical machine learning problem. Existing works often assume that features are readily available for each incoming data point. However, in many real world applications, both feature values and the labels are unknown *a priori* and can only be obtained at a cost. For example, in medical diagnosis, doctors have to choose which tests to perform (*i.e.*, making costly feature queries) on a patient in order to make a diagnosis decision (*i.e.*, predicting labels). We provide a fresh perspective to tackle this practical challenge. Our framework consists of an active planning oracle embedded in an online learning scheme for which we investigate several information acquisition functions. Specifically, we employ a surrogate information acquisition function based on adaptive submodularity to actively query feature values with a minimal cost, while using a posterior sampling scheme to maintain a low regret for online prediction. We demonstrate the efficiency and effectiveness of our framework via extensive experiments on various real-world datasets. Our framework also naturally adapts to the challenging setting of online learning with concept drift and is shown to be competitive with baseline models while being more flexible.

1 Introduction

Decision trees constitute one of the most fundamental and crucial machine learning models, due to their interpretability and extensibility. An important variant developed for online setting has been employed in various impactful real-world applications such as medical diagnosis [Podgorelec *et al.*, 2002], intrusion detection [Jiang *et al.*, 2013], network troubleshooting [Rozaki, 2015], etc.

Classical models aim to construct online decision trees incrementally with streaming data. However, such models have several disadvantages. First, they require that all features are presented to determine splitting node ([Das *et al.*, 2019; Féraud *et al.*, 2016]). However, querying feature values can be costly in real-world scenarios, e.g., conducting medical tests for medical diagnosis can be quite expensive. Second,

classical models are typically not *fully* trained online, where labels are assumed to be known for each point in the data stream [Shim *et al.*, 2018]. In contrast, our work takes feature acquisition cost (formally defined in section 3) into considerations and aim at the more challenging fully online case: we receive a data point at each step, and need to make accurate prediction of its label with low feature acquisition cost; the true label will only be observed *after* we make the prediction.

Concretely, consider the medical diagnosis problem: at each round t , a patient \mathbf{x}^t comes in, and the system is asked to predict the treatment \mathbf{y}^t for the patient. Naturally, we take results of medical tests (e.g., a CT scan) as patients’ features: assume that there exist n medical tests, each patient \mathbf{x}^t can be represented by $(x_1^t, x_2^t, \dots, x_n^t)$, where x_i^t is their i -th test results. To reiterate, the problem has two crucial characteristics, making the setting challenging yet more practical: (1) **cost of feature query**: we assume that features of a data point is initially unknown but can be acquired with a cost; (2) **(fully) online learning**: we assume that we only have prior belief on the data and have to refine it via online interaction with the data streams (*i.e.*, patients) whose labels (*i.e.*, treatments) are initially unknown in each round.

Our goal is to construct online decision trees efficiently. Specifically, we interpret the efficiency of our framework from two aspects: first, it requires less streaming data points (or time steps) to learn a well-performed decision tree, *i.e.*, *faster learning*; second, it incurs lower cost for the label prediction for each data point, *i.e.*, *cheaper prediction*. We refer our framework as **UFODT**, *i.e.*, Utility of Features for Online learning of Decision Trees. Our framework can be interpreted as an active planning oracle nested within an online learning model.

For the online learning part (in the outer loop), we employ posterior sampling [Osband and Van Roy, 2017] to learn the online decision tree model. In addition to the established regret guarantee in the canonical online learning setting, another advantage is that posterior sampling can effectively leverage data-dependent prior knowledge, which the classical online decision tree models often fail to capture.

The active planning oracle adopts a decision-theoretic design: we aim to optimize the *utility of the features*, (informally) defined as the expected prediction error for the incoming data point in the data stream, should we observe the value of the chosen features. In order to *efficiently* optimize the

utility of features, we consider adaptive surrogate objective functions following the key insight of [Golovin *et al.*, 2010] to sequentially query features and their values, which enables us to predict accurately with low cost. In particular, the sequential feature query based on our surrogate objective is a natural analogy to the information gain node splitting criterion in the classical decision tree literature.

Our contributions are summarized as follows¹:

- We introduce a novel setting of online decision tree learning where the learner does not have *a priori* access to both the feature values and the labels, and propose an efficient algorithmic framework for constructing online decision trees in a cost-effective manner.
- Our framework consists of several novel algorithmic contributions, including a novel surrogate objective as node splitting criterion (section 4.2), an extension to efficiently handle real-valued feature values (section 4.3), a variant to handle concept drift in streaming data (section 4.4), and an online feature selection scheme that further reduces the computational cost (appendix).
- We provide a rigorous theoretical analysis and justification of our algorithm, in terms of both a prior-independent and a prior-dependent regret bound.
- We perform extensive experiments on diverse real-world datasets to verify that our framework is able to achieve competitive (or even better) accuracy with much lower cost, compared to baseline models.

2 Related Work

Online Decision Tree. Traditional models consider to build online decision tree (ODT) incrementally. [Domingos and Hulten, 2000] first propose VFDT to learn decision tree from streaming data, and use Hoeffding bound to guarantee the model performance; VFDT later becomes the de facto baseline in this domain. [Hulten *et al.*, 2001] propose a variant to handle concept drift, but the construction for the tree growing process is complicated to implement. [Manapragada *et al.*, 2018] design Hoeffding Anytime Tree as an improvement for VFDT. [Das *et al.*, 2019] suggest a bootstrap strategy to enhance the memory efficiency of VFDT. It is important to note that all those models are not fully online, nor do they consider feature query cost. Another line of work considers applying reinforcement learning to build decision trees online [Garlapati *et al.*, 2015; Blake and Ntoutsi, 2018]. However, these works do not have any theoretical guarantees, nor do they utilize prior knowledge (e.g., on the underlying state transition distributions).

Posterior sampling based online learning. Posterior sampling, also called Thompson sampling, is first proposed in [Thompson, 1933] to solve bandit problems in clinical trials, and the central idea is to select actions according to its posterior probability to be optimal. It later becomes an important policy in online learning problems, showing excellent performance empirically and theoretically. [Osband *et al.*, 2013;

¹An extended version of this paper is available at arxiv.org/abs/2305.02093. The reader may refer to that version for appendices.

Agrawal and Jia, 2017; Fan and Ming, 2021] apply posterior sampling and prove its efficiency in reinforcement learning; this line of work is generally referred to as PSRL. Our work is closest to [Chen *et al.*, 2017b], which adapts PSRL to solve online information acquisition problems; however, in contrast to our work, [Chen *et al.*, 2017b] consider a more constrained application domain of interactive troubleshooting, and fail to tackle concept drift which is often crucial in data-streaming scenario; in addition, it tackles the hypothesis space in more restricted ways.

Active feature acquisition. The line of work on active feature acquisition (AFA) seeks to solve specific tasks like classification when data features are acquirable at a cost. [Kapoor and Horvitz, 2009] consider the restrictive setting where features and labels are boolean; [Bilgic and Getoor, 2007] propose a decision-theoretic strategy with Bayesian networks to calculate the value of information of features; [Shim *et al.*, 2018] suggest a joint framework to dynamically train the classifier while acquiring features. Conducting AFA online has not been discussed until recently, for example, [Beyer *et al.*, 2020] apply classical information acquisition techniques like *information gain* to handle streaming data with missing features.

Adaptive information acquisition for decision making. As fundamentals of sequential decision making, the goal of those works is to design an adaptive policy to identify an unknown target (i.e., a decision) by sequentially picking tests and observing outcomes (i.e., acquiring information). There are well-known greedy heuristics for the adaptive policy such as Information Gain (IG) [Dasgupta, 2005] and Uncertainty Sampling (US) which greedily maximize the uncertainty reduction (over different random variables). Recently, researchers propose to optimize w.r.t. submodular surrogates, e.g., EC² [Golovin *et al.*, 2010], HEC [Javdani *et al.*, 2014], ECED [Chen *et al.*, 2017a], which are proven to have near-optimal performance with low information acquisition cost. The above-mentioned policies naturally fit into our problem and can all be used in the active planning phase.

In Table 1, we provide a comparison of our work with existing ODT literature.

3 Problem Formulation

3.1 Efficient Online Decision Tree Learning

Simply put, our task is to predict labels (classes) of streaming data points by building a decision tree online with low feature acquisition cost. At each epoch (time) t , we receive a data point x^t , whose feature values and label are unknown. To make prediction for x^t , we can gather information by querying feature values; each query incurs a cost. The label of x^t will only be revealed at the end of each epoch after we make the prediction.

Formally, let $\mathbf{x} = (x_1, x_2, \dots, x_n)$ be the *data point* with n features. Let $X_i \in \mathcal{X} \triangleq \{0, 1\}$ denotes the random variable for the *feature value* of the i th feature², and

²For simplicity, in this section we assume features are binary. Our setting is extended to multicategorical or continuous feature cases in Section 4.3, where more details can be found in appendix.

Algorithms	Learning setting	Cost of feature query
Online decision tree (e.g., [Das <i>et al.</i> , 2019])	Semi-online (labels received in foresight)	No
Bandit tree (e.g., [Féraud <i>et al.</i> , 2016])	Online (no labels but rewards received)	No
Active feature acquisition (e.g., [Shim <i>et al.</i> , 2018])	Offline or semi-online (labels received in foresight)	Yes
UFODT (Ours)	Online (labels received only in hindsight)	Yes

Table 1: Comparison of our framework with existing ODT literature. We make the first concrete attempt on taking feature acquisition cost into online decision tree learning problems.

let $Y_j \in \mathcal{Y} \triangleq \{y_1, y_2, \dots, y_m\}$ be the random variable for the *label* of the data point. The superscript t denotes that the data is received at epoch t . We adopt the common naïve Bayes assumption to model underlying probabilistic structure: $\mathbb{P}[Y_j, X_1, \dots, X_n] = \mathbb{P}[Y_j] \prod_{i=1}^n \mathbb{P}[X_i | Y_j]$, i.e., features are conditionally independent given the class. Since we are in the online setting, we assume that the *joint distribution* $\mathbb{P}[Y, X_1, \dots, X_n]$ is initially unknown (though we may have prior knowledge on that) and needs to be learned via our online interactions.

We define $\theta_{ij} \triangleq \mathbb{P}[X_i = 1 | Y_j]$, and assume that θ follows a Beta distribution, $\text{Beta}(\alpha_{ij}, \beta_{ij})$. We use $\boldsymbol{\theta} = [\theta_{ij}]_{n \times m}$ to denote the probabilistic table for the data distribution and assume $\boldsymbol{\theta} \sim \text{Beta}(\boldsymbol{\alpha}, \boldsymbol{\beta})$. Under the above probabilistic model, each query on a feature value will provide some information about Y . We define the set of *queries* as $\mathcal{Q} \triangleq \{1, 2, \dots, n\}$, and a query $i \in \mathcal{Q}$ will reveal the value of the i th feature. We define *cost* of the feature query as $c : \mathcal{Q} \rightarrow \mathbb{R}_{\geq 0}$. Upon information gathered from feature query, we make a prediction. We define the loss of our prediction as $l : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}_+$. Our goal is to reach low prediction loss with low query cost on data stream.

We additionally define $H = [X_1, \dots, X_n]$ as the random variable for the *hypothesis* of a data point. Thus, each hypothesis h corresponds to a full realization of the outcome of all queries in \mathcal{Q} . Let $h \in \mathcal{H} \triangleq \{0, 1\}^n$. Importantly, the set \mathcal{H} can be partitioned into m disjoint *decision regions*, that each class in \mathcal{Y} corresponds to a decision region. Later, we may use the term “decision region” to implicitly refer “label” or “class”.

Under this construction, our goal then becomes building a decision tree which identifies the *decision region* for each data point arriving to us. Such identification of decision region should be done with low *cost*. This enables a decision-theoretic perspective as follows.

Algorithm 1 Online Decision Tree Learning

Input: Prior $\mathbb{P}(Y)$ and $\mathbb{P}(\boldsymbol{\theta})$.

- 1: **for** $t = 1, 2, \dots, T$ **do**
- 2: Sample $\boldsymbol{\theta}^t \sim \text{Beta}(\boldsymbol{\alpha}^{t-1}, \boldsymbol{\beta}^{t-1})$ and receive \mathbf{x}^t ;
- 3: Call Algorithm 3 with $\boldsymbol{\theta}_t$ to sequentially query features and predict the label (online);
- 4: Observe \mathbf{x}_F^t and true label y_j^t ;
- 5: Call Algorithm 2 to obtain $\text{Beta}(\boldsymbol{\alpha}^t, \boldsymbol{\beta}^t)$

3.2 Utility of Features

At each epoch, we perform a set of queries $\mathcal{F} \subseteq \mathcal{Q}$, and let the outcome vector be \mathbf{x}_F , which can be conceived as a partial realization for the hypothesis h of \mathbf{x} .

Let y be our label prediction, and denote its associated

loss w.r.t. the true data label y_{true} as l . We can then naturally define the *utility* of y as $u \triangleq -l$ and the *conditional expected utility* of y upon observing \mathbf{x}_F as $U(y | \mathbf{x}_F) \triangleq \mathbb{E}_{y_{\text{true}}} [u(y_{\text{true}}, y) | \mathbf{x}_F]$. Note that we could define the utility similarly upon h , since h is the full realization of features.

Definition 1. (*Utility of features \mathbf{x}_F*)

$$\mathbb{U}(\mathbf{x}_F) \triangleq \max_{y \in \mathcal{Y}} U(y | \mathbf{x}_F).$$

Here, $\mathbb{U}(\mathbf{x}_F)$ represents the maximal expected utility achievable given \mathbf{x}_F . This formulation is similar to the value of information [Howard, 1966], and can connect with the generalization error of the classical empirical risk minimization framework [Vapnik, 1992], however, what we would like to emphasize here is that such utility relies on the *partial realization of features*, that we seek to find the cheapest query set \mathcal{F} to achieve the maximal utility.

We then define the decision region for y as the set of hypotheses for which y is the optimal label prediction:

Definition 2. (*Decision region for y*)

$$\mathcal{R}_y \triangleq \{h : U(y | h) = \mathbb{U}(h)\}.$$

Directly optimizing $\mathbb{U}(\mathbf{x}_F)$ is usually intractable, and greedy heuristics may fail or be costly. In the next, we will show how we may use a *surrogate objective* of $\mathbb{U}(\mathbf{x}_F)$ by the notion of decision regions to achieve near optimal query planning.

4 Proposed Framework

We now present our UFODT framework for efficient online decision tree learning. The high-level structure can be conceived as an *active planning oracle* nested within an *online learning model*. We use the posterior sampling strategy for the online learning model, and a surrogate optimization algorithm on utility of features for the active planning oracle.

4.1 Online Learning by Posterior Sampling

Assume that we have access to the prior of the environment parameter $\boldsymbol{\theta}$. Firstly, at the beginning of each epoch t , we sample $\boldsymbol{\theta}^t$ from the (posterior) distribution of $\boldsymbol{\theta}$. Then, we run an adaptive policy which sequentially queries features (i.e., splitting nodes) of \mathbf{x}^t , in order to optimize some objectives (e.g., information gain, utility of features, etc.); importantly, such a policy can be conceived as an offline oracle, as its planning is fixed upon each sampled $\boldsymbol{\theta}^t$. The policy will suggest a label prediction for \mathbf{x}^t . Finally, the true label for \mathbf{x}^t is revealed, and is then used to update the posterior of $\boldsymbol{\theta}$ together with the query observation \mathbf{x}_F . The pseudo-code is provided in Algorithm 1.

Algorithm 2 Posterior Update

Input: $x_{\mathcal{F}}^t; y_j^t; (\alpha^{t-1}, \beta^{t-1})$.

- 1: **for** each $(i, x_i) \in x_{\mathcal{F}}^t$ **do**
- 2: **if** $x_i = 1$ **then** $\alpha_{ij}^t \leftarrow \alpha_{ij}^{t-1} + 1$
- 3: **else** $\beta_{ij}^t \leftarrow \beta_{ij}^{t-1} + 1$
- 4: **Return** (α^t, β^t)

4.2 Planning by Surrogate Optimization

Algorithm 3 Planning by Surrogate Optimization

Input: Prior $\mathbb{P}(Y)$ and θ .

- 1: Sample hypotheses by calling Algorithm 4
- 2: $\mathcal{O} = \emptyset$
- 3: **while** stopping condition for EC² not reached **do**
- 4: Use EC² to determine next feature $i \in \mathcal{Q}$
- 5: Query feature i
- 6: Add (i, x_i) to \mathcal{O}
- 7: Update $\mathbb{P}(h \mid \mathcal{O})$ based on $\mathbb{P}(Y)$ and θ
- 8: **Return** the decision region y

In the planning phase, we seek to optimize an objective of $\mathbb{U}(x_{\mathcal{F}})$ given the sampled environment. We propose to optimize for the *surrogate objective* of $\mathbb{U}(x_{\mathcal{F}})$. Specifically, we focus on the EC² algorithm [Golovin *et al.*, 2010], which uses the equivalence class edge cut as the surrogate objective of $\mathbb{U}(x_{\mathcal{F}})$. Importantly, this surrogate objective function is adaptive submodular, and hence a greedy algorithm could attain a near optimal solution, allowing us to make accurate prediction with low query and computational cost.

In EC², we define a weighted graph $G = (\mathcal{H}, E)$, where $E \triangleq \bigcup_{y \neq y'} \{\{h, h'\} : h \in \mathcal{R}_y, h' \in \mathcal{R}_{y'}\}$, denoting the pairs of hypotheses with different labels. The weight of each edge is $w(\{h, h'\}) \triangleq \mathbb{P}(h) \cdot \mathbb{P}(h')$. Specifically, $\mathbb{P}(h)$ can be conceived as posterior distribution upon query of existing feature values. We define the weight of a set of edges as $w(E') \triangleq \sum_{\{h, h'\} \in E'} w(\{h, h'\})$. Therefore, performing a *feature query* is considered as *cutting an edge*, which can also be conceived as removing inconsistent hypotheses with all their associated edges. We thus have the edge set $E(x_i)$ cut after observing the outcome of a feature query x_i : $E(x_i) \triangleq \{\{h, h'\} \in E : \mathbb{P}[x_i \mid h] = 0 \vee \mathbb{P}[x_i \mid h'] = 0\}$. Based on the graph G , we formally define the EC² objective as $f_{EC^2}(x_{\mathcal{F}}) \triangleq w(\bigcup_{v \in \mathcal{F}} E(x_v))$, and the score of feature query is defined as $\Delta_{EC^2}(u \mid x_{\mathcal{F}}) \triangleq \mathbb{E}_{x_u \mid x_{\mathcal{F}}} [f_{EC^2}(x_{\mathcal{F} \cup \{u\}}) - f_{EC^2}(x_{\mathcal{F}})]$. The policy π_{EC^2} will greedily query the feature which maximizes the gain cost ratio $\Delta_{EC^2}(v \mid x_{\mathcal{F}}) / c(v)$ and stops when only one decision region exists. We present the algorithm in Algorithm 3. Note that the EC² objective in the line 3 and line 4 can be flexibly replaced by other active information acquisition functions like Information Gain (IG) and Uncertainty Sampling (US), which we elaborate in the appendix.

Hypothesis Sampling Procedure. Information acquisition methods such as Information Gain and Uncertainty Sampling, and also EC² require enumeration of hypothesis space, which can be computationally challenging. Thereby, to reduce the number of hypotheses, we use a sampling procedure sketched

in Algorithm 4. In this algorithm we first sample a decision region using the prior distribution over the classes and then we exploit the current estimate of θ to build a new sample.

Algorithm 4 Hypotheses Sampling

Input: Prior $\mathbb{P}(Y)$ and θ .

- 1: $\tilde{\mathcal{H}} \leftarrow \emptyset$
- 2: Sample decision regions from $\mathbb{P}(Y)$
- 3: **for** each sampled decision region j **do**
- 4: $h \leftarrow \emptyset$
- 5: **for** each $i \in \mathcal{Q}$ **do**
- 6: Sample $X_i \sim \text{Ber}(\theta_{ij})$ and add to h
- 7: $\tilde{\mathcal{H}} = \tilde{\mathcal{H}} \cup h$
- 8: **Return** $\tilde{\mathcal{H}}$

Algorithm 5 Threshold selection

Input: η

- 1: $S_{ik}^{(0)} \leftarrow 0$ for all features i and thresholds k
- 2: **for** $t = 1, 2, \dots, T$ **do**
- 3: **for** each feature i **do**
- 4: **if** feature i can be queried **then**
- 5: Calculate the threshold sampling distribution:

$$\Pi_{ti}(k) = \frac{\exp(\eta S_{ik}^{(t-1)})}{\sum_{k'} \exp(\eta S_{ik'}^{(t-1)})}$$
- 6: Sample threshold $B_{ti} \sim \Pi_{ti}$ and observe gain Δ_{ti}
- 7: Calculate $S_{ik}^{(t)}$:

$$S_{ik}^{(t)} = S_{ik}^{(t-1)} + \frac{\mathbb{1}\{B_{ti}=k\} \Delta_{ti}}{\Pi_{ti}(k)}$$
, for all k

4.3 Handling Continuous Features

One way to extend our framework for handling continuous data is to “binarize” real-valued features. In particular, for each feature we consider K different thresholds for binarization, and in each training time step, we select the threshold that maximizes the gain based on the information acquisition function (e.g., Δ_{EC^2}). By collecting the history for each threshold, we can easily calculate the posterior distribution of the parameters associated to the binary feature corresponding to that threshold. We provide the details in the Appendix D. This naive way of *exhaustively* searching for the best threshold causes a significant computational running time in each time step. Thereby, we propose a more efficient algorithm for learning the best discretization for each feature.

Learning discretizations for continuous features. We model the threshold selection process for each feature as an *adversarial bandit problem* [Auer *et al.*, 2002] with arms and rewards being the thresholds and gains, respectively. Let $\Pi_{ti} : [K] \rightarrow \mathbb{R}_{\geq 0}$ ($\sum_{k \in [K]} \Pi_{ti}(k) = 1$) be the probability distribution according to which we select the binarization threshold for feature i at time step t . Then, we do threshold selection and update Π_{ti} with the procedure sketched in Algorithm 5 (adapted from the Exp3 algorithm [Auer *et al.*, 2002]). $S_{ik}^{(t)}$ is the sum of estimated gains for the k -th threshold of the i -th feature until time t . In time t , we use a threshold sampled from Π_{ti} and observe the gain for that threshold. Then we calculate $S_{ik}^{(t)}$ based on $S_{ik}^{(t-1)}$ and the observed gain. Specifically, for each feature, we add unbiased estimates of gains ($\frac{\mathbb{1}\{B_{ti}=k\} \Delta_{ti}}{\Pi_{ti}(k)}$) for different thresholds to the

previous sum of gains.

4.4 Handling Concept Drift

Concept drift is a crucial problem in streaming scenarios, where the dependency of features on the data label is changing over time. Classical ODTs use complicated updating criteria to handle concept drift [Hulten *et al.*, 2001]. Thanks to our posterior sampling scheme, we are able to adopt an exceptionally easy solution to tackle the concept drift problem, by simply adding two lines of code upon Algorithm 2, which is shown in Algorithm 6. This inspiration comes from non-stationary posterior sampling [Russo *et al.*, 2017]. The central idea is that we need to keep exploring in order to learn the time-varying concept. This technique encourages exploration by adding a discount parameter γ for the history, and injecting a random distribution Beta($\bar{\alpha}, \bar{\beta}$) to increase uncertainty.

5 Theoretical Analysis

In this section, we discuss the bound of the expected regret for our fully online framework (Sections 4.1 and 4.2). Here we focus on the EC² objective function due to its theoretical guarantees.

Let $\mathbb{U}(\pi) \triangleq \mathbb{E}_h [\max_{y \in \mathcal{Y}} \mathbb{E}_{y_{\text{true}}} [u(y_{\text{true}}, y) \mid \mathcal{S}(\pi, h)]]$ be the expected utility of features achieved by a policy π ; here, $\mathcal{S}(\pi, h)$ represents the set of features and their values queried by policy π upon a hypothesis h . As proved by [Golovin and Krause, 2011], by the submodularity of the EC² objective function, π^{EC^2} is able to achieve the same utility as the optimal policy π^* does under a same environment θ , with at most $(2 \ln(1/p_{\min}) + 1) \cdot c_{\pi^*}$ query cost, where p_{\min} denotes the minimal probability of a hypothesis h across environments and c_{π^*} represents the cost of the optimal policy.

Definition 3. Let θ^* denote the true environment, and let θ^t denote the sampled environment at epoch t as in line 2 of Algorithm 1. Let $\pi_{\theta^*}^*$ denote the optimal policy for θ^* , and $\pi_{\theta^t}^{\text{EC}^2}$ denote the policy with EC² as in Section 4.2. We define the immediate regret at epoch t for Algorithm 1 with the EC² objective as:

$$\Delta^t \triangleq \mathbb{U}(\pi_{\theta^*}^*) - \mathbb{U}(\pi_{\theta^t}^{\text{EC}^2}).$$

We then define the total regret at epoch T as:

$$\text{Regret}(T) = \sum_{t=1}^T \Delta^t.$$

5.1 Prior-Independent Regret Bound

Based on the result of [Osband *et al.*, 2013], we have the following regret bound for Algorithm 1:

Theorem 4. (Prior-independent regret bound) Let $L = (2 \ln(1/p_{\min}) + 1) \cdot c_{\pi^*}$ denote the worst-case cost (i.e., number of queries) for Algorithm 1 with the EC² objective in any epoch, where p_{\min} denotes the minimal probability of a hypothesis h across environments and c_{π^*} represents the cost of the optimal policy. Let S be the number of possible realizations of L queries and n be the total number of features. Assume that the sampling of decision regions by Algorithm 4 is

sufficient, such that all hypotheses with non-zero probability in the hypothesis space are enumerated. The expected total regret at epoch T for Algorithm 1 with the EC² objective is:

$$\mathbb{E}[\text{Regret}(T)] = O(LS\sqrt{nLT \log(SnLT)}).$$

We provide the proof in Appendix C.1.

The above regret bound depends on the worst-case cost L , which could potentially be huge, and the bound is also prior-independent such that the benefit of a good prior knowledge by the posterior sampling scheme is not reflected (as we illustrate in the experiments of Appendix F.5 on the impact of priors). In the appendix, we in addition provide a prior-dependent bound based on the results from [Russo and Van Roy, 2016] and [Lu *et al.*, 2021].

Algorithm 6 Handling Concept Drift

```

Input:  $x_{\mathcal{F}}^t; y_j^t; (\alpha^{t-1}, \beta^{t-1}); \gamma; (\bar{\alpha}, \bar{\beta})$ .
1:  $\alpha^t \leftarrow (1 - \gamma)\alpha^{t-1} + \gamma\bar{\alpha}$ 
2:  $\beta^t \leftarrow (1 - \gamma)\beta^{t-1} + \gamma\bar{\beta}$ 
3: for each  $(i, x_i) \in x_{\mathcal{F}}^t$  do
4:   if  $x_i = 1$  then
5:      $\alpha_{ij}^t \leftarrow \alpha_{ij}^{t-1} + 1$ 
6:   else
7:      $\beta_{ij}^t \leftarrow \beta_{ij}^{t-1} + 1$ 
8: Return  $(\alpha^t, \beta^t)$ 
```

6 Experimental Results

We now empirically validate our framework with real-world datasets. Unless otherwise specified, we assume that we have a uniform prior on θ . We evaluate the methods introduced in Section 4 from different aspects. We compute the average number of queries per online session to compare the costs of algorithms. We also evaluate the generalization power of classifiers via holdout test sets. Also, in the appendix F, we measure the prediction performance on training sets together with various other aspects of our framework.³

Datasets. We have used three stationary datasets in our experiments that are standard binary classification datasets taken from UCI repository [Dua and Graff, 2017]. Furthermore, we conduct experiments on the ProPublica recidivism (Compas) dataset [Larson *et al.*, 2016] and the Fair Isaac (Fico) credit risk dataset [FICO *et al.*, 2018] as in [Hu *et al.*, 2019]. In Compas dataset, we predict the individuals arrested after two years of release, and in Fico we predict if an individual will default on a loan. For concept drifting experiments, we adopt the non-stationary Stagger dataset [Widmer and Kubat, 1996; López Lobo, 2020], where each data has three nominal attributes and the target concept will change abruptly at some point. For extensions to continuous features (as well as for feature selection in the appendix), we use Prima Indians Diabetes Dataset [Smith *et al.*, 1988], Breast Cancer Wisconsin Dataset [Street *et al.*, 1999] and Fetal Health Dataset [Ayres-de Campos *et al.*, 2000].

³Source code of UFODT is available here on GitHub.

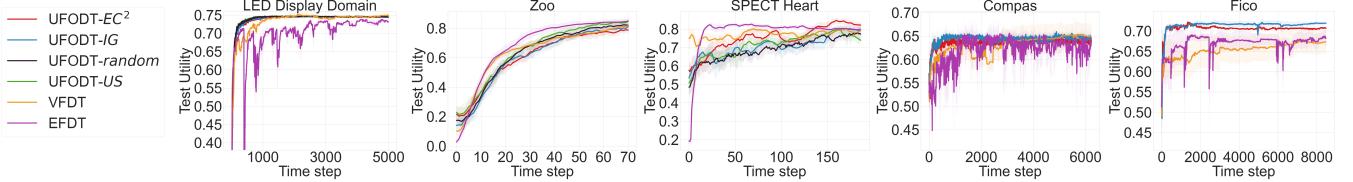


Figure 1: Test utilities during training: UFODT reaches test utilities comparable with those from VFDT and EFDT but with significantly lower costs. UFODT performs even better than VFDT and EFDT on LED, Heart and Fico datasets.

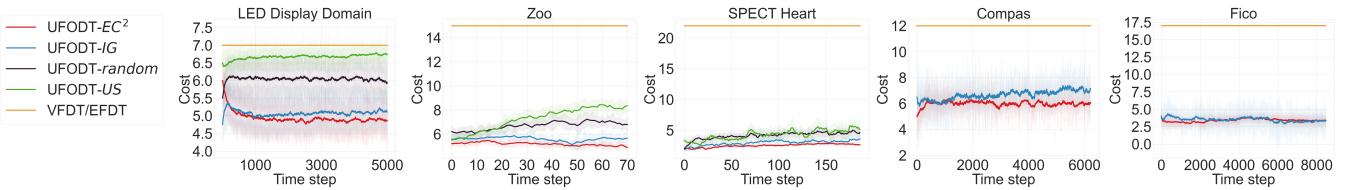


Figure 2: Querying costs during training: UFODT-EC² yields the lowest cost during training for all datasets. The cost of our framework is significantly lower than the VFDT and EFDT algorithms which require all feature values during training steps.

Algorithms. The VFDT algorithm [Domingos and Hulten, 2000] is used as a classical baseline ODT model. We also compare our method with the EFDT algorithm [Manapragada *et al.*, 2018]. Within our proposed UFODT framework, we use four different information acquisition functions. The first one is EC² which is proved to have near-optimal cost in offline planning. In addition to EC², we use Information Gain (IG) which selects the feature that maximizes the reduction of entropy in labels. Moreover, we conduct experiments with Uncertainty Sampling (US) which finds the feature causing the highest reduction in entropy of hypotheses. We also use random feature selection which randomizes the order of querying features.

6.1 Experiments on Stationary Datasets

Figure 1 shows the average utility achieved in each training time step by different methods over a holdout test set. To compute the test utility for our UFODT framework (with different objectives) we use the current estimation of the parameters of the conditional distributions (i.e., the estimated θ at time t) to obtain the test predictions. For VFDT and EFDT, we use the last version of the tree at time t . If a dataset is balanced we use accuracy as the utility, whereas we use f-measure for imbalanced datasets. Figure 2 shows the average cost (i.e., the number of features queried) in each training time step for different algorithms⁴. For all datasets, we observe that our UFODT framework reaches a very competitive test utility during training with a much lower cost. UFODT-EC² yields the lowest cost among other information acquisition functions which is compatible with the theoretical results. As discussed earlier, VFDT and EFDT are costly (i.e., require access to all features) and not fully online (labels are known in advance during training), while our proposed

framework is cost-effective and fully online. We observe that, with exceptionally lower cost, our framework with different algorithms still reaches comparable or even better test utilities, compared with VFDT and EFDT. The number of sampled hypotheses in UFODT are 95, 161, 34, 500 and 50 for LED Display Domain, Zoo, SPECT Heart, Compas and Fico respectively. In Appendix F, we investigate other aspects of our UFODT framework using these datasets.

6.2 Extension to Continuous Features

We here experimentally investigate the effectiveness of our UFODT framework for non-binary real-valued features (see Section 4.3). We conduct our experiments on three different classification datasets: i) Prima Indians Diabetes dataset (D) which has several medical predictor variables and two classes indicating the onset of diabetes mellitus, ii) Breast Cancer Wisconsin dataset (B), and iii) Fetal Health dataset (F). We employ the two methods discussed in Section 4.3 to handle continuous features, i.e., exhaustive search over thresholds and learning best the thresholds via Algorithm 5 (shown by UFODT-criterion-Exp3, e.g., UFODT-IG-Exp3). The number of sampled hypotheses in UFODT are 120, 500 and 500 for Diabetes, Breast Cancer and Fetal Health respectively. We use $\eta = 0.01$ in Algorithm 5.

In Figure 3(a), we illustrate the average cost (number of queried features) in each time step of training. We compare the UFODT framework with EFDT and VFDT. In Figure 3(b), we show the test utility during the training process (similar to Figure 1). We repeat these experiments with 5 different random seeds and report the averaged results together with one-standard error. The results demonstrate again that our framework (UFODT-EC² and UFODT-IG) has competitive prediction accuracy compared to EFDT and VFDT while having an exceptionally lower feature acquisition cost. UFODT-EC² generally has the lowest cost for feature querying; UFODT-IG yields a slightly higher cost than UFODT-EC² but reaches the same or even better test utilities than EFDT and VFDT algorithms. Moreover, we observe that

⁴To make the results more clear, we do not show the results of UFODT-random and UFODT-US for Fico and Compas datasets as they have higher query costs and generally lower test performances compared to UFODT-IG and UFODT-EC².

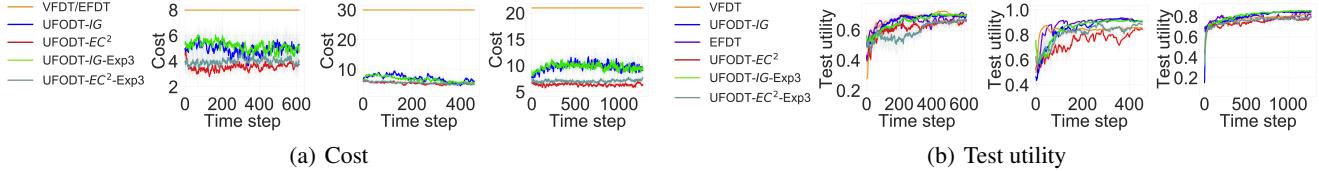


Figure 3: The average cost (3(a)) and average test utility (3(b)) during training for data sets (left to right) D, B and F. Our framework enjoys significantly lower cost while maintaining competitive prediction accuracy. Using Algorithm 5 for learning thresholds yields competitive results with those of exhaustive search, but with significantly lower running time.

the incurred querying costs and test utilities achieved by the threshold learning algorithm (Algorithm 5) is competitive with the exhaustive search method with a significantly lower running time. For instance, in case of UFODT-EC², we see that UFODT-EC²-Exp3 yields even better test utility with slightly higher cost.

6.3 Experiments with Concept Drift Dataset

Method	UFODT-EC ² (Adaptive)	UFODT-IG (Adaptive)	UFODT-US (Adaptive)	EFDT
Cost ↓	343.3 ± 11.0	350.6 ± 5.1	477.0 ± 13.9	720.0

Table 2: Average feature querying costs for Stagger dataset where UFODT-based methods incur significantly lower costs.

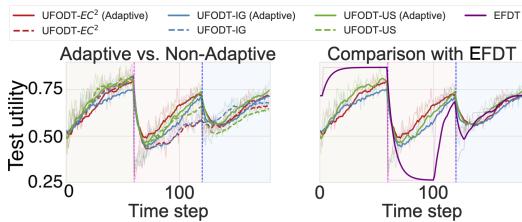


Figure 4: Time step vs. test utility on Stagger dataset. Each shaded area corresponds to one concept; the vertical dashed line shows when the drift happens.

In this section, we demonstrate the effectiveness and flexibility of our framework under the concept drift setting. Here, the non-stationary nature of the online data imposes extra difficulty for online decision tree problems. To handle the concept drift, we adopt non-stationary posterior sampling (Algorithm 6). We compare our proposed algorithm to EFDT, which is the SOTA baseline for solving concept drifting problem in online decision tree learning. To simulate the concept drift scenarios, we adopt the Stagger dataset. In this dataset, there are in total two concept drifting that happens abruptly at time steps 60 and 120. We repeat each experiment with 10 random seeds and report the averaged results along with one-standard error.

Test utility vs. time step. In Figure 4, we report the results of UFODT-EC², UFODT-IG and UFODT-US with both standard posterior sampling and non-stationary posterior sampling (denoted with *Adaptive*). For all of our methods, we

adopt the uniform prior. We can observe that all the three methods with non-stationary posterior sampling can adapt to the abrupt concept drift much faster, and also achieve higher test utility. We also compare these three methods against EFDT in the right part of Figure 4. Though, initially, EFDT can achieve higher utility than our methods, it has a big drop in utility after both the first and second concept drifting. This demonstrates advantages of our methods in quickly adapting to new concepts over EFDT. In addition, we also report the averaged total number of feature queries as a cost measure in Table 2. We observe that our approaches require significantly fewer feature queries (or lower cost) but still achieve higher utility than EFDT.

7 Conclusion

We make the first concrete step towards learning decision trees online with incorporating feature acquisition cost. Within the proposed framework, to learn efficiently with less time, we utilize a posterior sampling scheme; to predict efficiently with lower cost, we employ various information acquisition objectives including a surrogate objective function on utility of features, enabling near-optimal feature acquisition cost with competitive prediction accuracy. Our framework also provides several novel algorithmic contributions including a simple and flexible solution to the concept drift problem, an extension to efficiently handle real-valued features and a computationally efficient online feature selection scheme. In general, our work opens a new and practical direction of online decision tree learning on cost-sensitive applications, and we demonstrate the great potential of active information acquisition strategies in such applications.

Acknowledgements

The work of Arman Rahbar and Morteza Haghir Chehreghani was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. We would like to thank Chaoqi Wang and the anonymous reviewers for their constructive comments.

References

- [Agrawal and Jia, 2017] Shipra Agrawal and Randy Jia. Posterior sampling for reinforcement learning: worst-case regret bounds. In *Advances in Neural Information Processing Systems*, pages 1184–1194, 2017.

- [Auer *et al.*, 2002] Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. The nonstochastic multiarmed bandit problem. *SIAM journal on computing*, 32(1):48–77, 2002.
- [Ayres-de Campos *et al.*, 2000] Diogo Ayres-de Campos, Joao Bernardes, Antonio Garrido, Joaquim Marques-de sa, and Luis Pereira-leite. Sisporto 2.0: A program for automated analysis of cardiotocograms. *The Journal of maternal-fetal medicine*, 9:311–8, 09 2000.
- [Beyer *et al.*, 2020] Christian Beyer, Maik Büttner, Vishnu Unnikrishnan, Miro Schleicher, Eirini Ntoutsi, and Myra Spiliopoulou. Active feature acquisition on data streams under feature drift. *Annals of Telecommunications*, 75(9):597–611, 2020.
- [Bilgic and Getoor, 2007] Mustafa Bilgic and Lise Getoor. Voila: Efficient feature-value acquisition for classification. In *Proceedings of the national conference on artificial intelligence*, volume 22, page 1225. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007.
- [Blake and Ntoutsi, 2018] Christopher Blake and Eirini Ntoutsi. Reinforcement learning based decision tree induction over data streams with concept drifts. In *2018 IEEE International Conference on Big Knowledge (ICBK)*, pages 328–335. IEEE, 2018.
- [Chen *et al.*, 2017a] Yuxin Chen, Hamed Hassani, and Andreas Krause. Near-optimal bayesian active learning with correlated and noisy tests. In *Artificial Intelligence and Statistics*, pages 223–231. PMLR, 2017.
- [Chen *et al.*, 2017b] Yuxin Chen, Jean-Michel Renders, Morteza Haghir Chehreghani, and Andreas Krause. Efficient online learning for optimizing value of information: Theory and application to interactive troubleshooting. In *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence, UAI 2017, Sydney, Australia, August 11-15, 2017*. AUAI Press, 2017.
- [Das *et al.*, 2019] Ariyam Das, Jin Wang, Sahil M Gandhi, Jae Lee, Wei Wang, and Carlo Zaniolo. Learn smart with less: Building better online decision trees with fewer training examples. In *IJCAI*, pages 2209–2215, 2019.
- [Dasgupta, 2005] Sanjoy Dasgupta. Analysis of a greedy active learning strategy. *Advances in neural information processing systems*, 17:337–344, 2005.
- [Domingos and Hulten, 2000] Pedro Domingos and Geoff Hulten. Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 71–80, 2000.
- [Dua and Graff, 2017] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.
- [Fan and Ming, 2021] Ying Fan and Yifei Ming. Model-based reinforcement learning for continuous control with posterior sampling. In *International Conference on Machine Learning*, pages 3078–3087. PMLR, 2021.
- [Féraud *et al.*, 2016] Raphaël Féraud, Robin Allesiardo, Tanguy Urvoy, and Fabrice Clérot. Random forest for the contextual bandit problem. In *Artificial intelligence and statistics*, pages 93–101. PMLR, 2016.
- [FICO *et al.*, 2018] FICO, Google, Imperial College London, MIT, University of Oxford, UC Irvine, and UC Berkeley. Explainable machine learning challenge, 2018.
- [Garlapati *et al.*, 2015] Abhinav Garlapati, Aditi Raghunathan, Vaishnavh Nagarajan, and Balaraman Ravindran. A reinforcement learning approach to online learning of decision trees. *arXiv preprint arXiv:1507.06923*, 2015.
- [Golovin and Krause, 2011] Daniel Golovin and Andreas Krause. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *Journal of Artificial Intelligence Research*, 42:427–486, 2011.
- [Golovin *et al.*, 2010] Daniel Golovin, Andreas Krause, and Debajyoti Ray. Near-optimal bayesian active learning with noisy observations. In *Proceedings of NIPS, NIPS’10*, page 766–774, Red Hook, NY, USA, 2010. Curran Associates Inc.
- [Howard, 1966] Ronald A Howard. Information value theory. *IEEE Transactions on systems science and cybernetics*, 2(1):22–26, 1966.
- [Hu *et al.*, 2019] Xiyang Hu, Cynthia Rudin, and Margo Seltzer. Optimal sparse decision trees. *Advances in Neural Information Processing Systems*, 32, 2019.
- [Hulten *et al.*, 2001] Geoff Hulten, Laurie Spencer, and Pedro Domingos. Mining time-changing data streams. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 97–106, 2001.
- [Javdani *et al.*, 2014] Shervin Javdani, Yuxin Chen, Amin Karbasi, Andreas Krause, Drew Bagnell, and Siddhartha Srinivasa. Near optimal bayesian active learning for decision making. In *Artificial Intelligence and Statistics*, pages 430–438. PMLR, 2014.
- [Jiang *et al.*, 2013] Feng Jiang, Yuefei Sui, and Cungen Cao. An incremental decision tree algorithm based on rough sets and its application in intrusion detection. *Artificial Intelligence Review*, 40(4):517–530, 2013.
- [Kapoor and Horvitz, 2009] Ashish Kapoor and Eric Horvitz. Breaking boundaries: Active information acquisition across learning and diagnosis. *Advances in neural information processing systems*, 2009.
- [Larson *et al.*, 2016] J. Larson, S. Mattu, L. Kirchner, and J. Angwin. How we analyzed the compas recidivism algorithm. *SIAM journal on computing*, 2016.
- [Lu *et al.*, 2021] Xiuyuan Lu, Benjamin Van Roy, Vikrant Dwaracherla, Morteza Ibrahimi, Ian Osband, and Zheng Wen. Reinforcement learning, bit by bit. *arXiv preprint arXiv:2103.04047*, 2021.
- [López Lobo, 2020] Jesús López Lobo. Synthetic datasets for concept drift detection purposes, 2020.

- [Manapragada *et al.*, 2018] Chaitanya Manapragada, Geoffrey I Webb, and Mahsa Salehi. Extremely fast decision tree. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1953–1962, 2018.
- [Osband and Van Roy, 2017] Ian Osband and Benjamin Van Roy. Why is posterior sampling better than optimism for reinforcement learning? In *International conference on machine learning*, pages 2701–2710. PMLR, 2017.
- [Osband *et al.*, 2013] Ian Osband, Daniel Russo, and Benjamin Van Roy. (more) efficient reinforcement learning via posterior sampling. *arXiv preprint arXiv:1306.0940*, 2013.
- [Podgorelec *et al.*, 2002] Vili Podgorelec, Peter Kokol, Bruno Stiglic, and Ivan Rozman. Decision trees: an overview and their use in medicine. *Journal of medical systems*, 26(5):445–463, 2002.
- [Rozaki, 2015] Eleni Rozaki. Design and implementation for automated network troubleshooting using data mining. *International Journal of Data Mining & Knowledge Management Proces*, 5(3), 2015.
- [Russo and Van Roy, 2016] Daniel Russo and Benjamin Van Roy. An information-theoretic analysis of thompson sampling. *The Journal of Machine Learning Research*, 17(1):2442–2471, 2016.
- [Russo *et al.*, 2017] Daniel Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, and Zheng Wen. A tutorial on thompson sampling. *arXiv preprint arXiv:1707.02038*, 2017.
- [Shim *et al.*, 2018] Hajin Shim, Sung Ju Hwang, and Eunho Yang. Joint active feature acquisition and classification with variable-size set encoding. *Advances in neural information processing systems*, 31:1368–1378, 2018.
- [Smith *et al.*, 1988] Jack Smith, J. Everhart, W. Dickson, W. Knowler, and Richard Johannes. Using the adap learning algorithm to forcast the onset of diabetes mellitus. *Proceedings - Annual Symposium on Computer Applications in Medical Care*, 10, 11 1988.
- [Street *et al.*, 1999] Nick Street, William Wolberg, and O Mangasarian. Nuclear feature extraction for breast tumor diagnosis. *Proc. Soc. Photo-Opt. Inst. Eng.*, 1993, 01 1999.
- [Thompson, 1933] William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.
- [Vapnik, 1992] Vladimir Vapnik. Principles of risk minimization for learning theory. In *Advances in neural information processing systems*, pages 831–838, 1992.
- [Widmer and Kubat, 1996] Gerhard Widmer and Miroslav Kubat. Learning in the presence of concept drift and hidden contexts. *Machine learning*, 23(1):69–101, 1996.