

Time-Constrained Learning

Sergio Filho¹, Eduardo Laber¹, Pedro Lazera¹, and Marco Molinaro¹

¹Computer Science Department, PUC-RIO
 {sfilhofreitas,eduardo.laber1,molinaro.marco,pedrolazera}@gmail.com

February 7, 2022

Abstract

Consider a scenario in which we have a huge labeled dataset \mathcal{D} and a limited time to train some given learner using \mathcal{D} . Since we may not be able to use the whole dataset, how should we proceed? Questions of this nature motivate the definition of the Time Constrained Learning Task (TCL): Given a dataset \mathcal{D} sampled from an unknown distribution μ , a learner \mathcal{L} and a time limit T , the goal is to obtain in at most T units of time the classification model with highest possible accuracy w.r.t. to μ , among those that can be built by \mathcal{L} using the dataset \mathcal{D} .

We propose TCT, an algorithm for the TCL task designed based that on principles from Machine Teaching. We present an experimental study involving 5 different Learners and 20 datasets where we show that TCT consistently outperforms two other algorithms: the first is a Teacher for black-box learners proposed in [Dasgupta et al., ICML 19] and the second is a natural adaptation of random sampling for the TCL setting. We also compare TCT with **Stochastic Gradient Descent** training – our method is again consistently better.

While our work is primarily practical, we also show that a stripped-down version of TCT has provable guarantees. Under reasonable assumptions, the time our algorithm takes to achieve a certain accuracy is never much bigger than the time it takes the batch teacher (which sends a single batch of examples) to achieve similar accuracy, and in some case it is almost exponentially better.

1 Introduction

A common problem that arises in many supervised machine learning applications is the difficulty of acquiring labeled data. To overcome this problem techniques as active learning and semi-supervised learning have been successfully employed.

However, in other situations one faces the complementary scenario where a large number of labeled examples are available but the computational resources to train a model over them are limited. That may arise when one has a limited financial budget to train a model on a cloud; in this case the limited budget naturally translates into a time limit. In another setting, very current, one may set a limit on the training time to reduce the environmental impact. There are also applications such as ad advertisement and learning from search logs where the number of labeled examples is massive and repeated training is necessary to track new user behaviour distributions, so that training with all labeled examples is not feasible.

For a more concrete situation, assume that our financial budget only allows 4 hours of cloud usage but we want to train a random forest on a huge set (e.g. billions) of examples. Given that we may not be able to use the whole dataset, how should we proceed?

Questions of this nature are faced by machine learning practitioners. This is the motivation behind the Time Constrained Learning task (TCL for short), which is the focus of our work. We consider the following formulation for TCL:

Input. A dataset \mathcal{D} of labeled examples sampled from an unknown probability distribution μ , a learner \mathcal{L} and a time limit T

Output. The most accurate classification model w.r.t. μ that can be built by Learner \mathcal{L} , spending at most T time units.

TCL admits two natural variations: one in which we have a considerable amount of information about the Learner (e.g. its hypothesis class and training/classification time complexity) and the other in which the information is limited (black-box Learners). We are interested in the latter due to its wider applicability. Given the lack of information about the Learner, the following question arises:

Is it possible to outperform random sampling (or some natural variation) for TCL?

To give a positive answer to this question, we approach the TCL task via a Machine Teaching framework Shinohara [1991], Zhu et al. [2018] where a Teacher and a Learner interact over multiple rounds and in each of them the former sends selected examples to the latter, which returns a trained model. Our goal is then to design a teacher that guides the Learner towards a model with high accuracy w.r.t. μ in a way that is as time efficient as possible.

While most of the initial works on Machine Teaching [Shinohara, 1991, Goldman and Kearns, 1995] assume that the Teacher has significant knowledge about the Learner, there have been several recent advances on the case of interest where there is limited knowledge about the Learner [Melo et al., 2018, Liu et al., 2018, Dasgupta et al., 2019, Cicalese et al., 2020, Devidze et al., 2020].

However, these methods do not exactly address the TCL task and instead focus on minimizing the number of samples sent to the Learner, that is, in the *teaching set*. While the size of the teaching set and time complexity are related, there are factors, such as training time and the amount of interaction with the Learner, that should be considered when the latter is taken into account. This aspect is illustrated in the beginning of Section 2. Indeed, methods that disregard these factors are not suitable for TCL, as indicated by our experiments. Thus, new methods shall be developed to properly handle time constrained learning.

1.1 Our contributions.

Our main contribution is the algorithm Time Constrained Teacher (TCT) for the Time Constrained Learning task that is designed based on well-established principles/ideas from Computational Learning Theory and Machine Teaching. We compare TCT with two other Teachers, using 5 different Learners over 20 datasets. The first Teacher can be viewed as an adaptation of random sampling for the TCL task and, thus, is a natural baseline. The second, denoted here by OSCT, is based on an algorithm for the Online Set Covering problem [Alon et al., 2009]. Its use for teaching black-box learners (aiming at minimizing the size of the teaching set) was proposed in [Dasgupta et al., 2019] and refined/extended in [Cicalese et al., 2020].

For two of the Learners, namely SVM and Logistic Regression, we also compare the error achieved by TCT with that achieved by training them via Stochastic Gradient Descent (SGD).

Our algorithm TCT consistently outperforms its competitors and, for some Learners, perhaps surprisingly, TCT is competitive even against the “oracle” teacher that sends (in one single batch) m^* random samples to the Learner, where m^* (which is guessed by the oracle) is the largest number of examples that the Learner can handle within the time limit. A nice feature

of our algorithm is its simplicity: it just employs one parameter that is easy to set and it does not require any information about the learner. Hence, we believe that it could be easily implemented as a wrapper in machine learning libraries to address Time Constrained Learning – the user provides the time limit and the Learner that she is comfortable with, and then TCT completes the job.

While our work is primarily practical, we also show that a stripped-down version of TCT has provable guarantees. Under reasonable assumptions, the time our algorithm takes to achieve a certain accuracy is never much bigger than the time it takes the batch teacher (which sends a single batch of examples) to achieve similar accuracy. Moreover, for learning a threshold function, a canonical problem in Active Learning and Machine Teaching, our algorithm is almost exponentially faster, despite not being tailored to this hypothesis class.

1.2 Related work.

Although some Learners admit online versions that can be employed to TCL task, we are not aware of works that directly address this task for black-box Learners. In [Dasgupta et al., 2019], this task is mentioned as a potential application for their proposed Teacher. [Jun Du, 2011] mention the possibility of minimizing the training time rather than the size of the teaching set, although it handles the latter.

On the other hand, there are quite a few papers aiming at minimizing the size of the teaching set. Among these works, we can find some that consider the batch setting Singla et al. [2014], Ma et al. [2018] and others that consider the sequential one [Jun Du, 2011, Liu et al., 2018, Chen et al., 2018]. We can also find Teachers that assume a considerable amount of information about the Learner [Singla et al., 2014] as well as some that require very limited information [Jun Du, 2011, Dasgupta et al., 2019, Cicalese et al., 2020]

Most of these Teachers do not admit a simple adaptation for the TCL task because they require a lot of information about the Learner’s hypothesis class or training algorithm (e.g., [Liu et al., 2018] considers learners that use SGD for training). One exception is the method **OSCT**, from Dasgupta et al. [2019] and Cicalese et al. [2020], that assumes very limited information about the Learner. Experiments from the latter, using **LGBM** and **Random Forest** as Learners, show that **OSCT** requires significantly fewer examples to reach a given accuracy (on the training set) than a Teacher that sends random examples. Despite these gains in *number of examples*, on *time constrained* learning **OSCT** performs much worse than our algorithm **TCT**, and is even worse than random sampling, as we show in our experiments.

Our Time Constrained Learning scenario can be related to the traditional Active Learning scenario Settles [2009]. On one hand, the scenarios are quite different: in Active Learning, labeled examples are the constrained resource, while in our setting they are abundant. On the other hand, we cannot use all the available labeled examples and, thus, we need to choose (carefully) the ones to be used for training; this is related to the key aspect of Active Learning, that is, the selection of informative examples. Thus, Active Learning strategies could be used to select examples for Time-Constrained Learning. One issue here is that classical strategies, e.g. uncertainty sampling, may require assumptions about the Learner (e.g. the ability to produce class probabilities), which is not aligned with our goal of training **black-box** learners. One strategy that does not need these assumptions is Query by Committee, but it is too time-consuming for our setting. In Section 3, we present experiments where we evaluate a variation of **TCT** that uses ideas from a classical active learning strategy to select examples.

2 A Teacher for the Time Constrained Learning task

In this section we describe our algorithm TCT. Its design takes into account the following principles:

- P1 The larger the number of examples, the better the learning;
- P2 Examples where a Learner fails are more helpful to improve its accuracy than those in which it succeeds;
- P3 Examples selected to train a Learner should follow approximately the probability distribution μ , employed in TCL’s definition.

Principle 1 is very natural and justified by both empirical studies and standard statistical guarantees. However, to obtain a model trained on a large number of examples by a time limit T , it is important not to send too few examples in each round, as illustrated below.

Example 1. *Consider a learner with quadratic running time, that is, $\Theta(m^2)$ time units are required to train a model when m examples are available. If the teacher always sends one single example per round, then the largest model will have $O(\sqrt[3]{T})$ examples, where T is the time limit. On the other hand, if the teacher always doubles the size of the set of examples sent to the learner, then the largest model will have $\Omega(\sqrt{T})$ examples.*

We note that in contrast to this idea, previous works on Machine Teaching that focus on minimizing the size of the teaching set suggest the addition of few examples per round [Jun Du, 2011, Dasgupta et al., 2019, Cicalese et al., 2020].

Principle 2 is motivated by human learning and also by works on Machine Teaching [Dasgupta et al., 2019, Cicalese et al., 2020] and boosting [Schapire, 1999], where wrong examples get larger priority than those in which the Learner succeeds. While wrong examples are helpful, their acquisition may be expensive in terms of computational time since finding them may require the classification of a large number of examples. Thus, we need to balance between the usefulness of having wrong examples and the computational cost to get them. Moreover, we have to add wrong examples with parsimony, otherwise we may build models using a set of examples that is not representative of the real population and hence with potential poor performance on unseen data. This is captured in Principle 3. Indeed, issues with using biased samples for learning is well-documented in active learning [Dasgupta, 2009].

Based on these observations, we designed TCT, presented in Algorithm 1. It receives an integer m_0 (number of examples for the first model), a learner L , a parameter $\alpha \in [0, 1]$ that defines the ratio between wrong and random examples provided to the learner at each round; a time limit T and a pool P of labeled examples. We note that α allows a trade-off between following Principles 2 and 3. For the ease of presentation we assume that the pool P is large enough so that it is always possible to obtain unselected examples from it.

At each round, TCT receives from the learner a model M , trained on a set S , and then builds and provides to the learner a new set of examples that contains (approximately) $\alpha|S|$ examples in which M fails and also $(1 - \alpha)|S|$ random unseen examples from P . More precisely, TCT first employs M to classify a random set A_1 containing $|S|$ examples (Lines 4 -5). As a result, it also obtains an unbiased estimation acc_1 for the accuracy of M . Next, based on acc_1 TCT builds a second set of samples A_2 , which is done to guarantee that in expectation we have at least $\alpha|S|$ examples in $A_1 \cup A_2$ where M fails. It then classifies this set A_2 using M (Lines 6-7), obtaining a second unbiased estimation for the accuracy of M , which is combined with acc_1 to evaluate whether the current model is better than the best so far (Lines 10-13). We note that

the **CurrentEstimator** at Line 9 corresponds to the lower limit of the 95% confidence interval of acc . To update S , TCT first adds a set U containing $(1 - \alpha)|S|$ random examples from A_1 . Then, it adds $\alpha|S|$ examples from $A_2 \cup (A_1 \setminus U)$, prioritizing the wrong ones.

Algorithm 1 TCT (m_0 : integer; L : Learner; α : real parameter; P : pool of examples; T : time limit)

```

1:  $S \leftarrow$  set of  $m_0$  random examples from  $P$ 
2: repeat
3:    $M \leftarrow$  model trained by learner  $L$  on set  $S$ 
4:    $A_1 \leftarrow$  set of  $|S|$  random examples from  $P$  that have not been selected so far
5:    $acc_1 \leftarrow \text{Classify}(M, A_1)$ 
6:    $A_2 \leftarrow$  set of  $\alpha|S|acc_1/(1 - acc_1)$  random examples from  $P$  that have not been selected so far
7:    $acc_2 \leftarrow \text{Classify}(M, A_2)$ 
8:    $acc \leftarrow (acc_1|A_1| + acc_2|A_2|)/(|A_1| + |A_2|)$ 
9:   CurrentEstimator  $\leftarrow acc - 1.96\sqrt{\frac{acc(1-acc)}{|A_1|+|A_2|}}$ 
10:  if ElapsedTime  $\leq T$  AND (first round OR CurrentEstimator  $>$  BestEstimator) then
11:    BestModel  $\leftarrow M$ 
12:    BestEstimator  $\leftarrow$  CurrentEstimator
13:  end if
14:   $U \leftarrow$  set of  $(1 - \alpha)|S|$  random examples from  $A_1$ 
15:   $V \leftarrow$  list of examples in  $A_2 \cup (A_1 \setminus U)$  with the wrong ones (w.r.t.  $M$ ) appearing before the
    correct ones.
16:   $W \leftarrow \alpha|S|$  first examples from  $V$ 
17:   $S \leftarrow S \cup U \cup W$ 
18: until ElapsedTime  $\geq T$ 
19: Return BestModel.

```

Some observations are in order:

- If the parameter α is very small, then TCT becomes similar to pure random sampling. In contrast, if α is large, the Learner is guided to learn a model that relies on a distribution that may be significantly different from the real one, which is not in line with Principle 3. Indeed, we present an example in Appendix B.1, where using a large α is problematic. In addition, a large value of α may have a negative impact on the running time due to the Lines 6 and 7. The results from our experiments suggest that $\alpha = 0.2$ works well in practice and also that the method is robust to moderate variations of this parameter.
- TCT is more suitable for the typical scenario where the classification time per example is (much) smaller than the training time per example. For scenarios where this property does not hold, TCT may spend a large amount of time classifying examples and, hence, end up with a model trained on a relatively small set.
- Since TCT classifies several new examples in each round, it obtains, at no additional cost, an unbiased estimation of the real accuracy of M . Thus, it is reasonable to use this estimation for selecting the model (Lines 10-13). The reason why TCT picks the lower bound of the 95% confidence interval (line 9) rather than the estimated accuracy is because in the first rounds it uses few examples and, as a consequence, the variance of the estimation is high.
- We have assumed that the pool P is large enough so that we can always sample examples that have not been considered so far (Lines 4 and 6). In practice, this does not necessarily occur. In that case, when TCT reaches the point in which all the examples have already been classified, it starts to use examples that have not been added to set S yet.

3 Experimental Study

In our first set of experiments, we compare the algorithm TCT with two other teachers, namely **Double** and **OSCT** [Dasgupta et al., 2019, Cicalese et al., 2020] that are briefly described below.

- **Double** is a Teacher that at each round i sends new $m_0 2^i$ randomly selected examples to the Learner, where m_0 is the number of examples used to train the first model. Next, the Learner returns a model trained on all examples received so far and, then, round $i + 1$ starts. The model returned by **Double** is the last one built within the given time limit;
- **OSCT** keeps a weight for each example in the training set. At each round, it receives a new model (hypothesis) from the Learner and uses it to classify all the examples in the training set. Next, it repeatedly doubles the weights of the wrong examples until their sum exceeds 1. At this point, it samples $O(\log n)$ examples following a distribution induced by these weights, where n is an estimate of the number of effective hypotheses in the Learner’s class. The Learner receives these sampled examples and use them to update its current model.

Double can be viewed as an adaptation of random sampling for the TCL task and, thus, we understand that it is a very natural baseline. **OSCT** is a recent method for teaching black-box learners that focuses on minimizing the size of the teaching set. In our experiments we employed the implementation discussed in [Cicalese et al., 2020]. A pseudo-code of the implementation can be found on Appendix A.3

We also compare TCT with Stochastic Gradient Descent (SGD) based training, a widely used strategy for online training. For this purpose, we use the class `SGDClassifier` from `Scikit-Learn`, with its default parameters.

To compare TCT with **Double** and **OSCT**, we considered 5 Learners: **LGBM**, **Random Forest**, **Decision Tree**, **SVM** and **Logistic Regression**. In our comparison with SGD, we only considered the last two Learners since it is not clear how to (directly) train the others via SGD.

LGBM belongs to the family of Gradient Boosting methods and it is very popular on contests like Kaggle. **Random Forest** is also widely used. We selected **Decision Trees** with small depth as a representative of interpretable methods, while **SVM** and **Logistic Regression** were selected as representatives of linear classifiers.

Our learners were implemented using Python, version 3.8.5, with the libraries `numpy` (1.20.1), `pandas` (1.2.2), `lightgbm` (3.1.1), `scikit-learn` (0.24.1), `scipy` (1.6.1). We use the default parameters for all learners but for **Decision Trees**, where we set `min_samples_split` = 30 and `max_depth` = 5 to build interpretable trees and for **Random Forest** where we set `min_samples_split` = 30 to prevent very long running time and, hence, limiting our experimental study. To obtain a **SVM** and a **Logistic Regression** classifiers via SGD we set the parameter `loss` from class `SGDClassifier` to `hinge` and `log`, respectively.

We considered 20 datasets in our experiments, whose main features are shown in Table 1. The experiments from Section 3.1-3.3 were executed using processor Core i9-7900X 3.3GHz, with 128GB RAM DDR4 and Windows 10, while those from Section A.6 were executed with the following settings: Core i7-4790 3.60GHz, 32GB RAM DDR3, Ubuntu 20.04.3 LTS. For timing we used the method `timeit.default_timer` from `timeit` library. Our code can be found at <https://github.com/sfilhofreitas/TimeConstrainedLearning>. More details about the datasets and learners can be found in Appendix A.

For each combination of dataset \mathcal{D} and Learner \mathcal{L} , we calculated the time $t_{\mathcal{D},\mathcal{L}}$ required by \mathcal{L} to build a classification model for the whole dataset \mathcal{D} (averaged over 4 runs) and kept the pairs $(\mathcal{D}, \mathcal{L})$ for which $t_{\mathcal{D},\mathcal{L}}$ is at least 10 seconds. We define $Valid(\mathcal{L}) := \{\mathcal{D} \mid t_{\mathcal{D},\mathcal{L}} > 10\}$.

Table 1: Datasets. m : size of training set, d : number of attributes; k : number of classes

Dataset	m	d	k
vehicle_sensIT	68969	100	2
MiniBooNE	91044	50	2
SantanderCustomer	140000	200	2
BNG_spambase	699993	171	2
BNG_spectf_test	700000	44	2
Diabetes130US	71236	2518	3
BNG_wine	700000	13	3
jannis	58613	54	4
BNG_eucalyptus	700000	95	5
BNG_satimage	700000	36	6
covtype	406708	54	7
volkert	40817	180	10
cifar_10	42000	3072	10
mnist	60000	784	10
BNG_mfeat_fourier	700000	76	10
poker_hand	1000000	85	10
Sensorless_drive	40956	48	11
BNG_letter_5000_1	700000	16	26
GTSRB-HueHist	36287	256	43
aloi	75600	128	1000

For each valid combination $(\mathcal{D}, \mathcal{L})$, we run each of the teachers to build a classification model within time limit $t_{\mathcal{D}, \mathcal{L}}$. During these trainings, whenever a new model was obtained we evaluated it on the testing set (pausing the timing), which allowed us to plot the evolution curves that are presented next. For all experiments we set m_0 , the size of the first set of examples sent to the learner, as 0.5% of the size of the full dataset. In practice, m_0 should be set as the maximum value for which we are confident that a training set with m_0 labeled examples can be trained within the time limit.

3.1 Comparisons with OSCT and Double

Figure 1 shows 5 images, each of them corresponding to a comparison between TCT ($\alpha = 0.2$) and OSCT for each of the Learners. The horizontal axis corresponds to the normalized time limit $t \in [0, 1]$ (as a fraction of $t_{\mathcal{D}, \mathcal{L}}$), and the vertical axis corresponds to the (average) accuracy on testing sets. More precisely, for a Teacher \mathcal{T} and a learner \mathcal{L} , the accuracy associated with the normalized time $t \in [0, 1]$ is given by

$$\sum_{\mathcal{D} \in \text{Valid}(\mathcal{L})} \frac{\text{acc}(\mathcal{T}, \mathcal{L}, \mathcal{D}, t \cdot t_{\mathcal{D}, \mathcal{L}})}{|\text{Valid}(\mathcal{L})|}, \quad (1)$$

where $\text{acc}(\mathcal{T}, \mathcal{L}, \mathcal{D}, t \cdot t_{\mathcal{D}, \mathcal{L}})$ is the accuracy of the model obtained by Teacher \mathcal{T} , with learner \mathcal{L} , over dataset \mathcal{D} when the time limit for training is $t \cdot t_{\mathcal{D}, \mathcal{L}}$.

As an example, consider a dataset where training an SVM with all datapoints took 1000 secs. Then, Figure 1 "SVMLinear" at x -axis equal to 0.6, for example, shows the accuracy of the algorithms TCT and OSCT when given $0.6 \cdot 1000 = 600$ secs of execution time (actually this figure shows the average of such accuracy over all datasets).

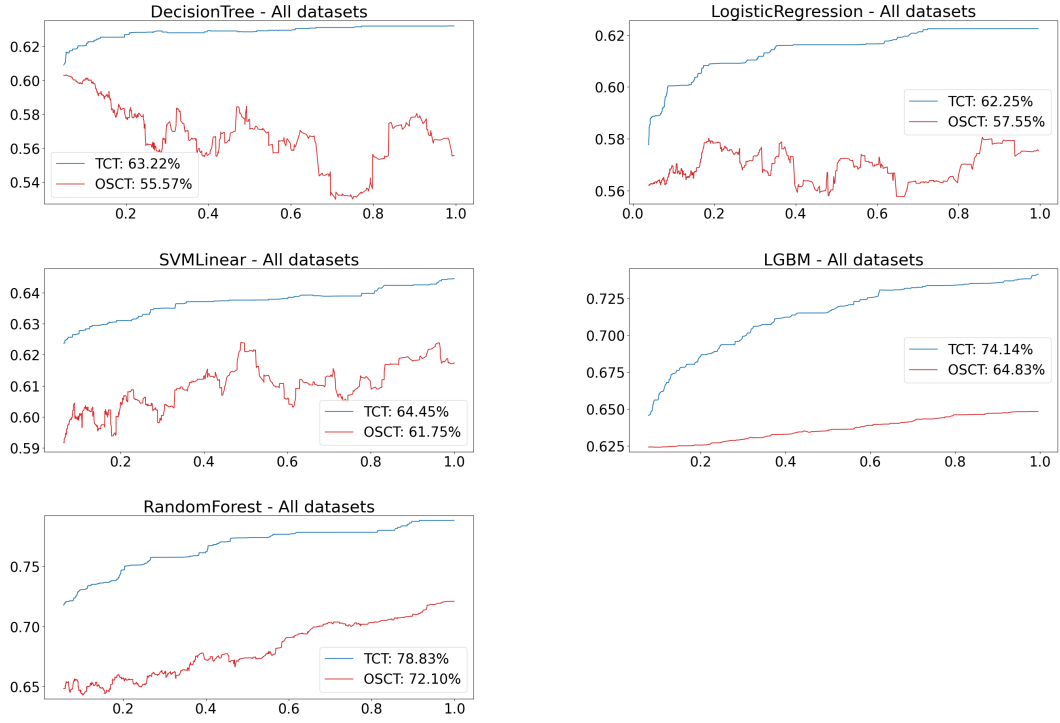


Figure 1: Average accuracies on testing set along normalized time for TCT and OSCT. The numbers next to the labels are their average accuracies at the last normalized time limit $t = 1$. The initial guess for n is 2.

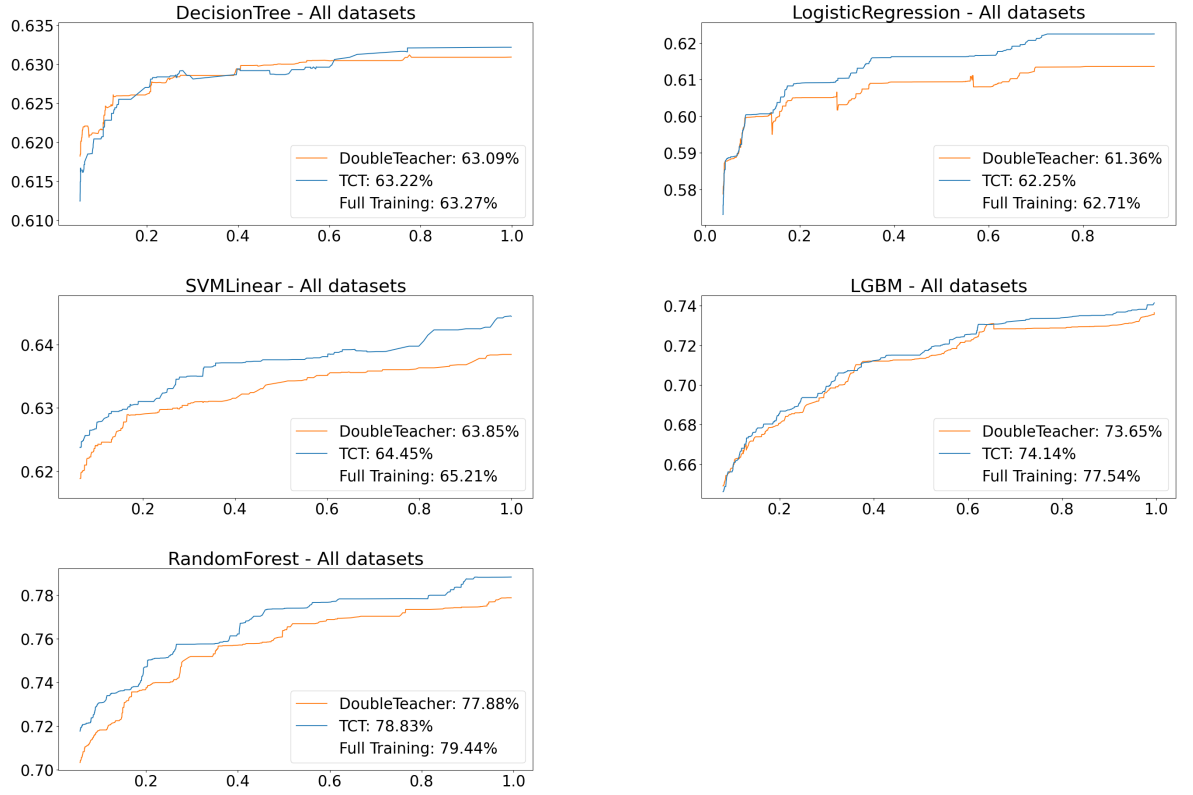


Figure 2: Average accuracies on testing set along normalized time for TCT, Double and OSCT. The numbers next to the labels are their average accuracies at the last normalized time limit $t = 1$.

Table 2: Additional Relevant Statistics

Learner	# Datasets	TCT vs Double		TCT vs Full		% of Training Set	
		Win	Loss	Win	Loss	Double	TCT
LGBM	15	9	1	6	7	24.1	17.1
Random Forest	20	13	1	4	4	45.0	45.5
SVM	13	7	2	5	5	39.0	34.6
Log. Regression	13	2	2	1	8	29.2	23.3
Decision Tree	7	2	3	2	4	33.4	25.0
Overall	68	33	9	18	28		

We see that TCT presents a huge advantage compared to OSCT. One of the reasons is that OSCT does not manage to build models on large training set since it adds a few examples per round and spends a non-negligible time classifying examples. We tried variations of OSCT to improve its accuracy (see Appendix A.3), but the results did not change significantly.

Figure 2 shows a similar comparison where Double, rather than OSCT, is used. We observe that TCT outperforms Double for all Learners but Decision Trees, where their performances are very similar. We also note that at the final time limit $t_{\mathcal{D},\mathcal{L}}$, for most of the Learners, both TCT and Double are able to reach accuracy comparable to that of training using the whole dataset (label Full Training). The accuracy associated with training on the whole dataset can be thought as what can be achieved by an “oracle” Teacher for TCL that knows beforehand the size of the largest batch of (random) examples that can be trained within a given time limit ($t_{\mathcal{D},\mathcal{L}}$ in this case) and sends such batch to the Learner.

Table 2 shows other relevant statistics at the final time limit $t_{\mathcal{D},\mathcal{L}}$. The multi-column TCT vs Double (resp. TCT vs Full) gives, for each Learner, the number of datasets where the classifier built by TCT outperforms that of Double (resp. Full) with 95% confidence (Section 5.5 of [Mitchell, 1997]). As an example, for LGBM, TCT outperformed Double 9 times and it was outperformed just once. We observe a clear advantage of TCT over Double for LGBM, Random Forest and SVM. For Logistic Regression and Decision Trees these algorithms have similar performance. Perhaps surprisingly, TCT is even competitive against the training with the full dataset for Random Forest and SVM and, thus, also competitive against the aforementioned “oracle” Teacher. The multi-column “% of the Training Set” gives the average of the number of examples, relative to the size of the full training set, employed by the models built by each pair (Teacher, Learner). We observe that TCT builds more accurate models than Double, despite using 15% fewer examples (simple average over the different learners).

3.2 Comparisons with SGD

We compare TCT and SGD for training SVM and Logistic Regression. For that, we use the SGDClassifier module from the sklearn library with its default settings. In each iteration the Learner receives a set (mini-batch) of random labeled examples and SGD, via partial_fit method, is used to update the classification model. This is repeated as long as we do not reach the time limit (possibly with several passes over the training set).

To choose the size of the mini batch we considered all the possibilities in the set $\{64, 128, 256, 512\}$. The results reported here consider those that achieved the best results, namely 256 for hinge loss (SVM) and 512 for log loss (Logistic Regression).

TCT had a much better performance: for Logistic Regression, with 95% statistical

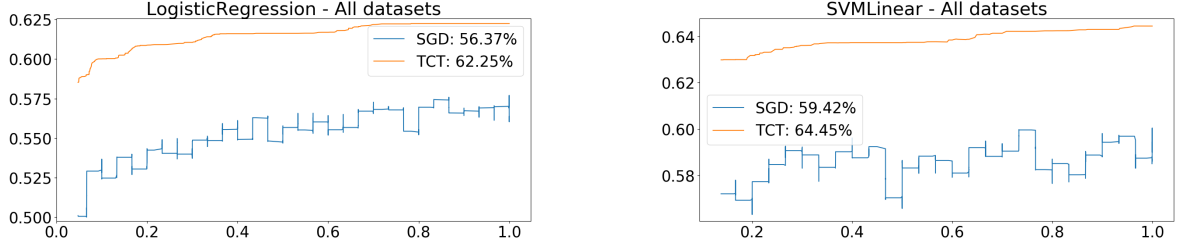


Figure 3: Average accuracies on testing set along normalized time for SGD and TCT. The numbers next to the labels are their average accuracies at the last normalized time limit $t = 1$.

Table 3: Sensibility to α : Win and Losses of TCT over **Double** and average accuracy of TCT

α	Win	Loss	Total	Avg. Accuracy
0.05	24	3	27	69.9
0.1	31	4	35	70.2
0.2	33	9	42	70.3
0.3	37	12	49	70.4
0.45	32	21	53	70.0
0.6	25	35	60	69.5
0.9	19	38	57	68.6

confidence, it outperformed **SGD** in 11 out of 13 datasets and was worse in only 1; for **SVM**, also out of 13 datasets, it was better in 10 and worse in 2. Figure 3 shows the average accuracy over time on the testing set. Additional tables regarding this experiment are given in the appendix A.5.

We note that the advantage of **TCT** may have to do with the fact that the **Scikit-Learn** classes that it uses to train **Logistic Regression** and **SVM** employ optimization techniques and have hyper-parameters that are different from those of **SGD**. That said, the key information revealed by our experiments is that **TCT** obtains much better results than using a very natural alternative for a practitioner, that is, training via the **SkLearn**’s implementation of **SGD** with its default parameters. This confirms the practical appeal of a wrapper for Time Constrained Learning that relies on our proposed method.

3.3 Sensibility to hyper-parameter α

Finally, we performed some experiments to understand the impact of parameter α that controls the number of wrong examples allocated at each round. Table 3 shows the number of wins and losses of **TCT** over **Double** for different values of α . In general, the larger the value of α the larger the number of pairs $(\mathcal{L}, \mathcal{D})$, given by column **Total**, for which there is a difference with 95% confidence between the accuracy of **TCT** and that of **Double**. This is not surprising since the smaller the α the larger the intersection between the training sets employed by **TCT** and **Double**. The most interesting result is the deterioration of the average accuracy with the growth of α , which is in line with Principle 3. These experiments suggest that **TCT** is robust with respect to the choice of α : one can set it safely on the interval $[0.05, 0.3]$ and expect consistent gains.

Table 4: Comparison Between TCT and variation of TCT that selects examples via active learning

	# Datasets	Win	Loss
LGBM	15	7	4
Random Forest	20	13	1
Log. Regression	15	0	4
Decision Tree	8	1	5

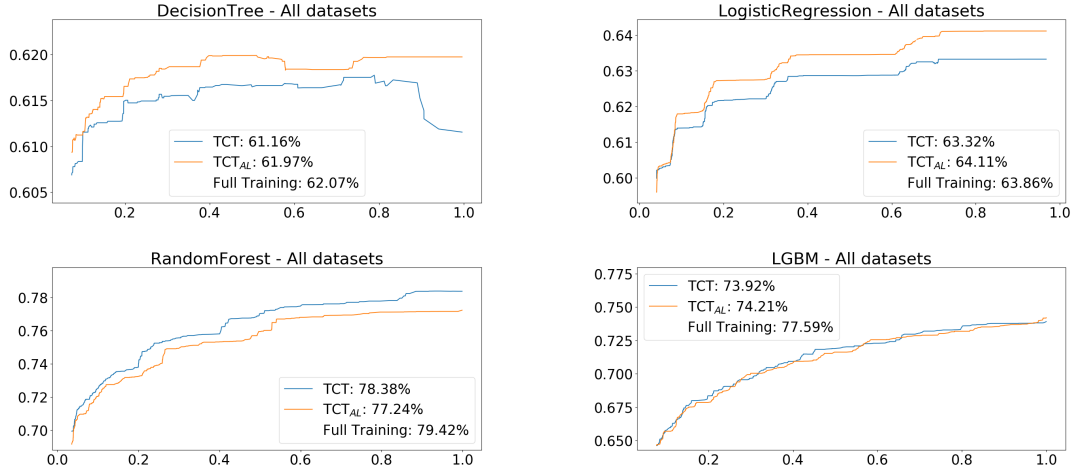


Figure 4: Average accuracies on testing set along normalized time for TCT and TCT_{AL} . The numbers next to the labels are their average accuracies at the last normalized time limit $t = 1$.

3.4 Selecting Examples via Active Learning

We also evaluate the possibility of replacing the wrong examples in our strategy with examples selected via uncertainty sampling, a classic active learning strategy.

More precisely, we consider the following variation of TCT, dubbed TCT_{AL} . At the beginning of each round, TCT_{AL} uses the current training set to build a new model. Next, this model is employed to classify the examples of a set S , containing $2i$ random examples, where i is the number of examples employed by the last trained model. Then, TCT_{AL} selects a set $S' \subset S$ containing i examples and adds it to the current training set, which triggers the beginning of a new round. The set S' is built by picking the $\alpha \cdot i$ most uncertain examples from S and $(1 - \alpha) \cdot i$ additional random examples from S , where the uncertainty of an example is given by the difference between the probabilities (assigned by the Learner) to the two most probable classes. In our experiments we used $\alpha = 0.2$ for both TCT and TCT_{AL} .

The results are presented in Table 4. The “standard” TCT is clearly better for **Random Forest**, it has some advantage for **LGBM** and it is worse for both **Log. Regression** and **Decision Tree**. We do not have results for **SVM** because the probabilities required for the strategy are not directly available.

Figure 4 shows the normalized accuracy of TCT and TCT_{AL} over time. Additional tables can be found in Appendix A.6.

4 Theoretical Analysis

To complement our work, we present theoretical results that provide some insight on how the parameter α affects the performance of TCT. The main conclusions of this analysis are: a large α can be harmful; with a small α TCT is never much worse than random sampling and, in some situations, it is significantly better. To carry out the analysis we consider the following setting:

- (i) The training algorithm has access to as many labeled examples from the unknown distribution μ as it wants;
- (ii) The training time of the learner \mathcal{L} can be approximated by a non-sublinear function that does not grow very fast, that is, training \mathcal{L} with m examples takes time $m^k f(m)$, where k is a small positive integer and f is a sublinear non-decreasing function;
- (iii) The learner is an empirical risk minimizer (ERM), that is, it returns a hypothesis h in its hypothesis class \mathcal{H} that makes the smallest number of mistakes in the set of examples S it receives;
- (iv) It takes no time to pick an example and have it classified by the learner \mathcal{L} using its current classification model

Assumption (i) can be approximated by having a huge dataset \mathcal{D} sampled from μ and is exactly the one that motivates our research, since for smaller datasets Time-Constrained Learning is not particularly relevant. The second assumption is also reasonable in the sense that most of the known learning methods neither take sublinear time nor have a high time complexity. Assumption (iii) is a standard assumption employed to perform theoretical analyses.

With regards to the last item, it is motivated by the quite common situation in which the classification time is very small compared to the training time (e.g. decision trees and SVM's). In fact, we could have replaced assumption (iv) by a weaker one but that would compromise the clarity of the presentation without changing our main conclusions regarding the impact of the parameter α .

To understand the accuracy of the models obtained by TCT, we analyze a stripped-down version of the algorithm denoted by **TCTbase**; we compare it against the batch teacher **TBatch** that receives a time limit T and simply sends to the learner in one round the largest number of random examples from μ that the learner can be trained over within time T . As TCT, in each round **TCTbase** sends to the learner a $(1 - \alpha)$ fraction of examples from the original distribution and an α fraction of examples where the learner is currently wrong. The pseudo-code of **TCTbase** is presented below.

Algorithm 2 **TCTbase** (α : real parameter; T : time limit)

Start with a random set S_0 with a single labeled example from μ .

For each round i (starting with $i = 1$):

1. Run the Learner on examples S_i . Get back hypothesis h_i
2. Get $(1 - \alpha) 2^i$ unbiased labeled examples from μ .

Then repeatedly sample from μ until getting $\alpha 2^i$ labeled examples (x, y) where h_i is wrong, namely $h_i(x) \neq y$.

3. Set S_{i+1} as S_i plus these new 2^i samples

Return the last hypothesis h_i found within the time limit T

We note that due to assumption (iv), the second step of the algorithm incurs negligible running time. If h_i has a small error, however, this assumption becomes unrealistic since we would need to sample a huge number of examples from μ to obtain $\alpha 2^i$ wrong ones. This issue can be fixed by stopping the algorithm as soon as it obtains a hypothesis with error at most some ε (e.g. $< 1\%$). This only incurs an additional $+\varepsilon$ in the bounds of Theorems 1 and 2 presented further in this section.

Before presenting our results, we briefly recall some definitions from statistical learning. For an unknown distribution μ over labeled examples $\mathcal{X} \times \mathcal{Y}$, the true error of a classifier h is

$$\text{err}(h) := \Pr_{(X,Y) \sim \mu} (h(X) \neq Y).$$

Given a set of samples $S = ((X_1, Y_1), \dots, (X_m, Y_m))$ from μ , the sample error of h is

$$\text{err}_S(h) := \frac{1}{m} \sum_{i=1}^m \mathbf{1}(h(X_i) \neq Y_i).$$

Let \mathcal{H} be the Learner's set of hypotheses. In the *realizable* setting all the labels of the samples are given by a $h^* \in \mathcal{H}$, namely $y = h^*(x)$ for every (x, y) in the support of μ . If the setting is not realizable then it is called *agnostic*.

Fallback analysis. As discussed in Principle 3 above, there is a concern that by including “wrong examples” we bias the distribution of the examples sent to the learner and compromise the real accuracy of the hypothesis learned. Indeed, we construct a small instance where **TCTbase** set with a large value of α (thus, sending a large fraction of “wrong examples”) is significantly worse than **TBatch**; see Appendix B.1 for details.

Despite this difficulty, we prove that even in the worst case **TCTbase** returns a hypothesis with the same accuracy as **TBatch** as long as: the percentage α of wrong examples is not so big and it is given a slightly bigger time limit (again, crucially this includes the total time consumed by the Learner during the executions of these algorithms).

To make this concrete, we discuss here the realizable case. Notice that in this case any ERM learner trained by **TBatch** returns some hypothesis h with zero sample error. Let m_T be the number of examples that **TBatch** sends to the learner under time limit T and let $\varepsilon_T = \varepsilon_T(\mathcal{H}, \mu, \delta)$ be the smallest value such that

$$\Pr \left(\exists h \in \mathcal{H} \text{ such that } \text{err}_S(h) = 0 \text{ but } \text{err}(h) > \varepsilon_T \right) < \delta,$$

where the probability is taken over sets S of m_T examples sampled according to μ . In words, ε_T is the best provable guarantee in terms of error for **TBatch**, with $1 - \delta$ probability, when the time limit is T .

Theorem 1. *Given $\delta \in (0, 1)$ and time limit T , let ε_T be defined as above. Under assumptions (i)-(iv), in the realizable setting, with probability at least $1 - \delta$, **TCTbase** returns in time at most $T \cdot 2(\frac{2}{1-\alpha})^{k+1}$ a classifier with error at most ε_T .*

Proof. Again let m_T be the number of samples sent by **TBatch** when the time limit is T . Moreover, let \hat{i} be the first round in which **TCTbase** sends at least $\frac{1}{1-\alpha} m_T$ samples, that is, $\frac{1}{1-\alpha} m_T \leq 2^{\hat{i}} \leq \frac{2}{1-\alpha} m_T$. Due to the assumptions (ii) and (iv), the time **TCTbase** takes to finish

round \hat{i} is at most

$$\begin{aligned} \sum_{i=0}^{\hat{i}} \left((2^i)^k \cdot f(2^i) \right) &\leq f(2^{\hat{i}}) \frac{(2^{\hat{i}+1})^k}{2^k - 1} \leq 2f(2^{\hat{i}})2^{k\hat{i}} \\ &\leq 2f\left(\frac{2}{1-\alpha}m_T\right) \left(\frac{2}{1-\alpha}m_T\right)^k \leq 2\left(\frac{2}{1-\alpha}\right)^{k+1} T, \end{aligned} \quad (2)$$

the last inequality holding because of the sublinearity of f and because $(m_T)^k \cdot f(m_T) \leq T$, by definition of m_T .

Let S be the set of samples sent by TCTbase to the Learner at round \hat{i} , and let h be the returned hypothesis. The choice of \hat{i} guarantees the existence of a subset U of S containing $(1-\alpha)|S| \geq m_T$ samples that were drawn unbiasedly from μ . Since we are in the realizable case, we have that $\text{err}_U(h) = 0$ and by definition of ϵ_T the probability of h having true error at most ϵ_T is at least $1 - \delta$. \square

We shall note that when α is small, the time overhead is approximately 2^{k+2} , which is not big due to the assumption that k is small. A similar result for the agnostic setting is presented in the appendix B.2.

An almost exponential speedup. Importantly, not only TCTbase always takes time similar to that of TBatch as long as α is not big, but we also show that in some cases TCTbase is almost exponentially faster.

We consider the classic problem of learning a threshold function on the real line \mathbb{R} (so the classifiers are of the type $h(x) = 1$ if $x \geq v$ and $h(x) = -1$ if $x < v$, for $v \in \mathbb{R}$), in the realizable case. This is the canonical example where Active Learning gives an exponential improvement in sample complexity compared to standard PAC learning [Dasgupta, 2009].

The next theorem (proof in Appendix B.3) shows that even though the teaching algorithm TCTbase is not tailored to this problem, it also achieves an almost exponential speedup. Recall that $2^{O(\sqrt{\log x})}$ is asymptotically smaller than x^c for any constant $c > 0$.

Theorem 2 (Improvement over TBatch). *Consider $\varepsilon, \delta \in (0, 1)$. Let T_{TBatch} be the smallest time limit that guarantees that TBatch returns a hypothesis with error at most ε with probability at least $1 - \delta$ for all realizable instances of the problem of learning a threshold function on the real line, and define T_{TCTbase} analogously.*

Then, $T_{\text{TCTbase}} \leq 2^{c\sqrt{\log T_{\text{TBatch}}}}$, where c is a constant that depends on k, α, δ .

5 Concluding Remarks

We introduced the time-constrained learning task and the algorithm TCT for tackling it. Our algorithm relies on methodologically sound ideas, is supported by theoretical results, and, most importantly, experiments including 20 datasets, 5 different Learners, and two other baselines suggest that it is a good choice for the time-constrained learning task. Due to its simplicity and generality, TCT could be easily implemented as a wrapper in machine learning libraries to address Time Constrained Learning

As a future work it would be interesting to investigate ways of mixing examples selected by TCT and active learning strategies. This seems to be a promising direction as indicated by the experiments presented in Section 3.4.

References

- Ayumi Shinohara. Teachability in computational learning. *New Generation Comput*, 8(4): 337–347, 1991.
- Xiaojin Zhu, Adish Singla, Sandra Zilles, and Anna N. Rafferty. An overview of machine teaching. *CoRR*, abs/1801.05927, 2018. URL <http://arxiv.org/abs/1801.05927>.
- Sally A. Goldman and Michael J. Kearns. On the complexity of teaching. *J. Comput. Syst. Sci*, 50(1):20–31, 1995.
- Francisco S. Melo, Carla Guerra, and Manuel Lopes. Interactive optimal teaching with unknown learners. In Jérôme Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 2567–2573. *ijcai.org*, 2018. doi: 10.24963/ijcai.2018/356. URL <https://doi.org/10.24963/ijcai.2018/356>.
- Weiyang Liu, Bo Dai, Xingguo Li, Zhen Liu, James M. Rehg, and Le Song. Towards black-box iterative machine teaching. In Jennifer G. Dy and Andreas Krause 0001, editors, *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 3147–3155. PMLR, 2018. URL <http://proceedings.mlr.press/v80/>.
- Sanjoy Dasgupta, Daniel Hsu, Stefanos Poulis, and Xiaojin Zhu. Teaching a black-box learner. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 1547–1555. PMLR, 2019. URL <http://proceedings.mlr.press/v97/>.
- Ferdinando Cicalese, Sergio Filho, Eduardo Sany Laber, and Marco Molinaro. Teaching with limited information on the learner’s behaviour. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 2016–2026. PMLR, 2020. URL <http://proceedings.mlr.press/v119/cicalese20a.html>.
- Rati Devidze, Farnam Mansouri, Luis Haug, Yuxin Chen, and Adish Singla. Understanding the power and limitations of teaching with imperfect knowledge. In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 2647–2654. *ijcai.org*, 2020. doi: 10.24963/ijcai.2020/367. URL <https://doi.org/10.24963/ijcai.2020/367>.
- Noga Alon, Baruch Awerbuch, Yossi Azar, Niv Buchbinder, and Joseph Naor. The online set cover problem. *SIAM J. Comput*, 39(2):361–370, 2009.
- Charles X. Ling Jun Du. Active teaching for inductive learners. In *Proceedings of the Eleventh SIAM International Conference on Data Mining, SDM 2011, April 28-30, 2011, Mesa, Arizona, USA*, pages 851–861. SIAM / Omnipress, 2011. doi: 10.1137/1.9781611972818.73. URL <https://doi.org/10.1137/1.9781611972818.73>.
- Adish Singla, Ilija Bogunovic, Gábor Bartók, Amin Karbasi, and Andreas Krause. Near-optimally teaching the crowd to classify. In *ICML*, volume 32 of *JMLR Workshop and Conference Proceedings*, pages 154–162. JMLR.org, 2014. URL <http://proceedings.mlr.press/v32/>.

- Yuzhe Ma, Robert Nowak, Philippe Rigollet, Xuezhou Zhang, and Xiaojin Zhu. Teacher improves learning by selecting a training subset. In Amos J. Storkey and Fernando Pérez-Cruz, editors, *International Conference on Artificial Intelligence and Statistics, AISTATS 2018, 9-11 April 2018, Playa Blanca, Lanzarote, Canary Islands, Spain*, volume 84 of *Proceedings of Machine Learning Research*, pages 1366–1375. PMLR, 2018. URL <http://proceedings.mlr.press/v84/ma18a.html>.
- Yuxin Chen, Adish Singla, Oisín Mac Aodha, Pietro Perona, and Yisong Yue. Understanding the role of adaptivity in machine teaching: The case of version space learners. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 1476–1486. 2018.
- Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009. URL <http://axon.cs.byu.edu/~martinez/classes/778/Papers/settles.activelearning.pdf>.
- Robert E. Schapire. A brief introduction to boosting. In Thomas Dean, editor, *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI 99, Stockholm, Sweden, July 31 - August 6, 1999. 2 Volumes, 1450 pages*, pages 1401–1406. Morgan Kaufmann, 1999. URL <http://ijcai.org/Proceedings/99-2/Papers/103.pdf>.
- Sanjoy Dasgupta. The two faces of active learning. In Ricard Gavaldà, Gábor Lugosi, Thomas Zeugmann, and Sandra Zilles, editors, *Algorithmic Learning Theory*, pages 1–1, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. ISBN 978-3-642-04414-4.
- Tom Mitchell. *Machine Learning*. McGraw-Hill Education, 1997.
- David A. Freedman. On tail probabilities for martingales. *Annals of Probability*, 3:100–118, 1975. doi: 10.1214/aop/1176996452.
- Martin Anthony and Peter L. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, 1999. doi: 10.1017/CBO9780511624216.

A Experimental Study: Additional Details

A.1 Dataset transformations

We performed some transformations on the datasets.

- Each dataset was randomly shuffled.
- Each dataset (with size m) was split into a *training set* (with size $0.7 \cdot m$) and a *test set* (with size $0.3 \cdot m$). The split ensures that both sets have (roughly) the same class distribution as the original set ¹.
- Each non-numerical feature with $n_{categories}$ possible values were converted into $n_{categories}$ binary features, with one of them 1, and all others 0.
- Each numerical feature was standardized.

¹The *minist* dataset were already split, so the relative sizes of the training set and the dataset in this case are not 0.7 and 0.3

A.2 Dataset sources

Most of the datasets were obtained from the OpenML Repository, the UCI Machine Learning Repository and Kaggle. Most of the datasets have the “Public Domain licence type”. Below we make some additional citation requests:

- Diabetes130US: Beata Strack, Jonathan P. DeShazo, Chris Gennings, Juan L. Olmo, Sebastian Ventura, Krzysztof J. Cios, and John N. Clore, “Impact of HbA1c Measurement on Hospital Readmission Rates: Analysis of 70,000 Clinical Database Patient Records,” BioMed Research International, vol. 2014, Article ID 781670, 11 pages, 2014.
- covtype: copyright for Jock A. Blackard and Colorado State University.
- vehicle_sensIT: M. Duarte and Y. H. Hu.
- MiniBooNE: B. Roe et al., ‘Boosted Decision Trees, an Alternative to Artificial Neural Networks’ <https://arxiv.org/abs/physics/0408124>, Nucl. Instrum. Meth. A543, 577 (2005).
- cifar_10: Alex Krizhevsky (2009) Learning Multiple Layers of Features from Tiny Images, Tech Report.
- GTSRB-HueHist: <https://www.openml.org/d/41990>
- aloi: <http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass.html>

A.3 TCT \times OSCT - additional comparisons

Algorithm OSCT

Input: Set of m examples \mathcal{X} , (guess of) the number of Learner’s hypotheses N

1. Initialize weights $W_e^0 = \frac{1}{2m}$ for all examples $e \in \mathcal{X}$
2. For each round $t = 1, 2, \dots$:
 - Receive hypothesis $H_t \in \mathcal{H}$ from the Learner
 - If H_t is correct in all examples, stop and return H_t
 - **(Weight update)** Double the weights of all wrong examples until their weight adds up to at least 1. That is, define

$$W_e^t = \begin{cases} 2^\ell \cdot W_e^{t-1} & , \text{ if } e \in \text{wrong}(H_t) \\ W_e^{t-1} & , \text{ if } e \notin \text{wrong}(H_t), \end{cases}$$

where ℓ is the smallest non-negative integer such that $W^t(H_t) := \sum_{e \in \text{wrong}(H_t)} W_e^t \geq 1$

- **(Sending examples)** For every example e , let $D_e^t := W_e^t - W_e^{t-1}$ be the weight increase of example e (note $D_e^t = 0$ if H_t is not wrong on e)
Repeat $4 \log N$ times: sample at most one example so that e is sampled with probability D_e^t , and send it to Learner together with its correct label (note that H_t is wrong on this example)
- If no examples were sent, return OSCT $(\mathcal{X}, N^2, \omega)$

Figure 5: Teacher’s algorithm based on an algorithm for the Online Set Covering problem.

The OSCT algorithm corresponds to the algorithm \mathcal{A}_{base} that was discussed and empirically evaluated in [Cicalese et al., 2020]. \mathcal{A}_{base} is a refinement of the algorithm proposed in Dasgupta et al. [2019]. It maintains weights W_e^t over the examples $e \in \mathcal{X}$ for each round t . When a new hypothesis h comes from the Learner, the Teacher verifies whether h makes no mistakes on the examples from \mathcal{X} . If so, it accepts h . Otherwise, it increases in exponential fashion the weights of the examples where h fails until the sum of these weights becomes at least 1; then it randomly sends examples to Learner with probability proportional to the increase of the weights of the examples in this round. If no example is sent by the end of the round, the algorithm starts again with a new guess of N . Although not explicitly stated in the pseudo-code, in our use for Time Constraint Learning, OSCT returns the last model trained within the given time limit.

We tested some variations for the OSCT with the aim of improving its performance. First, we increased the initial guess N on the size of the Learner’s class. By doing so we prevent the Teacher sending few examples in the first rounds. Figure 6 shows the results for $N = 2^{0.005m}$, which ensures that the Teacher sends approximately 0.005 m wrong examples per round before the estimation of N is updated. We observe that these new results are very similar to those presented in Figure 1, that is, there was no significant impact.

In another attempt, we adopted the same approach employed by TCT to select the final classification model: among the several models built by OSCT within the time limit, we return the one with the largest lower limit for the (95%) accuracy’s confidence interval. This incurs no additional cost because OSCT, by design, classifies all the examples from the training set to select the new ones that are sent to the Learner. The results for this test are shown in Figure 7, where a significant improvement of OSCT can be observed, in particular with regards to its stability. Despite of this improvement, TCT still outperforms OSCT for all Learners and for every (normalized) time $t \in [0, 1]$.

A.4 Additional Tables and Plots

Table 5 (resp. Tables 6, 7, 8 and 9) shows the average accuracy obtained by LGBM (resp. Random Forest, SVM, Decision Tree and Logistic Regression) on each dataset for TCT ($\alpha = 0.2$), Double, OSCT and the average accuracy obtained using the entire training set (column Full). The results for OSCT refer to the version of the algorithm that returns the model with the best accuracy estimate (Figure 7). The Time Limit column denotes the average time in seconds taken to train with the entire dataset. This is also the time given as a limit for the Teachers. It is noteworthy that some tables have more rows than others because combinations $(\mathcal{D}, \mathcal{L})$ in which the time limit is less than 10 seconds are discarded.

We boldfaced the datasets for which there is a statistical difference (95% of confidence) between the accuracies of Double and TCT. More specifically, we boldface Double (TCT) for dataset \mathcal{D} if the accuracy of Double (TCT) is larger than that of TCT (Double) and

$$|acc_{Dbl} - acc_{TCT}| - 1.645 \sqrt{\frac{acc_{Dbl}(1 - acc_{Dbl})}{m_{test}} + \frac{acc_{TCT}(1 - acc_{TCT})}{m_{test}}} > 0,$$

where m_{test} is the size of the testing set for dataset \mathcal{D} . To calculate the confidence interval we assume that the examples of the testing set are drawn independently from an unknown distribution μ (Chapter 5 of Mitchell [1997]). We have not considered OSCT in this comparison because it is not competitive with the two other Teachers.

Figure 8 shows how the accuracy of both TCT and Double evolve over the normalized time for some datasets. The images for the other datasets can be found in https://github.com/sfilhofreitas/TimeConstrainedLearning/tree/main/experiments/results/graphics_by_dataset.

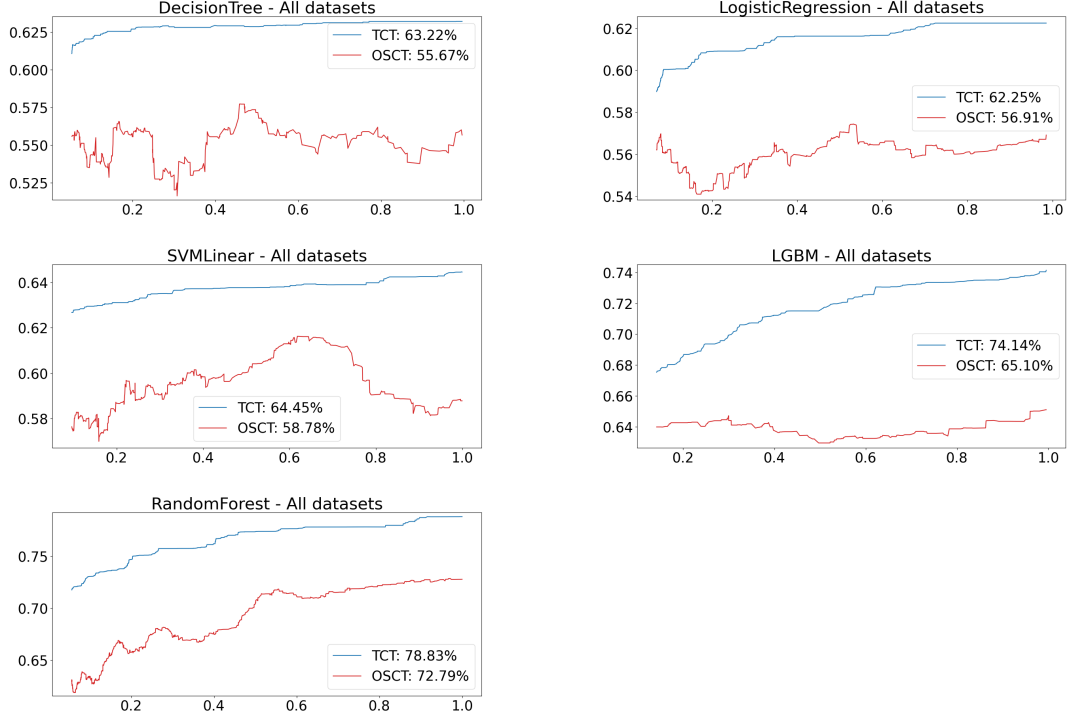


Figure 6: Average accuracies on testing set along normalized time for TCT and OSCT starting with $N = 2^{0.005m}$. The numbers next to the labels are their average accuracies at the last normalized time limit $t = 1$.

Table 5: LGBM accuracies in the testing sets for TCT, Double, OSCT and Full Training for each dataset.

Dataset	Time Limit	TCT	Double	OSCT	Full
BNG_letter_5000_1	198.5	75.9%	74.8%	59.5%	76.7%
poker_hand	48.7	73.0%	72.8%	52.5%	83.3%
SantanderCustomerSatisfaction	18.6	91.3%	90.6%	90.0%	90.7%
BNG_spectf_test	14.5	82.5%	82.4%	77.6%	82.9%
BNG_wine	19.7	96.1%	95.8%	94.2%	96.0%
BNG_eucalyptus	49.8	74.9%	73.9%	63.0%	74.3%
mnist	134.5	97.4%	96.0%	92.5%	97.7%
covtype	15.5	86.6%	85.3%	73.8%	85.3%
cifar_10	793.4	38.6%	38.5%	37.9%	52.8%
volkert	34.0	60.3%	60.1%	52.3%	69.1%
BNG_satimage	77.2	92.2%	91.6%	87.2%	92.1%
Sensorless_drive_diagnosis	11.7	99.3%	98.8%	93.8%	99.9%
GTSRB-HueHist	258.0	38.9%	39.9%	33.0%	57.6%
BNG_mfeat_fourier	214.6	93.7%	92.9%	86.4%	93.4%
aloi	638.1	11.6%	11.4%	4.6%	11.3%

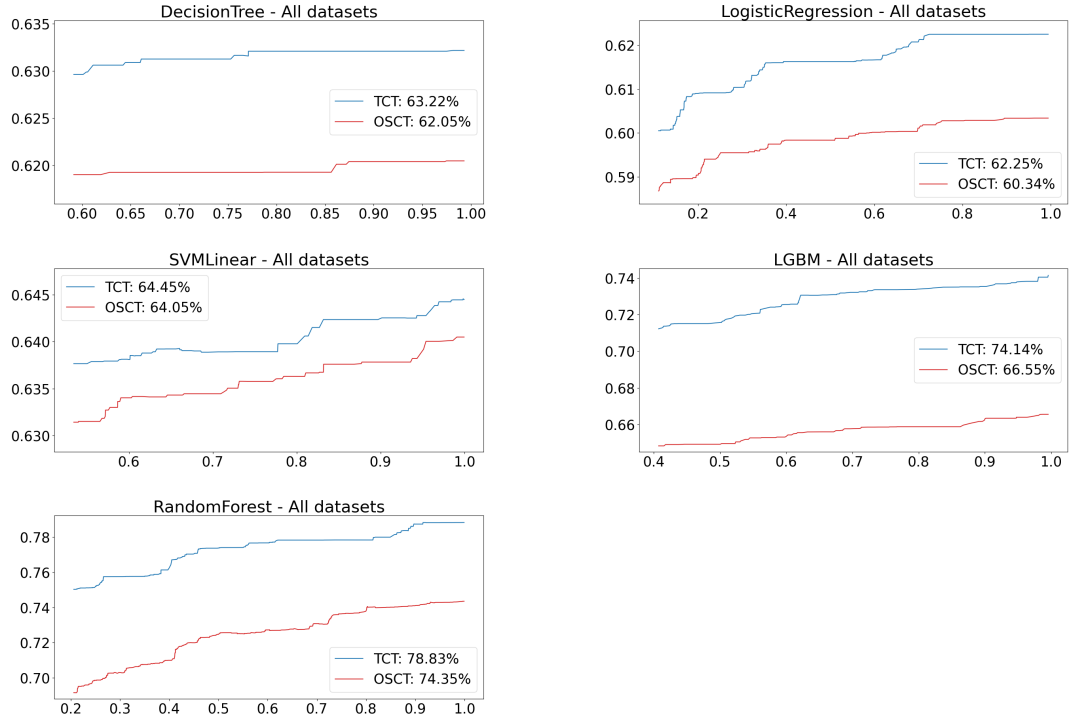


Figure 7: Average accuracies on testing set along normalized time for TCT and OSCT which returns the model with the best accuracy estimate. The numbers next to the labels are their average accuracies at the last normalized time limit $t = 1$.

Table 6: Random Forest accuracies in the testing sets for TCT, Double, OSCT and Full Training for each dataset.

Dataset	Time Limit	TCT	Double	OSCT	Full
Diabetes130US	76.4	58.5%	58.4%	55.1%	59.0%
BNG_letter_5000_1	231.9	71.9%	70.1%	68.5%	72.9%
poker_hand	277.5	92.2%	91.8%	92.3%	92.2%
SantanderCustomerSatisfaction	485.8	90.0%	90.0%	88.4%	90.0%
BNG_spectf_test	781.3	80.0%	79.0%	78.0%	79.2%
BNG_wine	417.6	95.6%	95.5%	95.6%	95.6%
vehicle_sensIT	112.2	87.0%	86.7%	83.5%	86.9%
MiniBooNE	59.7	93.9%	92.5%	92.6%	93.2%
BNG_eucalyptus	166.2	69.2%	67.5%	64.9%	69.1%
mnist	32.2	96.3	95.9%	92.7%	96.3%
BNG_spambase	409.6	66.6%	66.7%	66.0%	66.7%
covtype	92.7	93.0%	88.1%	88.7%	92.4%
cifar_10	123.6	42.9%	42.1%	40.8%	45.4%
jannis	36.2	69.6%	68.5%	65.5%	70.0%
volkert	18.9	62.4%	60.6%	61.1%	64.4%
BNG_satimage	680.2	89.7%	88.2%	88.7%	89.0%
Sensorless_drive_diagnosis	13.7	99.8%	99.4%	99.7%	99.8%
GTSRB-HueHist	42.2	47.8%	43.8%	44.2%	47.3%
BNG_mfeat_fourier	1021.6	88.9%	88.4%	87.8%	88.9%
aloi	26.0	81.1%	84.4%	32.8%	90.6%

Table 7: SVM accuracies in the testing sets for TCT, Double, OSCT and Full Training for each dataset.

Dataset	Time Limit	TCT	Double	OSCT	Full
Diabetes130US	81.6	57.3%	57.6%	51.5%	58.1%
BNG_letter_5000_1	105.6	42.8%	41.3%	42.6%	41.3%
poker_hand	17.7	48.1%	50.0%	47.8%	50.0%
MiniBooNE	11.6	90.1%	89.7%	89.8%	90.1%
BNG_eucalyptus	80.3	57.0%	56.4%	54.5%	56.4%
mnist	1320.7	90.4%	89.0%	89.6%	91.7%
BNG_spambase	16.5	66.5%	66.6%	66.1%	66.6%
covtype	123.7	70.8%	70.4%	70.0%	70.5%
volkert	117.9	57.5%	57.3%	55.4%	57.8%
BNG_satimage	33.8	81.6%	80.7%	81.8%	80.7%
Sensorless_drive_diagnosis	156.7	78.9%	73.7%	89.1%	74.3%
GTSRB-HueHist	154.2	14.2%	14.3%	12.3%	27.2%
BNG_mfeat_fourier	71.4	82.7%	83.1%	82.1%	83.2%

Table 8: Decision Tree accuracies in the testing sets for TCT, Double, OSCT and Full Training for each dataset.

Dataset	Time Limit	TCT	Double	OSCT	Full
Diabetes130US	12.9	57.1%	57.0%	54.4%	57.5%
SantanderCustomerSatisfaction	17.7	89.0%	89.7%	86.9%	89.9%
BNG_spectf_test	30.0	77.6%	78.5%	77.4%	78.5%
BNG_eucalyptus	12.5	53.8%	54.2%	52.7%	54.2%
cifar_10	33.6	24.9%	25.2%	24.8%	25.9%
BNG_satimage	25.7	73.8 %	73.0%	74.0%	73.0%
BNG_mfeat_fourier	47.5	66.3%	64.1%	64.0%	63.9%

Table 9: Logistic Regression accuracies in the testing sets for TCT, Double, OSCT and Full Training for each dataset.

Dataset	Time Limit	TCT	Double	OSCT	Full
Diabetes130US	289.6	58.6%	58.7%	56.0%	59.0%
BNG_letter_5000_1	31.5	45.6%	45.8%	44.1%	45.8%
poker_hand	391.5	48.1%	50.0%	47.7%	50.0%
BNG_eucalyptus	35.5	57.8%	57.8%	55.9%	57.9%
mnist	184.7	91.6%	91.3%	91.5%	92.6%
BNG_spambase	34.1	66.4%	66.6%	66.2%	66.6%
covtype	84.0	68.2%	67.6%	62.5%	68.8%
cifar_10	465.0	37.1%	37.6%	33.7%	39.6%
volkert	29.6	57.4%	57.5%	54.4%	58.6%
BNG_satimage	18.6	83.8%	83.7%	83.4%	83.7%
Sensorless_drive_diagnosis	11.3	83.9%	70.0%	81.7%	78.8%
GTSRB-HueHist	134.9	26.3%	26.6%	24.6%	29.1%
BNG_mfeat_fourier	46.1	84.4%	84.6%	82.8%	84.8%

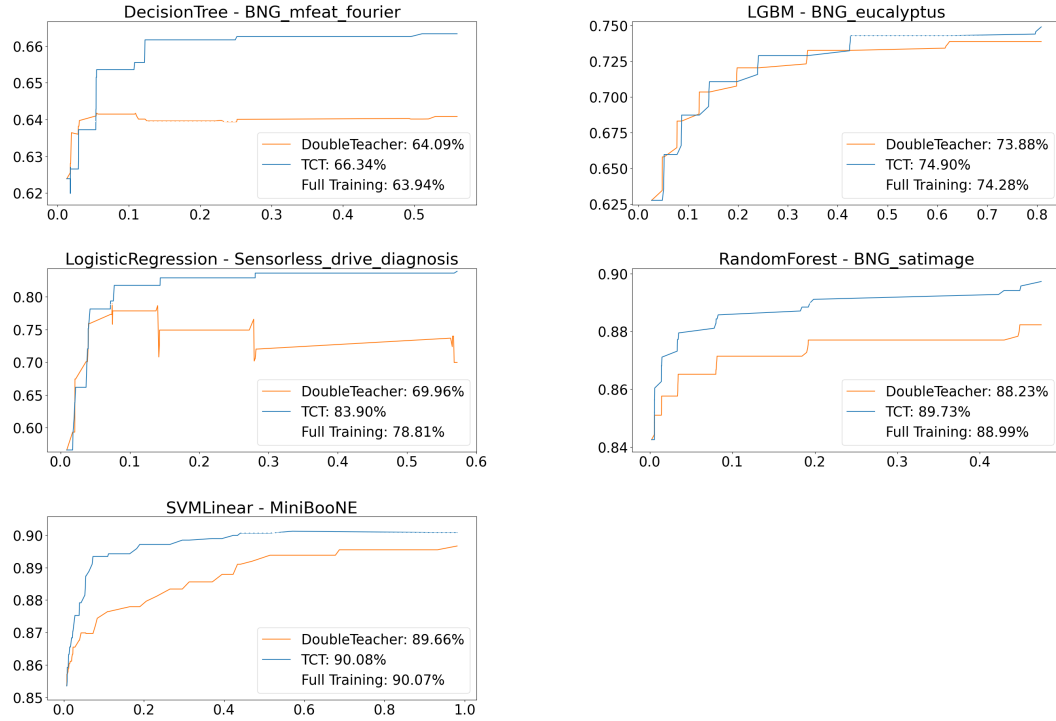


Figure 8: Average accuracies on testing set along normalized time for TCT and Double per dataset. The numbers next to the labels are their average accuracies at the last normalized time limit $t = 1$.

A.5 TCT \times SGD - additional tables

Table 10 (resp. Table 11) shows the average accuracy obtained by **SVM** (resp. **Logistic Regression**) on each dataset for TCT ($\alpha = 0.2$) and SGD.

Table 10: SVM accuracies in the testing sets for TCT and SGD.

Dataset	Time Limit	TCT	SGD
Diabetes130US	81.6	57.3 %	53.1%
BNG_letter_5000_1	105.6	42.8%	29.9%
poker_hand	17.7	48.1%	49.4%
MiniBooNE	11.6	90.1%	86.8%
BNG_eucalyptus	80.3	57.0%	53.2%
mnist	1320.7	90.4%	85.3%
BNG_spambase	16.5	66.5%	66.5%
covtype	123.7	70.8%	55.5%
volkert	117.9	57.5%	53.4%
BNG_satimage	33.8	81.6%	80.3%
Sensorless_drive_diagnosis	156.7	78.9%	61.3%
GTSRB-HueHist	154.2	14.2%	16.2%
BNG_mfeat_fourier	71.4	82.7%	81.7%

Table 11: Logistic Regression accuracies in the testing sets for TCT and SGD..

Dataset	Time Limit	TCT	SGD
Diabetes130US	289.6	58.6%	48.2%
BNG_letter_5000_1	31.5	45.6%	43.3%
poker_hand	391.5	48.1%	50.0%
BNG_eucalyptus	35.5	57.8%	55.9%
mnist	184.7	91.6%	87.1%
BNG_spambase	34.1	66.4%	66.5%
covtype	84.0	68.2%	49.2%
cifar_10	465.0	37.1%	25.6%
volkert	29.6	57.4%	54.7%
BNG_satimage	18.6	83.8%	81.3%
Sensorless_drive_diagnosis	11.3	83.9%	69.3%
GTSRB-HueHist	134.9	26.3 %	19.3%
BNG_mfeat_fourier	46.1	84.4%	82.4%

A.6 Selecting examples via active learning - additional tables

Tables 12, 13, 14 and 15 show the average accuracy of TCT and TCT_{AL} for the different Learners and datasets.

Table 12: Logistic Regression accuracies in the testing sets for TCT and TCT_{AL}.

Dataset	Time Limit	TCT	TCT _{AL}
BayesianNetworkGenerator_spambase	45.6	66.4%	66.6%
BNG_eucalyptus	48.3	57.8%	57.9%
BNG_letter_5000_1	45.1	45.6%	46.0%
BNG_mfeat_fourier	80.5	84.4%	84.5%
BNG_satimage	26.3	83.8%	83.8%
BNG_spectf_test	10.6	76.9%	78.4%
cifar_10	938.8	37.1%	37.5%
covtype	110.7	68.2%	67.9%
Diabetes130US	350.7	58.6%	58.8%
GTSRB-HueHist	237.7	26.3%	26.6%
jannis	12.9	63.7%	64.2%
mnist	266.1	91.6%	91.8%
poker_hand	526.5	48.1%	52.1%
Sensorless_drive_diagnosis	16.9	83.9%	88.1%
volkert	48.9	57.4%	57.4%

Table 13: Random Forest accuracies in the testing sets for TCT and TCT_AL.

Dataset	Time Limit	TCT	TCT_AL
aloi	25.1	81.1%	68.7%
BayesianNetworkGenerator_spambase	517.3	66.6%	66.7%
BNG_eucalyptus	194.2	69.2%	68.0%
BNG_letter_5000_1	264.6	71.9%	70.8%
BNG_mfeat_fourier	889.9	88.8%	88.5%
BNG_satimage	864.6	89.3%	89.1%
BNG_spectf_test	829.8	80.0%	79.3%
BNG_wine	289.5	95.7%	95.5%
cifar_10	133.8	42.9%	42.1%
covtype	63.8	92.9%	89.5%
Diabetes130US	69.4	58.5%	58.7%
GTSRB-HueHist	35.0	40.0%	39.8%
jannis	35.5	69.6%	68.8%
MiniBooNE	62.8	93.9%	93.2%
mnist	40.3	96.3%	96.4%
poker_hand	189.8	91.4%	92.1%
SantanderCustomerSatisfaction	460.7	90.5%	90.0%
Sensorless_drive_diagnosis	14.9	99.8%	99.6%
vehicle_sensIT	77.9	86.9%	86.7%
volkert	20.4	62.4%	61.3%

Table 14: LGBM accuracies in the testing sets for TCT and TCT_AL.

Dataset	Time Limit	TCT	TCT_AL
aloi	982.4	12.6%	13.4%
BNG_eucalyptus	57.6	74.9%	74.2%
BNG_letter_5000_1	162.4	74.1%	73.5%
BNG_mfeat_fourier	279.5	93.7%	93.1%
BNG_satimage	83.7	92.2%	91.5%
BNG_spectf_test	18.5	82.6%	82.6%
BNG_wine	20.7	95.9%	95.9%
cifar_10	977.3	42.6%	42.7%
covtype	18.1	86.6%	85.8%
GTSRB-HueHist	270.7	34.0%	36.8%
mnist	109.0	96.4%	96.8%
poker_hand	54.7	73.0%	77.9%
SantanderCustomerSatisfaction	23.0	91.3%	90.9%
Sensorless_drive_diagnosis	14.1	99.3%	99.0%
volkert	36.1	59.7%	59.1%

Table 15: Decision Tree accuracies in the testing sets for TCT and TCT_{AL}.

Dataset	Time Limit	TCT	TCT_{AL}
BNG_eucalyptus	13.3	53.1%	54.5%
BNG_mfeat_fourier	52.1	62.4%	64.0%
BNG_satimage	22.1	73.8%	73.0%
BNG_spectf_test	27.1	77.6%	78.5%
cifar_10	34.0	24.9%	25.4%
Diabetes130US	10.8	57.1%	57.1%
poker_hand	10.6	51.3%	53.5%
SantanderCustomerSatisfaction	18.4	89.0%	89.8%

B Proofs of Section 4

B.1 Sending too many “wrong examples” is bad

We construct a simple instance where the algorithm **TCTbase** set with the “wrong samples” percentage α too high has very poor accuracy.

More precisely, this non-realizable instance has points $\mathcal{X} = \{1, 2, \dots, 6\}$ and a hypothesis class with 4 classifiers $\mathcal{H} = \{h_1, h_2, h_3, \bar{h}\}$ that classify points as +1 as follows (the remaining points are classified as -1):

$$\begin{aligned} h_1 : & \{1, 2, 3, 4\} \\ h_2 : & \{1, 2, 5, 6\} \\ h_3 : & \{3, 4, 5, 6\} \\ \bar{h} : & \{1, 2, 3, 4, 5, 6\}. \end{aligned}$$

The correct classification h^* classifies as +1 the odd points $\{1, 3, 5\}$. Finally, the distribution μ puts $\frac{2}{9}$ probability on each odd number and probability $\frac{1}{9}$ on each even number of \mathcal{X} .

The best classifier in \mathcal{H} is \bar{h} , which has error $\text{err}(\bar{h}) = \frac{1}{3}$, while all other classifiers have error $\text{err}(h_i) = \frac{4}{9}$.

We conducted experiments with **TCTbase** where it sends $\alpha = 90\%$ “wrong” samples in each round. We ran the algorithm for 20 rounds, so at the last round it has a total of $2^{21} - 1 \approx 2,000,000$ samples (there can be/are multiple copies the same sample $(x, h^*(x))$). Over 100 attempts, this algorithm only found the best classifier \bar{h} (in any of its rounds) 8% of the time. In contrast, **TBatch** with 1,000 samples found the best classifier 100% of the time.

Note that at the last round **TCTbase** has $(1 - \alpha) \cdot (2^{21} - 1) \approx 200,000$ unbiased samples, which is orders of magnitude larger than those used by **TBatch**, but still the “wrong samples” caused it to have a very poor performance.

Also notice that when α is large as in this example, the bound from Theorem 3 below is vacuous, due to the last error term.

B.2 The Agnostic Case

Consider the agnostic case. Let $\varepsilon_T^A = \varepsilon_T^A(\mathcal{H}, \mu, \delta)$ be such that **TBatch** with time limit T returns a hypothesis with true error at most that of the best classifier in \mathcal{H} plus ε_T^A with probability at least $1 - \delta$ regardless of the ERM learner; more precisely, let S be a set of m_T random samples from μ and let ε_T^A be the smallest value such that

$$\Pr \left(\sup_{h \in \mathcal{H}} |\text{err}(h) - \text{err}_S(h)| > \frac{\varepsilon_T^A}{2} \right) < \delta.$$

We then have the following guarantee.

Theorem 3. *Given $\delta \in (0, 1)$ and time limit T , let ε_T^A be the $(1 - \delta)$ -probability additional error of **TBatch** as defined above.*

*Under assumptions (i)-(iv), in the agnostic setting, with probability at least $1 - \delta$, **TCTbase** returns in time at most $T \cdot 2(\frac{2}{1-\alpha})^{k+1}$ a classifier h with additional error at most $\varepsilon_T^A + \frac{\alpha}{1-\alpha}$, namely*

$$\text{err}(h) \leq \min_{h' \in \mathcal{H}} \text{err}(h') + \varepsilon_T^A + \frac{\alpha}{1-\alpha}.$$

Proof. Again let m_T be the number of samples sent by **TBatch** when the time limit is T . Moreover, let \hat{i} be the first round in which **TCTbase** sends at least $\frac{1}{1-\alpha}m_T$ samples, that is, $\frac{1}{1-\alpha}m_T \leq 2^{\hat{i}} \leq \frac{2}{1-\alpha}m_T$. Again due to the assumptions (ii) and (iv), inequality (2) shows that **TCTbase** finishes round \hat{i} by time $2(\frac{2}{1-\alpha})^{k+1}T$.

Let S be the set of samples sent by **TCTbase** to the Learner by the end of round \hat{i} , and let \hat{h} be the returned hypothesis. Also let U be the subset of the samples S that were sampled unbiasedly from μ , and $W = S \setminus U$ the remaining ones.

Since U and W make up a $(1-\alpha)$ - and α -fraction of S respectively, we have

$$\text{err}_S(\cdot) = (1-\alpha)\text{err}_U(\cdot) + \alpha\text{err}_W(\cdot). \quad (3)$$

In addition, by definition of \hat{i} we have $|U| \geq (1-\alpha)|S| \geq m_T$, and so using the definition of ε_T^A we have that with probability at least $1-\delta$, for all $h \in \mathcal{H}$

$$|\text{err}(h) - \text{err}_U(h)| \leq \frac{\varepsilon_T^A}{2};$$

in particular, in light of (3), for all $h \in \mathcal{H}$

$$\text{err}_S(h) \leq (1-\alpha) \left[\text{err}(h) + \frac{\varepsilon_T^A}{2} \right] + \alpha \quad \text{and} \quad \text{err}_S(h) \geq (1-\alpha) \left[\text{err}(h) - \frac{\varepsilon_T^A}{2} \right].$$

Under this event, the classifier \hat{h} returned by the ERM learner satisfies the following bound against every $h \in \mathcal{H}$:

$$(1-\alpha)\text{err}(\hat{h}) \leq \text{err}_S(\hat{h}) + (1-\alpha)\frac{\varepsilon_T^A}{2} \leq \text{err}_S(h) + (1-\alpha)\frac{\varepsilon_T^A}{2} \leq (1-\alpha)\text{err}(h) + (1-\alpha)\varepsilon_T^A + \alpha,$$

and so taking an infimum over $h \in \mathcal{H}$ we get

$$\text{err}(\hat{h}) \leq \min_{h' \in \mathcal{H}} \text{err}(h') + \varepsilon_T^A + \frac{\alpha}{1-\alpha}.$$

This concludes the proof. \square

B.3 Proof of Theorem 2

To prove this result we will need to use martingales. Recall that a sequence of random variables X_1, \dots, X_n is a *martingale difference sequence* if $\mathbb{E}[X_i \mid X_1, \dots, X_{i-1}] = 0$ for all i . We need the classic Freedman's Inequality for martingales.

Theorem 4 (Theorem 1.6 of Freedman [1975]). *Consider a martingale difference sequence X_1, \dots, X_n such that $X_i \leq 1$, and its predictable quadratic variation $V := \sum_i \mathbb{E}[X_i^2 \mid X_1, \dots, X_{i-1}]$. Then for any $\lambda \geq 0$ and $v > 0$*

$$\Pr \left(\sum_{i \leq n} X_i \geq \lambda \quad \text{and} \quad V \leq v \right) \leq \left(\frac{v}{\lambda + v} \right)^{\lambda + v} e^\lambda.$$

Proof of Theorem 2. It suffices to prove that

$$T_{\text{TBatch}} \geq \left(\Omega\left(\frac{1}{\varepsilon} \log \frac{1}{\delta}\right) \right)^k$$

and

$$T_{\text{TCTbase}} \leq \left(2^{O(\sqrt{\log 1/\varepsilon})} \cdot \log \frac{1}{\delta} \cdot \left(\frac{1}{\alpha} \right)^{O(1)} \right)^{k+1}.$$

For the lower bound on T_{TBatch} , standard sample complexity lower bound for statistical learning threshold functions (for example Theorem 5.3 of Anthony and Bartlett [1999]) says that there is a realizable instance where $\Omega(\frac{1}{\varepsilon} \log \frac{1}{\delta})$ samples are required by the Learner to obtain error at most ε with probability at least $1 - \delta$. Thus, given assumption (ii), the time limit needs to be at least

$$T_{\text{TBatch}} \geq \left(\Omega\left(\frac{1}{\varepsilon} \log \frac{1}{\delta}\right) \right)^k \cdot f\left(\Omega\left(\frac{1}{\varepsilon} \log \frac{1}{\delta}\right)\right) \geq \left(\Omega\left(\frac{1}{\varepsilon} \log \frac{1}{\delta}\right) \right)^k$$

to allow the Learner to train with these many samples.

We now upper bound T_{TCTbase} by first seeing how many examples and rounds are required to attain error ε with probability at least $1 - \delta$. Recall that S_i is the set of examples that **TCTbase** sends to the learner in round i , and h_i is the hypothesis obtained in return. We again use the notation that $U_i \subseteq S_i$ is the set of samples up to the beginning of round i that were drawn unbiasedly from μ , and $W_i \subseteq S_i$ is the set “wrong examples” drawn thus far. We use μ_x to denote the marginal of μ on $\mathcal{X} = \mathbb{R}$.

Let $v^* \in \mathbb{R}$ be the correct threshold for the given instance. Let $I_i \subseteq \mathbb{R}$ be the maximal interval containing v^* that contains none of the examples S_i (the “uncertainty region” at this time). Define $E_i \subseteq \mathbb{R}$ as the points where h_i ’s classification is incorrect. Notice that this region is an interval that is either “to the left” or “to the right” of v^* (depending whether the threshold used in h_i is to the left or to the right of v^*). Also notice that E_i is contained in I_i : since we are in a realizable instance, the rightmost sample in S_i to the left of v^* (which is the starting point of the open interval I_i) forces the ERM LEARNER to classify all points before it correctly as -1 , and similarly to the points after the end of the open interval I_i .

We define the “left weight” $\text{Lw}(I)$ of the interval I to be the amount of μ_x -mass in the part of the interval that is to the left of v^* , i.e., $\text{Lw}(I) = \mu_x(I \cap (-\infty, v^*])$. Similarly, define the “right weight” $\text{Rw}(I) = \mu_x(I \cap [v^*, \infty))$. The following claim is the basis of the “automatic binary search” idea and says that with good probability in each round we significantly reduce the left/right-weight of the uncertainty region.

Claim 1. *For each realization of the samples drawn before round i where the error region E_i is to the left of v^* , with probability at least $1 - e^{-2^{3i/4}}$ (with respect to the samples drawn at round i) we have*

$$\text{Lw}(I_{i+1}) \leq \frac{\text{Lw}(I_i)}{\alpha 2^{i/4}}.$$

Similarly, if the error region E_i is to the right of v^ , with probability at least $1 - e^{-2^{3i/4}}$ we have*

$$\text{Rw}(I_{i+1}) \leq \frac{\text{Rw}(I_i)}{\alpha 2^{i/4}}.$$

Proof. We only prove the first statement, the second being analogous. Fix a realization of the samples up to the beginning of round i where E_i is to the left of v^* . By construction, at round i the algorithm takes $\alpha 2^i$ samples from the distribution μ_x conditioned to being in the set E_i , call them $X_1, \dots, X_{\alpha 2^i}$. Let \bar{v} be the point such that the interval $[\bar{v}, v^*]$ has μ_x -measure $\frac{1}{\alpha 2^{i/4}} \cdot \mu_x(E_i)$. The probability that none of the samples X_i lands in this interval is $\left(1 - \frac{1}{\alpha 2^{i/4}}\right)^{\alpha 2^i} \leq e^{-2^{3i/4}}$, so with probability at least $1 - e^{-2^{3i/4}}$ one of these samples lands in

$[\bar{v}, v^*]$. When this happens, the next uncertainty I_{i+1} set starts at/after the point \bar{v} , and so its left weight satisfies

$$\text{Lw}(I_{i+1}) \leq \mu_x([\bar{v}, v^*]) = \frac{1}{\alpha 2^{i/4}} \mu_x(E_i) \leq \frac{1}{\alpha 2^{i/4}} \text{Lw}(I_i),$$

where the last inequality is because $E_i \subseteq I_i$ and thus (since E_i is to the left of v^*) $\mu_x(E_i) \leq \mu_x(I_i \cap (-\infty, v^*]) = \text{Lw}(I_i)$. This gives the desired result. \square

Let $R := cst \cdot (\log \frac{1}{\alpha} + \sqrt{\log \frac{1}{\varepsilon}} + 1) + \log \log \frac{1}{\delta}$, for a sufficiently large constant cst . Using the above claim, we show that with probability at least $1 - \delta$ the error set E_{R+1} at round $R+1$ has μ_x -measure at most ε , i.e. h_{R+1} has error at most ε . Let B_i be the indicator of the bad event that the weight reduction prescribed by the previous claim did not happened at round i . Then $\mathbb{E}[B_i \mid B_1, \dots, B_{i-1}] \leq e^{-2^{3i/4}}$. Moreover, using Freedman's Inequality we have the following tail bound.

Claim 2.

$$\Pr \left(\sum_{i=2R/3}^R B_i \geq \frac{R}{6} \right) \leq e^{-2^{R/2}} \leq \delta,$$

Proof. Define $\tilde{B}_i := B_i - \mathbb{E}[B_i \mid B_1, \dots, B_{i-1}]$, so that the sequence $\tilde{B}_{2R/3}, \dots, \tilde{B}_R$ is a martingale difference sequence. Moreover, since B_i only takes value 0 or 1, we have $|\tilde{B}_i| \leq 1$, and so

$$(\tilde{B}_i)^2 \leq |\tilde{B}_i| \leq B_i + \mathbb{E}[B_i \mid B_1, \dots, B_{i-1}].$$

From Claim 1, for $i \geq \frac{2R}{3}$ we have

$$\begin{aligned} \mathbb{E}[B_i \mid B_1, \dots, B_{i-1}] &\leq e^{-2^{3i/4}} \leq e^{-2^{R/2}} \\ \text{and so } \mathbb{E}[\tilde{B}_i^2 \mid \tilde{B}_{2R/3}, \dots, \tilde{B}_{i-1}] &\leq 2e^{-2^{R/2}}. \end{aligned}$$

Then $v := 2 \cdot \frac{2R}{3} \cdot e^{-2^{R/2}}$ is an upper bound for both the shifts introduced in \tilde{B}_i and the predictable quadratic variation with probability 1:

$$\begin{aligned} \sum_{i=2R/3}^R \mathbb{E}[B_i \mid B_1, \dots, B_{i-1}] &\leq v \\ \text{and } \sum_{i=2R/3}^R \mathbb{E}[\tilde{B}_i^2 \mid \tilde{B}_{2R/3}, \dots, \tilde{B}_{i-1}] &\leq v. \end{aligned}$$

Then Freedman's Inequality gives

$$\begin{aligned} \Pr \left(\sum_{i=2R/3}^R B_i \geq \frac{R}{6} \right) &\leq \Pr \left(\sum_{i=2R/3}^R \tilde{B}_i \geq \frac{R}{6} - v \right) \leq \left(\frac{v}{R/6} \right)^{R/6} e^{R/6} \leq (8e \cdot e^{-2^{R/2}})^{R/6} \\ &\leq e^{-2^{R/2}}, \end{aligned}$$

where the last inequality uses the fact that $R \geq 10$ (by setting cst large enough). Since $R \geq \log \log \frac{1}{\delta} + \sqrt{\log \frac{1}{\varepsilon}}$ and by assumption $\varepsilon \leq \delta$, we have $e^{-2^{R/2}} \leq e^{-2^{\log \log 1/\delta}} \leq \delta$. This proves the claim. \square

It then suffice to show that whenever $\sum_{i=2R/3}^R B_i < \frac{R}{6}$, we have $\mu_x(E_R) \leq \varepsilon$. So fix a scenario satisfying the former and assume by contradiction that $\mu_x(E_R) > \varepsilon$. Define the subsets G, L, R of the rounds $\{1, \dots, R\}$ as follows: G is the set of “good” indices i such that $B_i = 0$, L the set of indices where E_i is to the left of v^* , and R the set of indices where E_i is to the right of v^* .

Then $L \cap G$ are the rounds where the reduction prescribed by Claim 1 happened on the left weight of I_i and $R \cap G$ where it happened on its right weight. Moreover, the sets I_1, I_2, \dots are monotonically decreasing, so even for the rounds outside of the good set G the left and right weights of I_i only decrease over time. Then

$$\text{Lw}(I_{R+1}) \leq \text{Lw}(\mathbb{R}) \cdot \prod_{i \in L \cap G} \frac{1}{\alpha 2^{i/4}} \leq \left(\frac{1}{\alpha}\right)^R \cdot \frac{1}{2^{\sum_{i \in L \cap G} i/4}}.$$

Combining with the fact $\text{Lw}(I_{R+1}) \geq \text{Lw}(E_{R+1}) \geq \mu_x(E_{R+1}) > \varepsilon$ and taking logs, this implies

$$\sum_{i \in L \cap G} i \leq 4R \log \frac{1}{\alpha} + 4 \log \frac{1}{\varepsilon}.$$

The same inequality holds with L replaced by R , and adding these inequalities gives

$$\sum_{i \in (L \cup R) \cap G} i \leq 8R \log \frac{1}{\alpha} + 8 \log \frac{1}{\varepsilon}. \quad (4)$$

By the assumption of the scenario, we know that at least $\frac{R}{3} - \frac{R}{6} = \frac{R}{6}$ of the rounds in the interval $\{\frac{2R}{3}, \dots, R\}$ are in the good set G , that is, $|G| \geq \frac{R}{6}$. Then

$$\sum_{i \in (L \cup R) \cap G} i = \sum_{i \in G} i \geq \sum_{i=1}^{|G|} i \geq \sum_{i=1}^{R/6} i \geq \frac{R^2}{72}.$$

But since $R \geq cst \cdot (\log \frac{1}{\alpha} + \sqrt{\log \frac{1}{\varepsilon}})$ for a sufficiently large constant cst , this contradicts inequality (4) ($cst = 8 \cdot 72$ suffices, but the constants throughout have not been optimized).

So with probability at least $1 - \delta$, by round R the algorithm obtains a classifies with error at most ε . Due to the assumptions (ii) and (iv), the same development as in inequality (2) shows that this round finishes by time

$$\sum_{i=1}^R \left((2^i)^k \cdot f(2^i) \right) \leq 2f(2^R) 2^{Rk} \leq 2^{R(k+1)+1}.$$

Using the value of R , this time is at most

$$\left(2^{O(\sqrt{\log 1/\varepsilon})} \cdot \log \frac{1}{\delta} \cdot \left(\frac{1}{\alpha}\right)^{O(1)} \right)^{k+1},$$

which then concludes the proof. \square