Thesis for The Degree of Licentiate of Engineering

Towards Better Representation Learning in the Absence of Sufficient Supervision

Arman Rahbar

Division of Data Science and AI
Department of Computer Science and Engineering
Chalmers University of Technology | University of Gothenburg
Gothenburg, Sweden, 2022

Towards Better Representation Learning in the Absence of Sufficient Supervision

Arman Rahbar

ISSN 1652-876X

Department of Computer Science and Engineering
Division of Data Science and AI
Chalmers University of Technology | University of Gothenburg
SE-412 96 Göteborg,
Sweden
Phone: +46(0)31 772 1000

Printed by Chalmers Digitaltryck,
Gothenburg, Sweden 2022.

To my Anahita, Mahmoud, Sakineh and Erfan

تقدیم به آناهیتا همسرم، پدرم محمود، مادرم سکینه و برادرم عرفان

Towards Better Representation Learning in the Absence of Sufficient Supervision

Arman Rahbar

Department of Computer Science and Engineering
Chalmers University of Technology | University of Gothenburg

Abstract

We focus on the problem of learning representations from data in the situation where we do not have access to sufficient supervision such as labels or feature values. This situation can be present in many real-world machine learning tasks. We approach this problem from different perspectives summarized as follows.

First, we assume there is some knowledge already available from a different but related task or model, and aim at using that knowledge in our task of interest. We perform this form of *knowledge transfer* in two different but related ways: i. using the knowledge available in kernel embeddings to improve the training properties of a neural network, and ii. transferring the knowledge available in a large model to a smaller one. In the former case, we use the recent theoretical results on training of neural networks and a multiple kernel learning algorithm to achieve a high performance in terms of both optimization and generalization in a neural network.

Next, we tackle the problem of learning appropriate data representations from an online learning point of view in which one should learn incrementally from an incoming source of data. We assume that the whole feature set of a data input is not always available, and seek a way to learn efficiently from a smaller set of feature values. We propose a novel online learning framework which builds a decision tree from a data stream, and yields highly accurate predictions, competitive with classical online decision tree learners but with a significantly lower cost.

Keywords

Representation learning, Supervision, Neural network, Knowledge transfer, Kernel embedding, Online learning, Feature acquisition, Decision tree

List of Publications

Appended publications

This thesis is based on the following papers:

[**Paper I**] **Arman Rahbar**, Emilio Jorge, Devdatt Dubhashi, and Morteza Haghir Chehreghani, *Do Kernel and Neural Embeddings Help in Training and Generalization?*
Neural Processing Letters, 2022.
[Contributed in: design of the study, empirical evaluation and analysis, writing the manuscript]

[**Paper II**] Ashkan Panahi, **Arman Rahbar**, Chiranjib Bhattacharyya, Devdatt Dubhashi, and Morteza Haghir Chehreghani, *Analysis of Knowledge Transfer in Kernel Regime*
In Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM '22) 2022.
[Contributed in: empirical evaluation and writing the manuscript]

[**Paper III**] **Arman Rahbar**, Ziyu Ye, Chaoqi Wang, Yuxin Chen, Morteza Haghir Chehreghani, *Efficient Online Decision Tree Learning by Utility of Features*
To be submitted.
[Contributed in: design of the study, empirical evaluation and analysis, writing the manuscript]

Acknowledgment

First and foremost, I would like to thank my main supervisor Morteza Haghir Chehreghani for all his support, and his invaluable guidance. I really feel blessed to be part of your research group. I am also very grateful to my co-supervisor Hossein Azizpour, as well as my examiner Graham Kemp.

In the last few years, I have enjoyed collaborating with exceptional people. Thus, I want to thank Devdatt, Ashkan, Emilio, Chiranjib, Ziyu, Chaoqi, and Yuxin for their help and support.

During these years of my PhD life, I am really happy to have met and worked with great people. Many thanks to Niklas, Fazeleh, Mehrdad, Emil, Amir, Mehrzad, Shirin, Hannah, Mohammad, Hamid, Mehdi, Siavash, Ebrahim, Tobias, Linus, David, Adam, Jonah, Peter, Adel, Christos, Victor, Ola, Hannes, Birgit, Divya, Hampus, Lovisa, Fredrik, Richard, Simon, Newton, Christopher, Anton, Nasser, Mena, Tobias, Simon, Vladimir, Markus, Alexander, Selpi, Lena, Kolbjörn, Juan, and Dag.

I also want to thank my dear friends Yoosof, Seyed Mohammad and Firooz. Thank you for keeping me happy, and providing pleasant distractions from work.

My deepest gratitude would, of course, go to my parents and parents in law. Thank you Mahmoud, Sakineh, Dariush, and Nazi for all your support and sacrifices. I would also like to express my gratitude to my little brothers Erfan and Danial. Thank you for always supporting me, and making me laugh.

Last, but not least, I have to thank my incredible wife Anahita. I could not have undertaken this journey without your endless love and belief in me.

This research has been supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation, project "Under-supervised Representation Learning" grant nr. 37200022.

Arman Rahbar
Gothenburg, Sweden
November, 2022

Contents

Part I

Introductory Chapters

Chapter 1

Introduction

The success of machine learning methods including deep neural networks and linear models like Support Vector Machines (SVMs) is highly dependant on the type of data features (also referred to as data representation) they use. A feature is a single descriptive aspect of a data point. For instance, assume we would like to predict if an email is spam or not (known as spam detection). Then a potential set of features is the frequencies of certain terms in the email body. Traditionally, in order to find relevant features from data, machine learning practitioners used to get help from domain experts. This classical way of finding features is called feature/representation *engineering*. Feature engineering can be highly expensive in both the required time and human effort. To address this challenge, there has been a lot of effort to substitute the feature engineering process with *feature/representation learning* [1] which refers to a set of methods that automatically find representations from data. *Feature selection* (i.e., selecting a subset of available features) can also be viewed as a special type of representation learning.

Representation learning can be conducted in both *supervised* and *unsupervised* ways. Similar to any other machine learning procedure, in supervised representation learning, we aim at learning a proper representation from data inputs together with their target variables (e.g., labels), whereas in the unsupervised setting the representations are learned without the target data.

Principal Component Analysis (PCA) [2], [3] and Independent Component Analysis (ICA) [4] are common methods used for unsupervised learning of representations. Unsupervised representation learning can also be implemented using neural networks. One of the most popular unsupervised neural networks for representation learning is Autoencoder [5]. Deep neural networks are also widely used for supervised representation learning, and are applicable in many applications such as Computer Vision, Speech Recognition, etc.

Representation learning can also be of a very high importance in the online learning setting where data points arrive in the form of a data stream and should be processed rapidly. In this setting, we are generally interested in feature selection. Most of the works in online feature selection (e.g., [6] and [7]) focus on the streaming features setting. Streaming features mean that the

features arrive sequentially, and we have a fixed number of training examples. On the other hand, works like [8] and [9] consider feature selection from a sequence of training examples. For instance, the work in [8] uses an online perceptron algorithm, and selects the features with highest weights in each time step.

In this thesis, we consider a specific setting for representation learning. In particular, we aim at learning proper representations from data when sufficient supervision is not available. We tackle this problem in a few different ways. One possible approach is using knowledge transfer. Inspired by the recent theoretical results in [10], we investigate knowledge transfer in two different settings. First, we consider transferring the knowledge available from kernels and neural embeddings to improve the training and generalization properties of neural networks. We study this setting in the appended [Paper I]. The second knowledge transfer approach we use for representation learning is based on the Student-Teacher scheme. In this scheme, the knowledge learned in a large model (the teacher) is distilled into a smaller model (the student). In the appended [Paper II], we introduce a new framework for knowledge transfer in neural networks and analyze it.

Finally, we consider interactive learning methods. Specifically, we aim at finding the most valuable features to learn from data streams. In the appended [Paper III], we propose a novel and efficient framework for learning an online decision tree. Our online decision tree learning method has several advantages over previous ones. This framework can be effective in the online learning settings where it is not always possible to access all feature values in the incoming data points.

Chapter 2

Background

In this chapter, we will briefly introduce some of the concepts and methods used throughout this thesis.

2.1 Neural networks and kernels

Neural networks have empirically shown to be very successful in different machine learning tasks from classification and regression to object detection [11] and recommender systems [12]. Despite their huge success, neural networks have not been theoretically well understood. However, recently there has been a line of work which tries to theoretically analyze the training behavior and generalization properties of neural networks. In this thesis, we exploit these results together with kernel methods to achieve a higher performance in terms of both generalization and optimization in a neural network. In what follows, we first provide an overview of a recent theoretical result that sheds new light on explaining the training behavior of neural networks. We continue with an introduction to kernels, and describe how we benefit from them to improve the training procedure in a neural network.

2.1.1 Theoretical results for optimization and generalization in neural networks

The work in [13], with cleverly designed experiments, shows that the traditional theory (e.g., VC-dimension and Rademacher complexity) may fail to explain the very good generalization of neural networks. They showed that a deep neural network can easily fit a random set of labels (instead of true labels) meaning that the network can reach zero error on training data even when using random labels, but the generalization error would definitely go high with this kind of randomization. This observation implies that the capacity of a deep neural network is enough to memorize the whole data set. The slight difference when training on random labels is that the training time is higher compared to that of true labels. They argue that the results of their experiments prevent the classic theory to explain the generalization in neural networks. Moreover,

they found out that the role of regularization in the generalization error of neural networks is not well explained by the traditional theory.

The study in [10] theoretically analyzes both optimization and generalization properties of neural networks. In particular, inspired by the findings in [13], they answer two fundamental questions about neural networks:

- Why do true labels result in a faster rate of convergence when using gradient descent than random labels?
- What is the complexity measure that differentiates the true set of labels and random ones?

The authors of [10] consider a neural network with one hidden layer and the ReLU activation function. Specifically, they analyze the following neural network:

$$f(\mathbf{x}) = \sum_{k=1}^{m} \frac{a_k}{\sqrt{m}} \sigma(\mathbf{w}_k^T \mathbf{x}), \tag{2.1}$$

where $\{\mathbf{w}_k\}_{k=1}^m$ are the weights of the hidden layer which has m units, $\sigma(.)$ is the activation function, $\{a_k\}_{k=1}^m$ are the weights of output layer, and are fixed. The network is trained with randomly initialized gradient descent with a quadratic loss over a data set $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$. Precisely, the optimization problem is:

$$\min_{\{\mathbf{w}_k\}_{k=1}^m} \sum_i (y_i - f(\mathbf{x}_i))^2. \tag{2.2}$$

The analysis is based on a *Gram matrix* defined over a set of data points $\{\mathbf{x_i}\}_{i=1}^n$:

$$\begin{aligned} \mathbf{H}_{i,j}^{\infty} &:= E_{\mathbf{w}\sim\mathcal{N}(0,\mathcal{I})}[\mathbf{x}_i^T \mathbf{x}_j 1\{\mathbf{w}^T \mathbf{x}_i \geq 0, \mathbf{w}^T \mathbf{x}_j \geq 0\}] \\ &= \frac{\mathbf{x}_i^T \mathbf{x}_j (\pi - \arccos(\mathbf{x}_i^T \mathbf{x}_j))}{2\pi}. \end{aligned} \tag{2.3}$$

The results in [10] answer the above mentioned questions by a spectral analysis of $\mathbf{H}^{\infty}$. Their analysis enables us to distinguish the convergence rates for different sets of labels. In particular, they found out that to achieve a faster rate of convergence, we want the projections of the label vector on top eigenvectors of $\mathbf{H}^{\infty}$ to be bigger.

In addition, they provide a bound on the generalization error which is controlled by:

$$\sqrt{\frac{2\mathbf{y}^T (\mathbf{H}^{\infty})^{-1} \mathbf{y}}{n}}, \tag{2.4}$$

where $\mathbf{y}$ is the label vector. More details can be found in appended [paper II].

In this thesis, we experimentally investigate the implications of the theoretical results mentioned above to deeper networks with the help of *(kernel) feature representations*. In the following, we introduce the notion of kernel (adapted from [14]), and explain how one can use them to produce new features for data points.

Figure 2.1: Feature extraction

2.1.2 Kernels

In a general learning setup, there is no assumption about the space $\mathcal{X}$ where input comes from. However, in many learning algorithms like Support Vector Machine (SVM) or Lasso/Ridge regression it is assumed that the input features come from $\mathcal{X} = \mathbb{R}^d$ for some integer $d > 0$. To be specific, for example for SVM, the output of the learning algorithm is an element of the following *hypotheses space* (here a collection of functions):

$$\mathcal{H} = \{\mathbf{x} \mapsto \mathbf{w}^T\mathbf{x} + b | \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}\} \tag{2.5}$$

So to make predictions on data that are not originally in $\mathbb{R}^d$ we need to somehow map them to $\mathbb{R}^d$ for some d. Therefore, in a sense, we wish to do *feature extraction* which is schematically shown in Figure 2.1.

Formally, feature extraction introduces a feature map $\Phi : \mathcal{X} \to \mathbb{R}^d$. Then the hypotheses space becomes:

$$\mathcal{H} = \{\mathbf{x} \mapsto \mathbf{w}^T\Phi(\mathbf{x}) + b | \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}\} \tag{2.6}$$

Here, there is no assumption on the space $\mathcal{X}$. For example $\mathcal{X}$ may be a subset of $\mathbb{R}^2$, but we might need other types of features to improve the learning procedure (imagine the case where with original features we can not separate two classes with a straight line).

But, what if we need a very large number of features? In that case, the memory and computational cost required for learning would be very high. In this situation *kernel methods* come into play. We say a learning method is *kernelized* if the inputs never appear outside an inner product like $\Phi(\mathbf{x}_i)^T\Phi(\mathbf{x}_j)$ for some $\mathbf{x}_i, \mathbf{x}_i \in \mathcal{X}$. The kernel function corresponding to a feature map Φ is defined as:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i)^T\Phi(\mathbf{x}_j) \tag{2.7}$$

The matrix K with $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ is called the *kernel matrix* (or sometimes Gram matrix). The reason for introducing this kernel function is that it is possible to compute $k(\mathbf{x}_i, \mathbf{x}_j)$ without a direct access to the feature map Φ. This is a significant computational advantage when we have a large number of features. Probably the most well-known kernelized method is SVM, and some important kernel functions are the Polynomial kernel, Gaussian kernel and Quadratic kernel.

For a kernelized method, the computational cost depends on the number of data points (not on the number features) once we have computed $k(\mathbf{x}_i, \mathbf{x}_j)$

for all $(\mathbf{x}_i, \mathbf{x}_j)$ pairs. This is known as the *kernel trick*. It is worth noting that some kernels (including the Gaussian kernel) correspond to an infinite number of features.

With this introduction to kernels, we now move on to the specific way we use kernels in this thesis. Actually, we do *not* use the kernel trick, and, conversely, we use kernels to produce new features for data points, and employ those features directly to improve the optimization (and generalization) behavior of a neural network.

As illustrated in section 2.1.1, the optimization and generalization properties of a neural network is related to the data representation through the Gram matrix $\mathbf{H}^{\infty}$. So, we provide better data representations to the network to improve both optimization and generalization. These better representations are obtained from three specific kernels: i. the Gaussian kernel, ii. kernels that try to mimic the representations produced by neural networks, and iii. an *optimal* kernel that is designed to make the data representations align with the label vector.

The optimal kernel mentioned above comes from the notion of *kernel-target alignment* [15], [16]. Since in [10] it is suggested that we need representations making the label vector align well with the top eigenvectors of the Gram matrix, we might use the kernel-target alignment to produce such feature representations.

Finally, since kernels like Gaussian kernel (and other kernels that we use in this thesis) have infinite number of features[1], we need to somehow approximate the feature representations corresponding to a kernel. Specifically, we use two main approximation methods which are briefly summarized below.

The first kernel feature approximation method is called *random Fourier features* (RFF) [17] which constructs an explicit feature map. The constructed feature map has a dimension much lower than the number of observations, and at the same time the resulting inner product is close to the kernel function. The feature map produced by RFF is randomized.

The second approximation method that we use originates from a low-rank matrix approximation method called the Nyström method [18]. In the Nyström method, an $n \times n$ positive semi-definite matrix (e.g., a kernel matrix) is approximated using m available rows where $m < n$. This method can be used to find a matrix $\mathbf{\Phi}$ such that $\mathbf{\Phi}\mathbf{\Phi}^T = K$ where K is the kernel matrix. Then the rows of $\mathbf{\Phi}$ are approximated kernel feature representations.

2.2 Knowledge transfer

A well-known technique to improve the performance of machine learning systems is to use the knowledge we have already learned in other tasks or models. This technique is generally known as *knowledge transfer* (Figure 2.2). Knowledge transfer has been studied in many different forms and settings, and has shown great effectiveness in many machine learning tasks including but not limited to

[1]The features corresponding to a kernel are sometimes called *kernel embedding*.

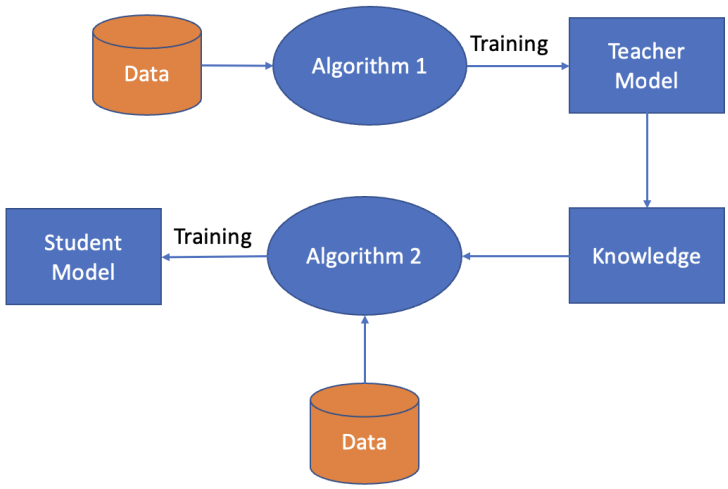

Figure 2.2: Knowledge transfer

computer vision [19], Natural Language Processing (NLP) [20] and recommender systems [21].

A very important type of knowledge transfer is that of *privileged information* which was introduced in [22]. In privileged information, there are two learning systems/models involved: a "student" and a "teacher". The teacher is most often a trained model, and the student will use the obtained information from the teacher during the training process. In particular, as discussed in [22], privileged information tries to imitate the notion of learning by humans so as to make the process of learning by examples more efficient. So, together with the data points and their labels (or any other target variables), the teacher provides some additional information.

More formally, in the classical learning setting, the learner is provided with a set of training data points in the form of pairs $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$. However, in the privileged information setting, a new piece of information $\mathbf{x}_i^*$ is also provided along with $(\mathbf{x}_i, y_i)$, and the training data points will be of the form $(\mathbf{x}_i, \mathbf{x}_i^*, y_i)$. The ultimate goal of privileged information setting is identical to that of classical learning: to learn a function f such that $f(\mathbf{x}_{new}) = y_{new}$ for a new example $(\mathbf{x}_{new}, y_{new})$. Note that the additional information is not available in the testing phase, so it can only be used for training.

In [23], the privileged information paradigm is extended to be used in training of neural networks. In this thesis, we introduce a knowledge transfer framework for neural networks which can be viewed as a special case of the one presented in [23]. We will analyze the introduced framework both theoretically and experimentally.

2.3 Online learning

In this thesis, we propose a novel framework for the problem of online decision tree learning. Online learning has received a significant attention in the past few years. The main reason behind this is the massive amount of data being produced constantly, where one would need an efficient method to exploit this amount of data. Previously, the main bottleneck for learning systems was the amount of data available to the learning algorithm. As a result, the learning procedure would be prone to overfitting due to small sample size, and a large fraction of computational power would remain unused.

Nowadays the situation is completely different, and the bottleneck has switched to the computational power. Therefore, we need efficient algorithms to exploit the huge amount of data produced, and thus prevent underfitting [24].

In what follows, we will introduce a general framework for online learning. After that, we will discuss the previous efforts related to online learning of decision trees, and finally we will briefly present the tools we use to develop our online decision tree learning framework.

2.3.1 An online learning framework

In online learning, the main goal is to learn from an incoming sequence of data points (or a data stream). In addition to learning, we most often also need to predict something about the current data point. This prediction might be a class label or some real-valued target variable (as in a regression problem). We will now continue with a formal definition of the online learning problem which is adapted from [25].

At each time (round) t, we receive data point $\mathbf{x}_t \in \mathcal{X}$ where $\mathcal{X}$ is called the instance domain. In real applications, the instance domain is very often a subset of $\mathbb{R}^d$ for some integer $d > 1$. The learning system is asked to output some prediction $\hat{y}_t$ for $\mathbf{x}_t$. Then the learning environment announces the true prediction y_t. The revealed prediction causes the learner to incur a loss $l(\hat{y}_t, y_t)$. Generally, $\hat{y}_t$ and y_t can come from different domains, but usually the domains are the same. The online learning paradigm is shown in Algorithm 1.

Algorithm 1 Online Learning

1: **for** t= 1, 2, ... **do**
2: receive data point $\mathbf{x_t}$
3: make a prediction $\hat{y}_t$
4: observe the true prediction y_t
5: suffer loss $l(\hat{y}_t, y_t)$
6: **end for**

The ultimate goal of an online learning algorithm is to minimize the cumulative loss it suffers from the start to the end of learning that is $\sum_{t=1}^{T} l(\hat{y}_t, y_t)$ where T is the total number of rounds. But it is possible that the environment (also sometimes called the *adversary*) makes the cumulative loss suffered by an

online learning algorithm arbitrarily large. For instance, if the environment sees the learner's prediction, it can just produce some y that makes the loss maximum.

In online learning literature, it is common to put some restrictions on the problem setting in order to solve the above stated problem. One usual restriction is to assume that the pairs $(\mathbf{x}_t, y_t)$ are generated from some *hypothesis* $h^* \in \mathcal{H}$ i.e., for all t we have $y_t = h^*(\mathbf{x}_t)$. This kind of simplification is often referred to as the *realizability* assumption. With the realizability assumption the online learning problem reduces to finding the true hypothesis h^*. Two famous online classification algorithms that operate in this setting are the *Consistent* and *Halving* algorithms. Both of these algorithms assume that the number of hypotheses in $\mathcal{H}$ is finite i.e. $|\mathcal{H}| < \infty$ (see chapter 21 of [26] for details).

If the realizability assumption is not desirable, we usually make the online learning algorithm to compete with the best hypotheses $h^* \in \mathcal{H}$. The notion of *regret* arises from this competition, and is defined by equation (2.8).

$$Regret_T(\mathcal{H}) = \sum_{t=1}^{T} l(\hat{y}_t, y_t) - \min_{h^* \in \mathcal{H}} \sum_{t=1}^{T} l(h^*(\mathbf{x}_t), y_t) \tag{2.8}$$

We want an online learning algorithm to incur the lowest regret possible with respect to $\mathcal{H}$. Specifically, we desire that the learning algorithm has a regret that is a sublinear function of time, meaning that $\lim_{T\to\infty} \frac{Regret_T(\mathcal{H})}{T} = 0$.

2.3.2 Online decision tree learning

Decision tree learning [27] is one of the most widely used learning algorithms that has been employed for different learning tasks including classification and regression. Like any other classification algorithm, a traditional (offline) decision tree classification algorithm receives a set of n training data points $\{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$ where $\mathbf{x}_i$ is the feature vector and y_i is the associated label. The goal of a decision tree classifier is to learn a function f in the form of a decision tree that is able to accurately predict the label y of a future feature vector $\mathbf{x}$. This means that ideally we want $f(\mathbf{x}) = y$. A decision tree is a tree whose nodes represent a test on a feature, and a leaf node corresponds to one of the possible classes. The tree is learned using a particular criterion. Specifically, we build the tree recursively by substituting the leaves with a test node. The feature for the new test node is the one that works best according to the criterion.

But with the procedure described above it is impossible to learn a decision tree in an online manner from a data stream as it is required that all training data points are available at the time of learning. To remedy this problem, there has been a line of work starting with the Very Fast Decision Tree (VFDT) algorithm [24], and continuing with Concept-adapting Very Fast Decision Tree (CVFDT) [28] and Extremely Fast Decision Tree (EFDT) [29]. Both CVFDT and EFDT are built based on VFDT but with minor improvements. In particular CVFDT tries to learn a decision tree in a concept-drifting[2]

[2]We say that a data stream has concept drift if the dependency of labels to features changes over time.

environment, and EFDT introduces a modification to VFDT that improves its performance and efficiency. We continue with an overview of the VFDT algorithm.

The main idea behind VFDT is to use the *Hoeffding bound* [30], [31] in order to choose the features to be tested at each node. The Hoeffding bound bounds the mean of a random variable based on its observations. Specifically, suppose we have made N independent observations of a random variable X with range R, and the empirical mean of these observations is $\bar{X}$. Based on the Hoeffding bound the true mean of X is at least $\bar{X} - \epsilon$ with probability $1 - \delta$, where:

$$\epsilon = \sqrt{\frac{R^2 \log(1/\delta)}{2N}} \tag{2.9}$$

VFDT uses this bound in a clever way. Suppose $M(X_i)$ is the criterion used to determine the test features at each node. The goal here is to be certain with a high probability that the feature chosen using N training examples is the same feature as the one chosen with the whole data set. Assuming that we want to maximize M, let X_1 and X_2 be the first and second best features in terms of M with N training samples for which we denote the averages with $\bar{M}(X_1)$ and $\bar{M}(X_2)$. Let $\Delta\bar{M} = \bar{M}(X_1) - \bar{M}(X_2) \geq 0$. Then with a desired δ, Hoeffding bound assures that X_1 is the best feature with probability $1 - \delta$ if $\Delta\bar{M} > \epsilon$. So for a node in the decision tree we need to collect training samples until $\Delta\bar{M} > \epsilon$. Then we can split that node with a high confidence using the best feature. Training samples arriving after that will be used to determine the best feature at a lower level. In [24], it is shown that VFDT learns trees that are asymptotically arbitrarily close (with a particular definition of closeness) to the ones learned by a traditional batch learner.

Despite its many advantages, VFDT has some limitations. The most important limitation of VFDT is that it does not comply with the online learning paradigm introduced in Section 2.3.1. Specifically, in Algorithm 1, we require the learner to observe the true prediction only *after* the prediction has been made, but in order to use the VFDT algorithm we need both data point and true prediction. So the VFDT algorithm and its successors are not completely online. Moreover, VFDT requires that *all* feature values of data points are present for learning. However, in real-world applications it is common that querying features is costly. For instance, conducting certain medical tests can be very expensive. In this thesis, we introduce a new framework for online learning of decision trees that tackles these limitations in the sense that i. it completely acts in accordance with the learning paradigm in Algorithm 1, and ii. it aims at making accurate predictions with a low cost.

2.3.3 Decision making with adaptive information acquisition

As mentioned above, our decision tree learning framework completely follows the online learning framework in Section 2.3.1. Our contribution mainly focuses on making the prediction (line 3 of Algorithm 1) with a low cost. To this

aim, we employ a line of work that designs an adaptive policy to identify an unknown target variable (e.g., a label). The identification of the target variable is done by sequentially querying tests and observing their outcomes. In particular, in this problem, one tries to find the best *hypothesis* with a sequence of observations. As an example, assume we would like to find the most accurate diagnosis through a set of costly medical observations. Then, different medical observations would constitute the set of tests, and the set of all different diagnoses is the hypotheses set. This problem which has been studied in different domains (e.g., [32], [33]) can be categorized into two settings: noiseless and noisy [34]. In the noiseless case, it is assumed that the outcome of a test is completely determined if the true hypothesis is known.

More formally, let $\mathcal{H} = \{h_1, h_2, \ldots, h_n\}$ be the set of all possible hypotheses. We also have B different tests available which are in a set $T = \{1, 2, \ldots, B\}$. Each of these B tests would result in a cost. Specifically, a test i will have cost $c(i)$ where $c : T \to \mathbb{R}^+$ is the cost function. The goal here is to find the true hypotheses by performing a sequence of tests with as low cost as possible. If H is the random variable for the hypotheses, and X_i is the random variable for the outcome of test i, then in the noiseless setting, the probability $P(X_1, \ldots, X_B | H = h)$ is deterministic [34] and otherwise we are in the noisy setting.

Finding a policy that always leads to the true hypothesis and has the lowest cost is computationally intractable. Here, a policy is a mapping from the current observations (of tests) to the remaining tests. More accurately, finding a policy which has cost at most $c^* \times o(\log n)$ is NP-hard, where c^* is the cost of optimal policy [34], [35].

There has been some heuristics proposed to solve the noiseless version of the problem stated above. The two most well known heuristics are *Information Gain* and *Generalized Binary Search*. Both of these algorithms are shown to have near-optimal cost in the noiseless setting of the problem. Unfortunately, in the noisy setting these algorithms may perform badly. To tackle this problem, in [34], another algorithm is proposed which uses the notion of adaptive submodularity [36], [37]. This algorithm is called *EC*^{2}. The details of EC^2 can be found in the appended [paper III]. In our proposed online decision tree learning framework, one can easily exploit each of the mentioned algorithms including Information Gain, Generalized Binary Search and EC^2. However, due to the provided theoretical guarantees for EC^2, we mainly focus on this algorithm.

Chapter 3

Summary of Included Papers

In this chapter, we provide a brief summary of the contributions and results from the papers included in this thesis.

3.1 Paper I

In [paper I], we experimentally investigate the implications of the theoretical results in [10] applied to deep neural networks, and illustrate how one can make use of kernel and neural embeddings to improve the generalization and optimization properties of neural networks. In particular, we design an extensive set of experiments with a careful choice of kernels which enable us to find better representations for data inputs.

A key part of this paper is the investigation of an optimized kernel embedding which is obtained from maximizing the kernel-target alignment. Specifically, we use the *multiple-kernel learning* algorithm proposed in [15] to learn a kernel with a high alignment with label vectors. The corresponding kernel is then used to find kernel embedding with the Nyström method. We show that using this kernel embedding reaches the best results in terms of both optimization and generalization. This result is consistent with the theoretical results in [10], and provides a practical way to improve the performance of neural networks.

Furthermore, we use kernel embeddings corresponding to neural networks which also help optimization and generalization. Our results can be a starting point to extend the recent theoretical results for (shallow) neural networks to deeper networks.

3.2 Paper II

In [Paper II], we consider the concept of knowledge transfer in neural networks, which is a practical way to improve the representations learned in small neural networks. Specifically, in [Paper II] we introduce an optimization framework

for neural knowledge transfer which uses the privileged knowledge from a large teacher network in the form of a regularization term. Then the training of the student is analyzed using the theoretical tools developed for understanding infinitely wide neural networks.

Our notion of knowledge transfer is general, but we focus on the case that privileged information is selected from the hidden neurons of a trained teacher neural network. We also provide extensive experiments for our knowledge transfer framework demonstrating its various aspects.

3.3 Paper III

[Paper III] introduces a novel framework for the construction of an online decision tree. This online decision tree is built in a cost-effective manner in the sense that it queries a low number of features. This cost-effectiveness is not considered in the classical algorithms for online decision tree learning (e.g., VFDT). In addition, our framework is truly online, meaning that we observe the target variable (i.e., label) only *after* making the prediction. This condition is also not met in the VFDT method (and its successors).

In our online learning framework, we employ the well-known posterior (Thompson) sampling [38] policy to trade off between exploration and exploitation. In simple words, in each online learning session, we sample an environment, and find the optimal prediction with low cost in the sampled environment. This is done through using efficient algorithms for low-cost hypothesis/decision region identification. At the end of the session, we update our knowledge about the environment which is in the form of a posterior distribution.

Using the theoretical results for Thompson sampling we provide regret bounds for our framework and evaluate its effectiveness through extensive experiments. The experiments show that our online decision tree learner achieves a high level of accuracy with a much lower cost compared to baseline models. Moreover, using Thompson sampling provides us with a simple and elegant way to handle a concept-drifting situation in online environments. This property is also verified in our experiments.

Chapter 4

Concluding Remarks

In this chapter, we provide brief discussions about the contributions of this thesis, and point to some possible future directions.

In this thesis we tackled the problem of learning appropriate representations when we do not have access to enough supervision. We approached this problem through different scenarios. We first considered the knowledge transfer setting in which we transfer the knowledge learned in other tasks to the task of interest. The knowledge transfer scenario was studied with two different but related approaches:

- Transferring the knowledge available in kernels to improve the representations learned in neural networks.
- Transferring the knowledge from a large pre-trained teacher model to a smaller student model which makes the student training more efficient.

Now we continue with an overview of the contributions provided in each of the items above:

We empirically studied the implications of the theoretical results in [10] to deeper networks. This was achieved by viewing the layers before the last layer as constructing better representations of data. We studied various kernel and neural embeddings, and experimentally showed that these representations can benefit both generalization and optimization properties of a neural network. Our experiments involve various benchmark data sets including CIFAR-10, LSUN [39], and MNIST[40].

Furthermore, we addressed knowledge transfer in neural networks in the student-teacher scheme. We considered the general concept of privileged information (reflected in our optimization framework), and provided insights for related paradigms such as knowledge distillation. We also conducted extensive numerical studies which confirms the effectiveness of the student-teacher scheme in efficient training of small neural networks, and their generalization power.

Next, we situated ourselves in an online learning environment where we would like to learn appropriate representations of data inputs which arrive in an incremental manner as a data stream. We provided a novel and efficient

framework to learn online decision trees. Our framework does not require the availability of all feature values, and it only observes the class label after making a prediction. These properties make our framework completely in accordance with the general online learning paradigm, and additionally makes it cost effective (i.e., it does not query all features of incoming data points).

To make our framework cost effective, we exploit an adaptive submodular surrogate objective function. The objective function is used to find informative features in a sampled environment which is provided via a posterior sampling procedure. Using posterior sampling also allows us to address the concept drift problem in a simple way. We analyze the regret of our online learning procedure in prior-independent and prior-dependant settings. Our extensive experiments confirm the cost-effectiveness of our online learning method, and show its competitiveness with baseline online decision tree learning models.

In the future, we can extend our method in different ways. First, we may generalize the online learning framework to address machine learning tasks other than classification (e.g., regression), and be able to accept various data types. Second, we can replace the active planning part of our framework with other techniques, including neural networks. For instance, a neural network might be used to estimate the informativeness of a given feature based on the previous observations, and thus help with the feature selection procedure. Lastly, one can extend the current setting to other kinds of feedback, meaning that we learn from observations other than the true label.

Bibliography

[1] Y. Bengio, A. C. Courville and P. Vincent, "Unsupervised feature learning and deep learning: A review and new perspectives," *CoRR*, vol. abs/1206.5538, 2012. arXiv: 1206.5538. [Online]. Available: http://arxiv.org/abs/1206.5538 (cit. on p. 3).

[2] H. Hotelling, "Analysis of a complex of statistical variables into principal components.," *Journal of educational psychology*, vol. 24, no. 6, p. 417, 1933 (cit. on p. 3).

[3] I. T. Jolliffe, *Principal component analysis for special types of data*. Springer, 2002 (cit. on p. 3).

[4] P. Comon, *Independent component analysis*, 1992 (cit. on p. 3).

[5] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006 (cit. on p. 3).

[6] X. Wu, K. Yu, W. Ding, H. Wang and X. Zhu, "Online feature selection with streaming features," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 5, pp. 1178–1192, 2012 (cit. on p. 3).

[7] J. Zhou, D. P. Foster, R. A. Stine, L. H. Ungar and I. Guyon, "Streamwise feature selection.," *Journal of Machine Learning Research*, vol. 7, no. 9, 2006 (cit. on p. 3).

[8] J. Wang, P. Zhao, S. C. Hoi and R. Jin, "Online feature selection and its applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 3, pp. 698–710, 2014. DOI: 10.1109/TKDE.2013.32 (cit. on p. 4).

[9] Y. Wu, S. C. H. Hoi, T. Mei and N. Yu, "Large-scale online feature selection for ultra-high dimensional sparse data," *ACM Trans. Knowl. Discov. Data*, vol. 11, no. 4, 2017, ISSN: 1556-4681. DOI: 10.1145/3070646. [Online]. Available: https://doi.org/10.1145/3070646 (cit. on p. 4).

[10] S. Arora, S. Du, W. Hu, Z. Li and R. Wang, "Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks," in *Proceedings of the 36th International Conference on Machine Learning*, K. Chaudhuri and R. Salakhutdinov, Eds., ser. Proceedings of Machine Learning Research, vol. 97, PMLR, 2019, pp. 322–332. [Online].

Available: https://proceedings.mlr.press/v97/arora19a.html (cit. on pp. 4, 6, 8, 15, 17).

[11] Z.-Q. Zhao, P. Zheng, S.-T. Xu and X. Wu, "Object detection with deep learning: A review," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 11, pp. 3212–3232, 2019. DOI: 10.1109/TNNLS.2018.2876865 (cit. on p. 5).

[12] S. Zhang, L. Yao, A. Sun and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," *ACM Comput. Surv.*, vol. 52, no. 1, 2019, ISSN: 0360-0300. DOI: 10.1145/3285029. [Online]. Available: https://doi.org/10.1145/3285029 (cit. on p. 5).

[13] C. Zhang, S. Bengio, M. Hardt, B. Recht and O. Vinyals, "Understanding deep learning (still) requires rethinking generalization," *Commun. ACM*, vol. 64, no. 3, 107–115, 2021, ISSN: 0001-0782. DOI: 10.1145/3446776. [Online]. Available: https://doi.org/10.1145/3446776 (cit. on pp. 5, 6).

[14] D. S. Rosenberg, *Lecture slides on foundations of machine learning*, 2017 (cit. on p. 6).

[15] C. Cortes, M. Mohri and A. Rostamizadeh, "Algorithms for learning kernels based on centered alignment," *J. Mach. Learn. Res.*, vol. 13, no. 1, 795–828, 2012, ISSN: 1532-4435 (cit. on pp. 8, 15).

[16] N. Cristianini, J. Shawe-Taylor, A. Elisseeff and J. Kandola, "On kernel-target alignment," in *Advances in Neural Information Processing Systems*, T. Dietterich, S. Becker and Z. Ghahramani, Eds., vol. 14, MIT Press, 2001. [Online]. Available: https://proceedings.neurips.cc/paper/2001/file/1f71e393b3809197ed66df836fe833e5-Paper.pdf (cit. on p. 8).

[17] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Advances in Neural Information Processing Systems*, J. Platt, D. Koller, Y. Singer and S. Roweis, Eds., vol. 20, Curran Associates, Inc., 2007. [Online]. Available: https://proceedings.neurips.cc/paper/2007/file/013a006f03dbc5392effeb8f18fda755-Paper.pdf (cit. on p. 8).

[18] C. Williams and M. Seeger, "Using the nyström method to speed up kernel machines," in *Advances in Neural Information Processing Systems*, T. Leen, T. Dietterich and V. Tresp, Eds., vol. 13, MIT Press, 2000. [Online]. Available: https://proceedings.neurips.cc/paper/2000/file/19de10adbaa1b2ee13f77f679fa1483a-Paper.pdf (cit. on p. 8).

[19] Y. Lu, L. Luo, D. Huang, Y. Wang and L. Chen, "Knowledge transfer in vision recognition: A survey," *ACM Comput. Surv.*, vol. 53, no. 2, 2020, ISSN: 0360-0300. DOI: 10.1145/3379344. [Online]. Available: https://doi.org/10.1145/3379344 (cit. on p. 9).

[20] Z. Alyafeai, M. S. AlShaibani and I. Ahmad, *A survey on transfer learning in natural language processing*, 2020. DOI: 10.48550/ARXIV.2007.04239. [Online]. Available: https://arxiv.org/abs/2007.04239 (cit. on p. 9).

[21] A. K. Sahu and P. Dwivedi, "Knowledge transfer by domain-independent user latent factor for cross-domain recommender systems," *Future Generation Computer Systems*, vol. 108, pp. 320–333, 2020, ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2020.02.024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X19316176 (cit. on p. 9).

[22] V. Vapnik and A. Vashist, "A new learning paradigm: Learning using privileged information," *Neural Networks*, vol. 22, no. 5, pp. 544–557, 2009, Advances in Neural Networks Research: IJCNN2009, ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2009.06.042. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608009001130 (cit. on p. 9).

[23] V. Vapnik and R. Izmailov, "Knowledge transfer in svm and neural networks," *Annals of Mathematics and Artificial Intelligence*, vol. 81, no. 1, pp. 3–19, 2017 (cit. on p. 9).

[24] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '00, Boston, Massachusetts, USA: Association for Computing Machinery, 2000, 71–80, ISBN: 1581132336. DOI: 10.1145/347090.347107. [Online]. Available: https://doi.org/10.1145/347090.347107 (cit. on pp. 10–12).

[25] S. Shalev-Shwartz, "Online learning and online convex optimization," *Foundations and Trends® in Machine Learning*, vol. 4, no. 2, pp. 107–194, 2012, ISSN: 1935-8237. DOI: 10.1561/2200000018. [Online]. Available: http://dx.doi.org/10.1561/2200000018 (cit. on p. 10).

[26] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014. DOI: 10.1017/CBO9781107298019 (cit. on p. 11).

[27] L. Breiman, J. Friedman, R. Olshen and C. Stone, *Classification And Regression Trees*. Routledge, 1984. DOI: 10.1201/9781315139470 (cit. on p. 11).

[28] G. Hulten, L. Spencer and P. Domingos, "Mining time-changing data streams," in *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '01, San Francisco, California: Association for Computing Machinery, 2001, 97–106, ISBN: 158113391X. DOI: 10.1145/502512.502529. [Online]. Available: https://doi.org/10.1145/502512.502529 (cit. on p. 11).

[29] C. Manapragada, G. I. Webb and M. Salehi, "Extremely fast decision tree," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '18, London, United Kingdom: Association for Computing Machinery, 2018, 1953–1962, ISBN: 9781450355520. DOI: 10.1145/3219819.3220005. [Online]. Available: https://doi.org/10.1145/3219819.3220005 (cit. on p. 11).

[30] W. Hoeffding, "Probability inequalities for sums of bounded random variables," in *The Collected Works of Wassily Hoeffding*, N. I. Fisher and P. K. Sen, Eds. New York, NY: Springer New York, 1994, pp. 409–426, ISBN: 978-1-4612-0865-5. DOI: 10.1007/978-1-4612-0865-5_26. [Online]. Available: https://doi.org/10.1007/978-1-4612-0865-5_26 (cit. on p. 12).

[31] O. Maron and A. Moore, "Hoeffding races: Accelerating model selection search for classification and function approximation," in *Proceedings of (NeurIPS) Neural Information Processing Systems*, Morgan Kaufmann, 1993, pp. 59 –66 (cit. on p. 12).

[32] S. Dasgupta, "Analysis of a greedy active learning strategy," in *Advances in Neural Information Processing Systems*, L. Saul, Y. Weiss and L. Bottou, Eds., vol. 17, MIT Press, 2004 (cit. on p. 13).

[33] R. Nowak, "Noisy generalized binary search," in *Advances in Neural Information Processing Systems*, Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams and A. Culotta, Eds., vol. 22, Curran Associates, Inc., 2009. [Online]. Available: https://proceedings.neurips.cc/paper/2009/file/556f391937dfd4398cbac35e050a2177-Paper.pdf (cit. on p. 13).

[34] D. Golovin, A. Krause and D. Ray, "Near-optimal bayesian active learning with noisy observations," in *Advances in Neural Information Processing Systems*, J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel and A. Culotta, Eds., vol. 23, Curran Associates, Inc., 2010 (cit. on p. 13).

[35] V. Chakaravarthy, V. Pandit, S. Roy, P. Awasthi and M. Mohania, "Decision trees for entity identification: Approximation algorithms and hardness results.," Jan. 2007, pp. 53–62 (cit. on p. 13).

[36] D. Golovin and A. Krause, "Adaptive submodularity: A new approach to active learning and stochastic optimization," *CoRR*, vol. abs/1003.3967, Jul. 2010 (cit. on p. 13).

[37] D. Golovin and A. Krause, "Adaptive submodularity: Theory and applications in active learning and stochastic optimization," *Journal of Artificial Intelligence Research*, vol. 42, Mar. 2010. DOI: 10.1613/jair.3278 (cit. on p. 13).

[38] W. R. Thompson, "On the likelihood that one unknown probability exceeds another in view of the evidence of two samples," *Biometrika*, vol. 25, no. 3-4, pp. 285–294, 1933 (cit. on p. 16).

[39] F. Yu, A. Seff, Y. Zhang, S. Song, T. Funkhouser and J. Xiao, "Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop," *arXiv preprint arXiv:1506.03365*, 2015 (cit. on p. 17).

[40] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998 (cit. on p. 17).

Part II

Appended Papers

Paper III

Efficient Online Decision Tree Learning by Utility of Features

Arman Rahbar, Ziyu Ye, Chaoqi Wang, Yuxin Chen, Morteza Haghir Chehreghani

To be submitted

EFFICIENT ONLINE DECISION TREE LEARNING BY UTILITY OF FEATURES

A PREPRINT

Arman Rahbar
Chalmers
armanr@chalmers.se

Ziyu Ye
University of Chicago
ziyuye@uchicago.edu

Chaoqi Wang
University of Chicago
chaoqi@uchicago.edu

Yuxin Chen
University of Chicago
chenyuxin@uchicago.edu

Morteza Haghir Chehreghani
Chalmers
morteza.chehreghani@chalmers.se

September 26, 2022

ABSTRACT

We consider the online decision tree learning problem. Existing works are usually limited in the settings where labels are given when streaming data arrives, which is impractical in fully online scenarios; or all features of the data are presented and queried to build the decision tree, which can be inefficient and costly when the feature dimension is large. Our work provides a fresh perspective to tackle those limitations. To learn efficiently with less time, we employ a posterior sampling scheme. To predict efficiently with lower cost, we propose a surrogate objective function on *utility of features*, enabling near-optimal performance with low feature acquisition cost. We provide a rigorous regret guarantee, and illustrate the efficiency and effectiveness of our framework, via extensive experiments on various real-world datasets. Our work also enables an exceptionally simple and elegant solution to the concept drift problem, 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], intrution detection [Jiang et al., 2013], network troubleshooting [Rozaki, 2015], etc.

Classical models aim to construct an online decision tree incrementally with streaming data. However, such models have several disadvantages. First, they require that all features are presented to determine splitting node. 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 steam. In contrast, our work takes feature acquisition cost (see formal definition in section 3) into considerations and we aim at the more challenging fully online case: we receive a data point at each step, and we need make accurate prediction of its label with low feature acquisition cost; the true label will only be observed *after* we make the prediction.

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., **U**tility of **F**eatures for **O**nline learning of **D**ecision **T**rees. As shown in Figure 1, our framework can be interpreted as an active planning oracle nested within an online learning model.

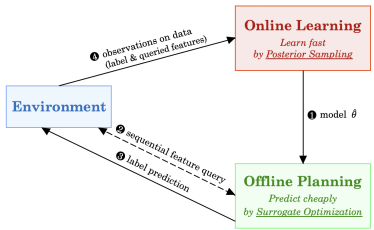

Figure 1: The proposed UFODT framework.

The online learning part employs a posterior sampling scheme to learn the environment of the streaming data, which is guaranteed to have good performance [Osband and Van Roy, 2017]; an additional 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 point in the data steam, should we observe the value of the chosen features. In order to *efficiently* optimize the utility of features, we consider an adaptive submodular surrogate objective following the key insight of Golovin et al. [2010] to sequentially query features and their values, which ensure us to predict accurately with (near-optimal) cost. In particular, the sequential feature query based on the surrogate objective of utility of features is a natural analogy to the node splitting based on direct objective of information gain in classical decision tree literature.

Our contributions are summarized as follows:

- We present a novel framework for efficiently constructing online decision trees in a cost-effective manner. Moreover, our use of surrogate objective as node splitting criteria also brings new algorithmic contributions.
- We provide a rigorous regret guarantee of our framework, and extensive experiments on diverse real-world datasets verify that our framework is able to achieve better accuracy with much lower cost, compared to baseline models.
- Based on our framework, we design a simple yet elegant variant to handle concept drift in streaming data, and we demonstrate the efficacy via various experiments on synthetic datasets.

Overall, the fully online setting and the considerations on feature query cost make it more practical to learn decision trees in the real world. We believe our unique perspective and flexible framework will enable new opportunities in online decision tree learning.

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 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], 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. [2017a], which adapts PSRL to solve online information acquisition problems; however, in contrast to our work, Chen et al. [2017a] 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 some greedy heuristics for the adaptive policy, e.g., Information Gain (IG) [Dasgupta, 2005], which greedily maximize the uncertainty reduction. However, such greedy heuristics may fail arbitrarily badly. Recently, researchers propose to optimize w.r.t. submodular surrogates, e.g., EC^2 [Golovin et al., 2010], HEC [Javdani et al., 2014], ECED [Chen et al., 2017b], which are proven to have near-optimal performance with low information acquisition cost. Those policies naturally fit into our problem and can be used as a plug-in solver in the offline planning phase.

3 Problem Formulation

3.1 Efficient Online Decision Tree Learning

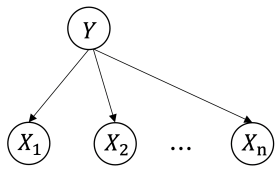

Figure 2: The Naïve Bayes model.

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

We now formally state the online decision tree learning problem. Let $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ 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 ith feature[1], and let $Y_j \in \mathcal{Y} \triangleq \{y_1, y_2, \ldots, y_m\}$ 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 strucutre: $\mathbb{P}[Y_j, X_1, \ldots, X_n] = \mathbb{P}[Y_j] \prod_{i=1}^{n} \mathbb{P}[X_i \mid Y_j]$, i.e., features are conditionally independent given the class. Since we are in the online setting, we assume that the *joint distribution* $\mathbb{P}[Y, X_1, \ldots, X_n]$ is initially unknown (though we may have prior knowledge on that) and needs to be learned via our online interactions.

We define $\theta_{ij} \triangleq \mathbb{P}[X_i = 1 \mid Y_j]$, and assume that θ is distributed by a Beta distribution, $\mathrm{Beta}(\alpha_{ij}, \beta_{ij})$. We use $\boldsymbol{\theta} = [\theta_{ij}]_{n \times m}$ to denote the probabilistic table for the data distribution and assume $\boldsymbol{\theta} \sim \mathrm{Beta}(\boldsymbol{\alpha}, \boldsymbol{\beta})$.

Under the above probabilistic model, each query on a feature value will provide some information about Y. We define the set of *queries* as $\mathcal{Q} \triangleq \{1, 2, \ldots, n\}$, and a query $i \in \mathcal{Q}$ will reveal the value of the ith feature. We define *cost* of the feature query as $c : \mathcal{Q} \to \mathbb{R}_{\geq 0}$. For simplicity, we assume $c = 1$ for each query.

Upon information gathered from feature query, we make a prediction. We define the loss of our prediction as $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$. Our goal is to reach low prediction loss with low query cost on data stream.

We additionally define $H = [X_1, \ldots, X_n]$ 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".

[1]For simplicity, we assume features are binary. Our setting can be easily extended to multicagorical or continuous feature cases.

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.

3.2 Utility of Features

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

Let y be our label prediction, and denote its associated loss w.r.t. the true data label y_{true} as l. We cab then naturally define the *utility of* y as $u \triangleq -l$ and the *conditional expected utility* of y upon observing $\boldsymbol{x}_{\mathcal{F}}$ as $U(y \mid \boldsymbol{x}_{\mathcal{F}}) \triangleq \mathbb{E}_{y_{\text{true}}}[u(y_{\text{true}}, y) \mid \boldsymbol{x}_{\mathcal{F}}]$. Note that we could define the utility similarly upon h, since h is the full realization of features.

Definition 1. *(Utility of features $\boldsymbol{x}_{\mathcal{F}}$)*

$$\mathbb{U}(\boldsymbol{x}_{\mathcal{F}}) \triangleq \max_{y \in \mathcal{Y}} U(y \mid \boldsymbol{x}_{\mathcal{F}}).$$

Here, $\mathbb{U}(\boldsymbol{x}_{\mathcal{F}})$ represents the maximal expected utility achievable given $\boldsymbol{x}_{\mathcal{F}}$. 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)*

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

Directly optimizing $\mathbb{U}(\boldsymbol{x}_{\mathcal{F}})$ is usually intractable, and greedy heuristics may fail or be costly, as described in Section 2. In the next, we will show how we may use a *surrogate objective* of $\mathbb{U}(\boldsymbol{x}_{\mathcal{F}})$ by the notion of decision regions to achieve near optimal query planning.

4 Proposed Framework

In this section, we present our UFODT framework for efficient online decision tree learning. The high-level structure is presented in Figure 1, which can be conceived as an *offline planning oracle* nested within on *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 offline 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 a adaptive policy which sequentially queries features (i.e., splitting nodes) of $\boldsymbol{x}^t$, 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 $\boldsymbol{x}^t$. Finally, the true label for $\boldsymbol{x}^t$ is revealed, and is then used to update the posterior of $\boldsymbol{\theta}$ together with the query observation $\boldsymbol{x}_{\mathcal{F}}$.

Below is the pseudo-code.

Algorithm 1 Online Decision Tree Learning

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

1: **for** $t = 1, 2, \ldots, T$ **do**
2: Sample $\boldsymbol{\theta}^t \sim \text{Beta}(\boldsymbol{\alpha}^{t-1}, \boldsymbol{\beta}^{t-1})$ and receive $\boldsymbol{x}^t$;
3: Call Algorithm 3 with $\boldsymbol{\theta}_t$ to sequentially query features (i.e., splitting nodes) and make prediction;
4: Observe $\boldsymbol{x}^t_{\mathcal{F}}$ and true label y^t_j;
5: Call Algorithm 2 to obtain $\text{Beta}(\boldsymbol{\alpha}^t, \boldsymbol{\beta}^t)$
6: **end for**

Algorithm 2 Posterior Update

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

for each $(i, x_i) \in \boldsymbol{x}_{\mathcal{F}}^t$ **do**
  if $x_i = 1$ **then** $\alpha_{ij}^t \leftarrow \alpha_{ij}^{t-1} + 1$
  else $\beta_{ij}^t \leftarrow \beta_{ij}^{t-1} + 1$
  end if
end for

4.2 Planning by Surrogate Optimization

In the planning phase, we seek to optimize an objective of $\mathbb{U}(\boldsymbol{x}_{\mathcal{F}})$ given the sampled environment. As mentioned earlier, traditional works usually use greedy heuristics (e.g., information gain) to optimize for that, which may fail arbitrarily badly. We instead propose to optimize for the *surrogate objective* of $\mathbb{U}(\boldsymbol{x}_{\mathcal{F}})$. Specifically, we adopt the EC^2 algorithm [Golovin et al., 2010], which uses the equivalence class edge cut as the surrogate objective of $\mathbb{U}(\boldsymbol{x}_{\mathcal{F}})$. Importantly, this surrogate objective is proved to be adaptive submodular, thus is near optimal, allowing us to make accurate prediction with low cost. Below we present the general statement of EC^2, where the description is adapted from Chen et al. [2017b], Golovin et al. [2010].

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. We define the weight of each edge as $w(\{h, h'\}) \triangleq \mathbb{P}(h) \cdot \mathbb{P}(h')$. Specifically, $\mathbb{P}(h)$ may be conceived as posterior distribution upon query of some 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: $E(x_i) \triangleq \{\{h, h'\} \in E : \mathbb{P}[x_i \mid h] = 0 \vee \mathbb{P}[x_i \mid h'] = 0\}$.

We now formally define the EC^2 objective as:

$$f_{EC^2}(\mathbf{x}_{\mathcal{F}}) \triangleq w\left(\bigcup_{v \in \mathcal{F}} E(x_v)\right),$$

and the score of feature query is defined as:

$$\Delta_{EC^2}(u \mid \mathbf{x}_{\mathcal{F}}) \triangleq w\left(\bigcup_{v \in \mathcal{F} \cup u} E(x_v)\right) - w\left(\bigcup_{v \in \mathcal{F}} E(x_v)\right).$$

The policy π_{EC^2} will greedily query the feature which maximizes the gain cost ratio $\Delta_{EC^2}(v \mid \mathbf{x}_{\mathcal{F}}) / c(v)$ and stops when only one decision region exists. We present the algorithm in Algorithm 3.

Algorithm 3 Planning by Surrogate Optimization

Sample hypotheses by calling Algorithm 4
$\mathcal{O} = \emptyset$
while stopping condition for EC^2 not reached **do**
  Use EC^2 to determine next feature $i \in \mathcal{Q}$
  Query feature i
  Add (i, x_i) to $\mathcal{O}$
  Update $\mathbb{P}(h \mid \mathcal{O})$
end while
Return the decision region y

4.3 Hypothesis Sampling Procedure

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 $\boldsymbol{\theta}$ to build a new sample.

Algorithm 4 Hypotheses Sampling

$\tilde{\mathcal{H}} \leftarrow \emptyset$
Sample decision regions from $\mathbb{P}(Y)$
for each sampled decision region j **do**
  $h \leftarrow \emptyset$
  for each $i \in \mathcal{Q}$ **do**
    Sample $X_i \sim$ Bernoulli(θ_{ij})
    Add the sample to h
  end for
  $\tilde{\mathcal{H}} = \tilde{\mathcal{H}} \cup h$
end for

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 codes upon Algorithm 2, which is shown in Algorithm 5.

Algorithm 5 Non-Stationary Posterior Update [Russo et al., 2017]

Input: $\boldsymbol{x}^t_{\mathcal{F}}$; y^t_j; $(\boldsymbol{\alpha}^{t-1}, \boldsymbol{\beta}^{t-1})$; γ; $(\bar{\boldsymbol{\alpha}}, \bar{\boldsymbol{\beta}})$.

$\boldsymbol{\alpha}^t \leftarrow (1-\gamma)\boldsymbol{\alpha}^{t-1} + \gamma\bar{\boldsymbol{\alpha}}$
$\boldsymbol{\beta}^t \leftarrow (1-\gamma)\boldsymbol{\beta}^{t-1} + \gamma\bar{\boldsymbol{\beta}}$
for each $(i, x_i) \in \boldsymbol{x}^t_{\mathcal{F}}$ **do**
  if $x_i = 1$ **then**
    $\alpha^t_{ij} \leftarrow \alpha^{t-1}_{ij} + 1$
  else
    $\beta^t_{ij} \leftarrow \beta^{t-1}_{ij} + 1$
  end if
end for

This inspiration comes from non-stationary Thompson 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 $\mathrm{Beta}(\bar{\boldsymbol{\alpha}}, \bar{\boldsymbol{\beta}})$ to increase uncertainty.

5 Algorithm Analysis

Since we are dealing with the fully online case, it is appropriate to examine the performance of our algorithm by regret analysis. In this section, we provide a guarantee on the regret bound of our framework by a direct adaptation of the results from Osband et al. [2013], Chen et al. [2017a]. Proof details can be found in the Appendix.

Let $\mathbb{U}(\pi) \triangleq \mathbb{E}_h\left[\max_{y\in\mathcal{Y}} \mathbb{E}_{y_{\text{true}}}[u(y_{\text{true}}, y) \mid \mathcal{S}(\pi, h)]\right]$ be the expected utility of features achieved by a policy π; here, $\mathcal{S}(\pi, h)$ represents the set of features and its values queried by policy π upon a hypothesis h.

As proved by Golovin and Krause [2011], we know that π^{EC^2} achieves the same utility as the optimal policy π^* under a same environment $\boldsymbol{\theta}$, 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. Simply put, EC^2 has the same prediction as the optimal policy, while having slightly higher cost.

Let $\boldsymbol{\theta}^\star$ be the true environment and $\boldsymbol{\theta}^t$ be the sampled environment at epoch t, we have:

Definition 3. *(Immediate regret at epoch t)*

$$\Delta^t \triangleq \mathbb{U}(\pi^*_{\boldsymbol{\theta}^\star}) - \mathbb{U}(\pi^{\text{EC}^2}_{\boldsymbol{\theta}^t}).$$

The total regret until epoch T is then $\text{Regret}(T) = \sum_{t=1}^{T} \Delta^t$. Let $H = (2\ln(1/p_{\min}) + 1) \cdot c_{\pi^*}$ denote the worst-case cost (i.e., number of queries) of EC^2 in any epoch, S be the number of possible realization of H queries, n be the total number of features, we thus have the following regret bound for Algorithm 1 based on the result of Osband et al. [2013]:

Theorem 4. *(Expected total regret at epoch T)*

$$\mathbb{E}[\text{Regret}(T)] = O(HS\sqrt{nT\log(SnT)}).$$

Notice that this bound depends on the worst-case cost, which could potentially be huge, and it is prior-independent such that the effect of prior knowledge by the posterior sampling scheme is not reflected.

We here in addition provide a prior-dependent bound based on recent results from Russo and Van Roy [2016], Lu et al. [2021]:

Theorem 5. *(Prior dependent expected regret)*

$$\mathbb{E}[\text{Regret}(T)] \leq \sqrt{\bar{\Gamma}\mathbb{H}(\boldsymbol{\theta}^\star)T}.$$

Here, $\bar{\Gamma}$ is the maximal information ratio[2] of the algorithm, and $\mathbb{H}(\boldsymbol{\theta}^\star)$ represents the initial entropy of $\boldsymbol{\theta}^\star$. A more informative prior will lead to smaller value of $\mathbb{H}(\boldsymbol{\theta}^\star)$, hence a better bound; this also aligns with our observations in Figure 8, showing that our framework has the advantage over traditional ODT models that we can effectively use prior knowledge.

6 Experimental Results

In this section, we study the empirical evaluation of our methods on various datasets. Unless otherwise specified, we assume that we have a uniform prior on $\boldsymbol{\theta}$ for each dataset initially. We evaluate the methods introduced in sections 4 from different aspects. We compute the average number of queries per session (i.e., per data point) to compare the costs of different algorithms. We also evaluate the classification performance on the training data during learning. Furthermore, we measure the generalization power of different classifiers via a holdout test dataset.

Datasets. We have used four stationary datasets in our experiments that are standard binary classification datasets taken from UCI repository [Dua and Graff, 2017]. 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.

Algorithms. The VFDT algorithm [Domingos and Hulten, 2000] is used as a classical baseline ODT model. Within our proposed UFODT framework, we use three classical greedy strategies as baselines to compare with EC^2. The Information Gain (IG) algorithm, selects the feature that maximizes the reduction of entropy in labels. Similarly, Uncertainty Sampling (US) finds the feature causing the highest reduction in entropy of hypotheses. We also use random feature selection which randomizes the order of querying features.

We present and discuss the results of our experiments on different datasets as follows.

6.1 Experiments on stationary datasets

The UCI datasets used in our experiments are: LED Display Domain, Zoo, SPECT Heart, and Congressional Voting Records which have 7, 15[3], 22, and 16 binary features respectively. The algorithms used here are UFODT-EC^2, UFODT-IG, UFODT-US, and UFODT-random all of which are using the method described in Algorithm 1. UFODT-IG, UFODT-US, and UFODT-random change the criterion determining the next queried feature to Information Gain, Uncertainty Sampling, and random respectively.

Utility and label complexity. Figure 3 shows the average label complexity (the number of features queried during one session) during training as a function of the total number of hypotheses sampled. Figure 4 shows the total utility during training versus the total number of sampled hypotheses. If a dataset is balanced we use accuracy as the utility whereas we use f-measure for imbalanced datasets. We observe that UFODT-EC^2 yields the lowest label complexity in three cases and its utility is competitive compared to the best results. This observation shows that the UFODT-EC^2 tends to find more informative features to query, meaning that with less number of features (lower cost) it can reach a good utility. UFODT-IG also yields a better label complexity than random feature selection and UFODT-US. On the other hand, both UFODT-random and UFODT-US required high label complexities which do not help them to yield high utilities. There is also one case (Voting Records) in which UFODT-EC^2 has the highest cost. This high cost helps the

[2]Details can be found in the Appendix, and we leave the exact analytical form of $\bar{\Gamma}$ as the future work.

[3]The zoo dataset has 17 features two of which are not binary.

algorithm to reach a utility that is significantly higher than any other algorithm.

Train and test utilities during training. Figures 5 and 6 show the utility (accuracy or f-measure) of the predictions over the training data points seen so far and the utility over a holdout test dataset. In order to compute the utility over the test dataset, we use the the current estimation of the parameters of the conditional distribution (i.e., the estimated $\boldsymbol{\theta}$ at time t) to obtain the test predictions. In this case, we do not have feature selection and we assume all features of a test datapoint are available. For LED, Zoo and Heart datasets, we observe that the UFODT-EC^2 algorithm reaches a very good utility during training despite its lower cost. In the case of Voting Recodrs dataset, we can see that UFODT-EC^2 reaches a significantly higher utility in the first steps of training with a slightly higher cost. In Figure 6 we have included the test utility results from the VFDT algorithm as well. We note that this algorithm requires access to full features and class label while training, unlike the other methods. We observe that our algorithm reaches test utilities that are comparable with VFDT despite its lower cost, and we perform even better than VFDT in the case of LED and Heart datasets.

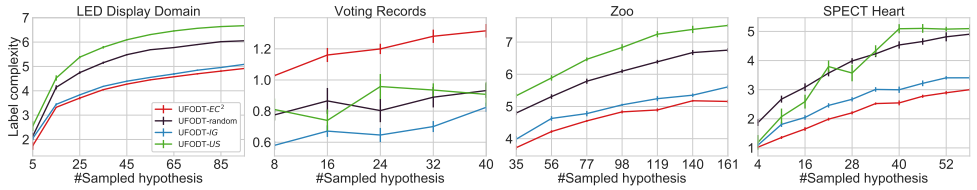

Figure 3: Training cost vs. #sampled hypotheses. UFODT-EC^2 has the lowest cost in three cases.

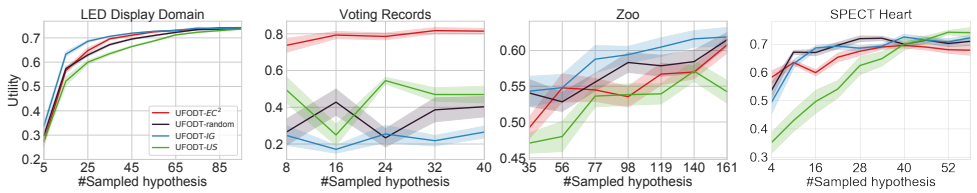

Figure 4: Training utility vs. #sampled hypotheses. We observe that despite the low cost in UFODT-EC^2, its utility is close to other methods. In the Voting Records dataset, it reaches a significantly higher utility with a slightly higher cost than others.

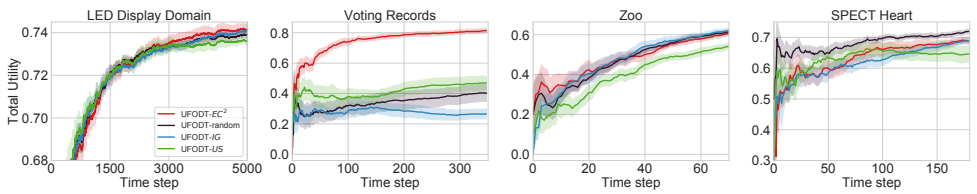

Figure 5: Training utility during training iterations. UFODT-EC^2 reaches a good utility during training steps with low cost. For the Voting Records dataset, it reaches a high utility in the first steps with a slightly higher cost than others.

6.2 Experiments with Concept Drift Dataset

In this section, we demonstrate the effectiveness and flexibility of our framework under the concept drifting setting. Under this setting, the non-stationary nature of the online data impose extra difficulty for online decision tree learning problem. To handle with the non-stationary distribution, we adopt the non-stationary Thompson sampling introduced in Russo et al. [2017]. Detailed descriptions can be found in Algorithm 5.

To simulate the concept drift scenarios, we adopt the Stagger dataset [Widmer and Kubat, 1996, López Lobo, 2020]. In this dataset, each data has three attributes, namely shape $\in$ {circular, non-circular}, size $\in$ {small, medium, large}, color $\in$ {red, green}. Initially, the instances are labeled as positive if (color=red)$\wedge$(size=small). There are in total two

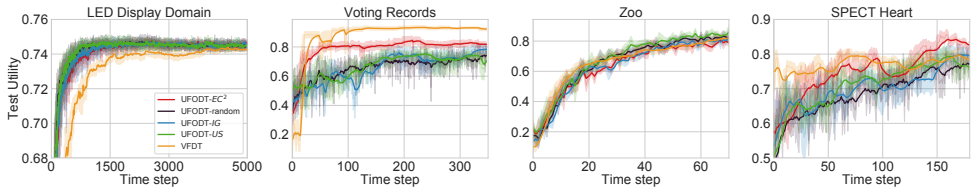

Figure 6: Test utility during training iterations. We see that UFODT-EC^2 reaches test utilities comparable with the ones from VFDT despite its lower cost, and we perform even better than VFDT in the case of LED and Heart datasets.

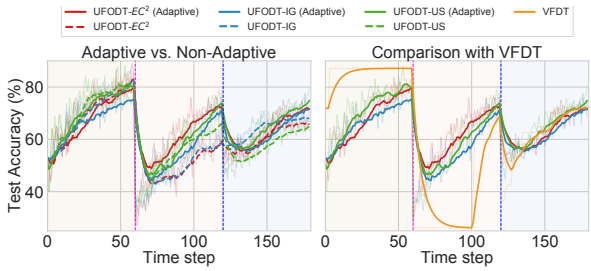

Figure 7: Time step vs. Test accuracy on Stagger dataset. Each shaded area corresponds to one target concept and the vertical dashed line is the point where concept drifting happens.

concept drifting that happens abruptly at some point. After the first and the second concept drifting, the positive concept will be (color=green)$\vee$(shape=circular) and (size=medium)$\vee$(size=large), respectively. In our experiments, we generate 60 data for each concept, and 200 data for testing. The concept drifting happens after iteration 60 and 120. To make the conclusion more robust, we repeat each experiment with 10 random seeds and report the averaged results along with one-standard error. In the next, we seek to understand the following questions through our carefully designed experiments.

Test Accuracy vs. Time step. In Figure 7, we report the results of UFODT-EC^2, UFODT-IG and UFODT-US with both standard Thompson sampling and non-stationary Thompson sampling (denoted with *Adaptive*). For all of our methods, we adopt the uniform prior. On the left, we can observe that all the three methods with non-stationary Thompson sampling can adapt to the abrupt concept drift much faster, and also achieve higher test accuracy. We also compared these three methods against to VFDT in the right figure. Though, initially, VFDT can achieve higher accuracy than our methods, it has a big drop in accuracy after both the first and second concept drifting. This demonstrates advantages of our methods in quickly adapting to new concepts over VFDT. In addition, we also report the averaged total number of feature queries as a cost measure in Table 1. We observe that our approaches requires significantly fewer feature queries (or lower cost) but still achieve higher accuracy than VFDT (see Figure 8 left).

The impact of priors. UFODT can easily incorporate expert's knowledge by using the corresponding prior distribution, which shows that UFODT is more flexible than classic decision tree learning algorithms. To simulate different experts, we generate a collection of priors that interpolate between the uniform prior (uninformative) and the 'optimal' prior (expert). We report the average test accuracy for different priors in Figure 8. We observe that as the quality of the prior improves, the average test accuracy will also increase and surpass VFDT by a larger margin. In general, with more informative priors, our approaches will also performs better, which is consistent with the regret analysis in Theorem 5.

The impact of #sampled hypothesis. Since UFODT relies on hypothesis sampling (See Algorithm 4) for constructing the graph, we further test how is the performance affected by the number of sampled hypothesis. The results are

Table 1: The averaged total number of feature queried.

Method	UFODT-EC^2 (Adaptive)	UFODT-IG (Adaptive)	UFODT-US (Adaptive)	VFDT
Cost $\downarrow$	343.3 ± 11.0	350.6 ± 5.1	477.0 ± 13.9	720.0

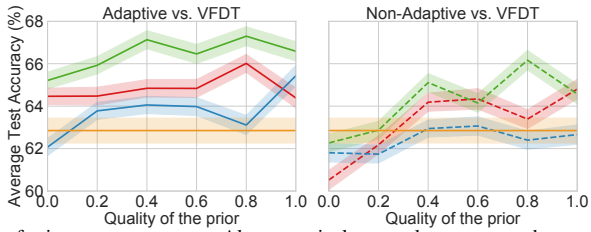

Figure 8: Quality of prior vs. test accuracy. Along x-axis, larger value corresponds to more accurate prior.

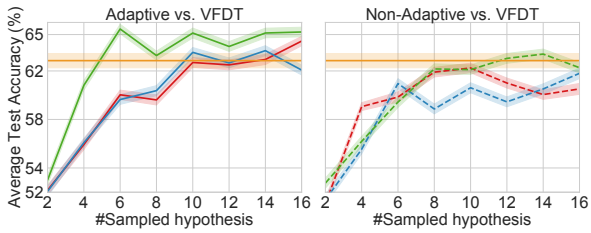

Figure 9: The effect of #sampled hypothesis on the test accuracy.

presented in Figure 9. As expected, as the number of sampled hypothesis increases, the test accuracy will also increase. However, the test accuracy usually saturates around #Sampled hypothesis=8 or 10. This shows that enumerating all the possible hypothesis may not be necessary, which can potentially save the computational cost.

7 Conclusion

We developed a novel framework for online decision tree problem which does not require availability of class labels when streaming data arrives, nor presence of all features which can be inefficient and costly. Within the framework, we employed a posterior sampling scheme in order to learn the unknown parameters efficiently. To predict (plan) efficiently, we developed a surrogate objective function on utility of features with near optimal performance and low feature acquisition cost. Our framework also provides a simple and elegant solution to the concept drift problem, which is competitive with baseline models while being more flexible. We then analyzed the regret of the online learning scheme in prior-independent and prior-dependent settings. Finally, we investigated the proposed framework on several datasets and showed its effectiveness in different settings.

References

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.

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.

Eleni Rozaki. Design and implementation for automated network troubleshooting using data mining. *International Journal of Data Mining & Knowledge Management Proces*, 5(3), 2015.

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.

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.

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.

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.

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.

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.

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.

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.

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.

Ian Osband, Daniel Russo, and Benjamin Van Roy. (more) efficient reinforcement learning via posterior sampling. *arXiv preprint arXiv:1306.0940*, 2013.

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.

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.

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, 2017a. URL `http://auai.org/uai2017/proceedings/papers/83.pdf`.

Ashish Kapoor and Eric Horvitz. Breaking boundaries: Active information acquisition across learning and diagnosis. *Advances in neural information processing systems*, 2009.

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.

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.

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.

Sanjoy Dasgupta. Analysis of a greedy active learning strategy. *Advances in neural information processing systems*, 17: 337–344, 2005.

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.

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, 2017b.

Ronald A Howard. Information value theory. *IEEE Transactions on systems science and cybernetics*, 2(1):22–26, 1966.

Vladimir Vapnik. Principles of risk minimization for learning theory. In *Advances in neural information processing systems*, pages 831–838, 1992.

Daniel Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, and Zheng Wen. A tutorial on thompson sampling. *arXiv preprint arXiv:1707.02038*, 2017.

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.

Daniel Russo and Benjamin Van Roy. An information-theoretic analysis of thompson sampling. *The Journal of Machine Learning Research*, 17(1):2442–2471, 2016.

Xiuyuan Lu, Benjamin Van Roy, Vikranth Dwaracherla, Morteza Ibrahimi, Ian Osband, and Zheng Wen. Reinforcement learning, bit by bit. *arXiv preprint arXiv:2103.04047*, 2021.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

Gerhard Widmer and Miroslav Kubat. Learning in the presence of concept drift and hidden contexts. *Machine learning*, 23(1):69–101, 1996.

Jesús López Lobo. Synthetic datasets for concept drift detection purposes, 2020. URL `https://doi.org/10.7910/DVN/5OWRGB`.

8 Supplementary Material

1 Proofs of The Main Theorems

1.1 Proof of Theorem 4

While in Section 6.1, we have discussed the impact of running algorithms with different numbers of sampled hypotheses, here for simplicity we consider the optimistic case that we have sampled a sufficient number of times from the decision region $\mathbb{P}(Y)$, such that all hypotheses with non-zero probability in $\mathcal{H}$ are enumerated[4].

Proof. Our results relies on the posterior sampling for reinforcement learning (PSRL) bound as follows:

Theorem 6. *(Theorem 1 of Osband et al. [2013]) Consider learning to optimize a random finite horizon $M = (\mathcal{B}, \mathcal{A}, R^M, P^M, H, \rho)$ in T repeated time epochs, where $\mathcal{B}$ denotes the state set with cardinality S, $\mathcal{A}$ denotes the action set with cardinality A, R^M denotes the reward function, $P_i^M(s' \mid s)$ represents the transition probability from state s to state s' upon choosing action i, H represents the time horizon (i.e., number of actions) of each epoch, ρ is the initial state distribution, and consider running the following algorithm: at the beginning of each epoch, we update a prior distribution over M and takes a sample from the resulting posterior distribution, then we follow the policy which is the optimal for this sampled distribution to take actions sequentially during the epoch. Then, for* any *prior distribution of M, we have the expected regret for our algorithm as follows:*

$$\mathbb{E}\left[\text{Regret}\left(T\right)\right] = O(HS\sqrt{AT\log(SAT)}).$$

Our problem can be viewed as a Partially Observable Markov Decision Process (POMDP) with a posterior sampling algorithm, thus we can directly adapt the above bound. Specifically:

- Time horizon H: The number of feature queries made during each epoch can be considered as the time horizon. This aligns with our definition of H in Section 5.
- Set of actions $\mathcal{A}$: Each feature query at a certain time step within an epoch can be considered as an action. Thus the cardinality A in the above bound is equivalent to n where n is defined as number of features.
- Set of states $\mathcal{B}$: Intuitively, we take each action based on current observations from the feature query. Thus, each sequential query set with the resulting outcomes can be considered as a state. The number of possible realizations during an epoch is then equivalent to the cardinality of the state set S.
- Transition probability $P_i^M(s' \mid s)$: This can be fully specified by $\mathbb{P}[X_i \mid Y]$ as described in Section 3.
- Initial distribution ρ: Similarly, this can be fully specified by $\mathbb{P}[X_i \mid Y]$ and the given $\mathbb{P}[Y]$.
- Reward function R^M: We consider the reward to be the expected utility achieved upon termination. To be specific, we receive zero reward if the algorithm continues to query features (i.e., stopping condition not reached), and receive expected reward $\mathbb{U}(\pi \mid h) \triangleq \max_{y \in \mathcal{Y}} \mathbb{E}_{y_{\text{true}}}[u(y_{\text{true}}, y) \mid \mathcal{S}(\pi, h)]$ upon termination by the proposed policy based on the true hypothesis.
- Optimal policy for M: As illustrated in Section 5, our offline planning algorithm EC^2 achieves the same utility as the optimal policy under the same environment $\boldsymbol{\theta}$. Thus, $\pi_{\boldsymbol{\theta}^t}^{\text{EC}^2}$ can be considered as the optimal policy for the sampled M in each epoch.

By replacing the notations on the cardinality of the action space in Theorem 6, we have the expected regret of Algorithm 1 as $\mathbb{E}[\text{Regret}(T)] = O(HS\sqrt{nT\log(SnT)})$, thus we complete the proof. □

1.2 Proof of Theorem 5

This prior-dependent bound for posterior sampling is first proposed by Russo and Van Roy [2016] for the multi-armed bandit problems. The core of the analysis is the *information ratio*, which precisely captures the exploration-exploitation tradeoff of the policy at each time epoch.

Such a bound may be potentially "better" than the previous bound in terms of its dependence on the (supremum of) information ratio and the dependence on the initial epistemic uncertainty of the environment. Firstly, the information ratio can be bounded by certain "dimension" of the problem (e.g., the feature dimension of linear bandits), which can be vastly smaller than the cardinality of action/state space. Secondly, the initial epistemic uncertainty reflects our prior knowledge on the environment, which previous regret bounds cannot benefit from.

[4] In a weaker form, it has been proved in Chen et al. [2017a] that sampling only the *most likely* hypotheses will leads to just an additive factor to the regret bound. Our framework holds the similar argument, while enjoying a simpler hypothesis generating scheme.

Proof. We define the *information ratio* of Algorithm 1 as follows:

$$\Gamma_t^{\mathrm{EC}^2} = \frac{\left(\mathbb{E}\left[\mathbb{U}\left(\pi^*_{\boldsymbol{\theta}^\star}\right) - \mathbb{U}\left(\pi^{\mathrm{EC2}}_{\boldsymbol{\theta}^t}\right)\right]\right)^2}{\mathbb{E}_h\left[\mathbb{I}\left(\boldsymbol{\theta}^\star; (\boldsymbol{x}_{\pi^{\mathrm{EC2}}_{\boldsymbol{\theta}^t}}, y^t, h) \mid \mathbb{H}^{t-1}\right)\right]},$$

where $\boldsymbol{x}_{\pi^{\mathrm{EC2}}_{\boldsymbol{\theta}^t}}$ represents all the queries and the associated outcomes made by EC^2 under the sampled $\boldsymbol{\theta}^t$, $\mathbb{H}^{t-1}$ represents all the decision and observation history up to the epoch $t-1$, and $\mathbb{I}(\cdot)$ represents the mutual information (i.e., entropy reduction). We omit the $\mathbb{E}_h$, $\mathbb{H}^{t-1}$ and h terms in the following to simplify notations.

Simply put, the numerator is the square of the expected *immediate regret* at epoch t, and the denominator captures the expected information gain on the true environment $\boldsymbol{\theta}^\star$ by implementing the current policy. The information ratio as a whole can be interpreted as "the expected regret incurred per bit of information acquired" [Russo et al., 2017].

Define the maximal information ratio for the algorithm as $\bar{\Gamma} = \max_{t\in\{1,\ldots,T\}} \Gamma_t^{\mathrm{EC}^2}$. Following the proof in Proposition 1 of Russo and Van Roy [2016], we derive the bound of Algorithm 1 as follows:

$$\begin{aligned}
\mathbb{E}[\mathrm{Regret}(T)] &= \sum_{t=1}^{T} \mathbb{E}\left[\mathbb{U}\left(\pi^*_{\boldsymbol{\theta}^\star}\right) - \mathbb{U}\left(\pi^{\mathrm{EC2}}_{\boldsymbol{\theta}^t}\right)\right] \\
&= \sum_{t=1}^{T} \sqrt{\Gamma_t^{\mathrm{EC}^2} \mathbb{I}\left(\boldsymbol{\theta}^\star; (\boldsymbol{\theta}^t, \mathbf{x}_{\pi^{\mathrm{EC2}}_{\boldsymbol{\theta}^t}}, y^t)\right)} \\
&\leq \sqrt{\bar{\Gamma} T \sum_{t=1}^{T} \mathbb{I}\left(\boldsymbol{\theta}^\star; (\boldsymbol{\theta}^t, \mathbf{x}_{\pi^{\mathrm{EC2}}_{\boldsymbol{\theta}^t}}, y^t)\right)} \qquad \text{(by Jensen's inequality)} \\
&\leq \sqrt{\bar{\Gamma} H\left(\boldsymbol{\theta}^\star\right) T} \qquad \text{(by the chain rule of mutual information)}.
\end{aligned}$$

This is exactly the bound we provide in Theorem 5. A promising direction to find the closed form of $\bar{\Gamma}$ (and the "dimension" of the problem) is to utilize the auxiliary function of entropy by Chen et al. [2017b] to establish a connection between immediate regret and information gain, which we will leave for future work.

□