Differentially Private Ensemble Classifiers for Data Streams

Lovedeep Gondara
School of Computing Science
Simon Fraser University
British Columbia, Canada
lgondara@sfu.ca

Ke Wang
School of Computing Science
Simon Fraser University
British Columbia, Canada
wangk@cs.sfu.ca

Ricardo Silva Carvalho
School of Computing Science
Simon Fraser University
British Columbia, Canada
rsilvaca@sfu.ca

ABSTRACT

Learning from continuous data streams via classification/regression is prevalent in many domains. Adapting to evolving data characteristics (concept drift) while protecting data owners' private information is an open challenge. We present a differentially private ensemble solution to this problem with two distinguishing features: it allows an *unbounded* number of ensemble updates to deal with the potentially never-ending data streams under a fixed privacy budget, and it is *model agnostic*, in that it treats any pre-trained differentially private classification/regression model as a black-box. Our method outperforms competitors on real-world and simulated datasets for varying settings of privacy, concept drift, and data distribution.

CCS CONCEPTS

• **Security and privacy** → **Privacy-preserving protocols**; • **Information systems** → Data stream mining.

KEYWORDS

Differential privacy; data streams; ensembles; concept drift

1 INTRODUCTION

Continuous data streams generate large volumes of data, with examples being data from wearables [10], biosensors in medicine [16], social media [31], news [29], mobile applications [2], electronic health records [8], credit card transactional flows [9], malware data [2]. To assist in decision making, machine learning models need to handle data streams efficiently. Scalability is not the only challenge; we have to consider that properties and patterns of data are subject to change over time, a phenomenon known as *concept drift*. For example, malware files and fake news evolve over time to evade detection [2, 29].

To further add to the challenge, data streams from many domains involve sensitive, personal information about contributing users, such as patients' records and user data in mobile applications, protection of which is of paramount interest. While concept drift and privacy have been extensively studied in isolation, works considering both are in infancy. See more discussion in Section 2. In this work, our goal is to allow machine learning models to deal with concept drift when training on potentially never-ending data streams involving sensitive data, where the model(s) learned can be published without disclosing sensitive information. To that end, we consider Differential Privacy (DP) [1, 13, 40] as the privacy definition, and widely used ensemble learning for data streams [7] as the modelling paradigm.

1.1 Challenges

Enforcing privacy on ensembles handling concept drift is not trivial. The main approaches of ensembles over data streams [7], such as weight modification [7, 28] and online ensemble update [7, 35] are not ideal for the privacy-preserving scenario where with new incoming instances, the former continuously measures the performance of a diverse set of classifiers to update the weights and the latter continuously updates the pool of online models. This *continuous* update could lead to privacy budget depletion due to composition of privacy loss, limiting the number of updates before the privacy budget runs out. Our goal is to deal with the never-ending data streams by allowing for an *unbounded* number of updates under a *fixed* privacy budget.

1.2 Our Proposals

We propose a DP temporal ensemble approach in the form of *dynamic ensemble line-up* [7, 41]. In the non-private setting, the data stream comes in the form of labeled data chunks $D_1, \cdots, D_t$ until the current time t, and at the time t the ensemble of size k, denoted $\mathcal{E}_t$, consists of k models $M_{\tau(1)}, \cdots, M_{\tau(k)}$ trained on the k corresponding data chunks $D_{\tau(1)}, \cdots, D_{\tau(k)}$. $\tau(i)$ maps the (relative) position i within $\mathcal{E}_t$ to the corresponding (absolute) time point in the data stream. With a weighting scheme, these k models collectively make the prediction for unlabeled data at the next time $t + 1$. As the labeled data chunk D_{t+1} becomes available, a new model is trained on D_{t+1} and the next ensemble $\mathcal{E}_{t+1}$ is obtained by replacing either the oldest model or the weakest model in $\mathcal{E}_t$ with the new model. By replacing (instead of updating) some existing model, this approach is particularly suited for limiting the accumulation of privacy loss. Our contributions are specifically described as follows.

(1) (Section 4) We formulate the problem of DP temporal ensembles for a release history with the formal definition provided in Section 6.

(2) (Section 5) At the core of our DP ensemble mechanism is the DP weighting scheme for aggregating the prediction of component models. We present the DP weighting scheme for classification and regression. Our method is *model agnostic*, that is, it treats DP models M_i as black-boxes.

(3) (Section 6) We present a DP ensemble mechanism for releasing the ensemble $\mathcal{E}_t = \{M_{\tau(1)}, \cdots, M_{\tau(k)}\}$ at any time t, to ensure that the DP guarantee holds even if the adversary has access to all released ensembles $\mathcal{E}_{t'}$ for $t' \leq t$. Our proposal allows an *unlimited* number of ensemble updates for never ending data streams at a constant privacy budget.

(4) (Section 7) To demonstrate the benefits of our method for deep neural network models where the potentially large number of model parameters present a challenge for retaining utility under DP guarantee, we consider a transfer learning option for boosting utility where a public dataset is available.

(5) (Section 7) We provide empirical evidence on the effectiveness of the proposed DP temporal ensemble using real-world and simulated datasets. The source code and datasets will be made publicly available for reproducibility[1].

2 RELATED WORK

Non-private temporal ensembles have been studied before [7, 28, 39, 41]. See [7, 20, 30] for a review. These methods can be classified into either explicit or implicit. *Explicit* methods use drift detection and only update the model when drift is detected. Examples are [11] and [44]. *Implicit* methods do not detect drift but adapt the model to account for changes automatically. The updates can be incremental in a single classifier [17], or weighted in an ensemble [41]. Ensemble models in the implicit setting usually outperform other approaches [30, 41]. Our work is an implicit method and adapts dynamic ensemble line-up [7] but deals with sensitive data.

For privacy preserving works on data streams, [45] proposes DP Bayesian classifiers with explicit drift detection. This method continuously updates the model parameters using incoming sensitive data which reduces the privacy budget by some amount after each update, thus, the privacy budget will run out after a finite number of updates. Also, the method does not account for the privacy loss for drift detection and the privacy loss of updates when no concept drift is detected. The works in [14, 18, 19, 26, 27] focused on releasing summary statistics such as counts, mean, mode, range queries, centroids, etc.

There are previous works on *static* differentially private ensembles such as [23, 42]. These works are not designed to handle streaming data because they cannot accommodate concept drift.

3 DIFFERENTIAL PRIVACY

DEFINITION 1 (NEIGHBORS AND SENSITIVITY). *Two data sets D and D' are neighboring if they differ due to the substitution of exactly one sample. The sensitivity of a function $f : \mathbb{D}^n \to \mathbb{R}^d$, denoted by Δf, is $max_{D,D'} ||f(D) - f(D')||$ over all neighboring pairs D and D'.*

DEFINITION 2 (DIFFERENTIAL PRIVACY [12]). *A randomized mechanism $\mathcal{M} : \mathbb{D}^n \to \mathbb{R}^d$ is (ε, δ)-differentially private if for any pair*

[1]https://github.com/lgondara/DPTemporalEnsemble

of neighbouring data sets $D, D' \in \mathbb{D}^n$, and for all sets S of possible outputs:

$$Pr[\mathcal{M}(D) \in S] \leq e^{\varepsilon} Pr[\mathcal{M}(D') \in S] + \delta \qquad (1)$$

Since neighboring datasets differ by the data of one user, the inequality above ensures that the output of a mechanism $\mathcal{M}$ satisfying DP will have a *small* impact, through the multiplicative factor e^{ε} and the additive factor δ, if we remove or add any single user from the dataset used to generate the output.

THEOREM 1 (PARALLEL COMPOSITION [32]). *Let $\mathcal{M}_i$ each provide $(\varepsilon_i, \delta_i)$-differential privacy. Let D_i be arbitrary disjoint subsets of D. The sequence of $\mathcal{M}_i(D_i)$ provides $(max_i\varepsilon_i, max_i\delta_i)$-differential privacy.*

THEOREM 2 (SEQUENTIAL COMPOSITION [32][15]). *Let $\mathcal{M}_i$ be $(\varepsilon_i, \delta_i)$-differentially private. The adaptive sequence of $\mathcal{M}_i$ is $(\sum_i \varepsilon_i, \sum_i \delta_i)$-differentially private.*

THEOREM 3 (POST PROCESSING [15]). *Let $\mathcal{M} : \mathbb{D}^n \to \mathbb{R}^d$ be a randomized mechanism that is (ε, δ)-differentially private. Let $f : \mathbb{R}^d \to \mathbb{R}^t$ be a deterministic function. Then $f \circ \mathcal{M} : D^n \to \mathbb{R}^t$ is (ε, δ)-differentially private.*

DEFINITION 3 (LAPLACE MECHANISM [13]). *Given any function $f : \mathbb{D}^n \to \mathbb{R}^d$, the Laplace mechanism is defined as: $\mathcal{M}(D, f(), \varepsilon) = f(D) + (Y_1, \cdots, Y_d)$, where Y_i are i.i.d. random variables drawn from $Lap(\Delta f/\varepsilon)$.*

THEOREM 4 (DP OF LAPLACE MECHANISM [13]). *The Laplace mechanism is $(\varepsilon, 0)$-differentially private.*

4 PROBLEM STATEMENT

We consider a data stream $D_1, \cdots, D_t$ where each D_i is a chunk of data generated at the time i and t is the *current time*. Samples in D_i are labeled with a class variable and we are interested in using the data $D_1, \cdots, D_t$ to predict the class for unlabeled samples at the next time $t + 1$. A new data chunk D_{t+1} that becomes available at the next time $t+1$ is added to the stream, so the stream is potentially unbounded. We consider the dynamic setting where the data stream is susceptible to concept drift.

An ensemble of size k at time t, $\mathcal{E}_t = \{M_{\tau(1)}, \cdots, M_{\tau(k)}, w_{\tau(1)}, \cdots, w_{\tau(k)}\}$, consists of k prediction models M_is respectively trained on $D_{\tau(1)}, \cdots, D_{\tau(k)}$, and their weights w_i, where $\tau(i)$ denotes the (absolute) time point corresponding to the (relative) position i within $\mathcal{E}_t$. We assume $\tau(1), \cdots, \tau(k)$ are listed in the ascending order.

For *classification*, we have a categorical class variable c, $M_{i,c}(x) \in [0, 1]$ denotes the prediction score by M_i for c on a sample x, and the overall score predicted by $\mathcal{E}_t$ is computed by

$$\mathcal{E}_{t,c}(x) = \frac{\sum_{i=1}^{k} w_{\tau(i)} \cdot M_{\tau(i),c}(x)}{\sum_{i=1}^{k} w_{\tau(i)}} \qquad (2)$$

The predicted class for x is the class c with the maximum $\mathcal{E}_{t,c}(x)$.

For *regression*, we have a continuous class variable, normalized to within the range $[0, 1]$, $M_i(x) \in [0, 1]$ denotes the predicted value by M_i for x and the overall predicted value by $\mathcal{E}_t$ is given by

$$\mathcal{E}_t(x) = \frac{\sum_{i=1}^{k} w_{\tau(i)} \cdot M_{\tau(i)}(x)}{\sum_{i=1}^{k} w_{\tau(i)}} \qquad (3)$$

As D_{t+1} becomes available at time $t + 1$, we use it to train a new model and re-estimate the weights for all models in the ensemble (more details later). Then we update $\mathcal{E}_t$ to $\mathcal{E}_{t+1}$ by replacing either the *oldest* model or the *worst* (i.e., with smallest weight) model in $\mathcal{E}_t$.

Problem of (ε, δ)-DP Temporal Ensembles. The focus of this work is the scenario where each D_i contains sensitive, private information about the contributing users. We assume that each sample in $\cup_i D_i$ has a unique identifier and all samples are independently generated. This independence assumption would allow us to treat all D_i as disjoint subsets of $\cup D_i$. We want to ensure that, at any current time t, the entire history of released ensembles up to t, i.e., $\mathcal{E}'_t$ for all $t' < t$, must satisfy (ε, δ)-DP for given ε, δ. A formal definition of (ε, δ)-DP for an ensemble and for a history will be given in Section 6.1 and in Section 6.2.

We develop noisy weight mechanisms in Section 5 and provide the privacy analysis for the overall temporal ensemble mechanism in Section 6. The given ε is split into ε_1 for training models M_i and ε_2 for computing noisy weights w_i^*, where $\varepsilon = \varepsilon_1 + \varepsilon_2$.

5 NOISY WEIGHT ESTIMATION

We assume that the labeled data D_i is split into training, validation, and testing subsets. For an ensemble $\mathcal{E}_t$ at time t, the weights $w_{\tau(1)}, \cdots, w_{\tau(k)}$ are measured using the performance on the validation subset of D_t, denoted by V_t. We choose the validation subset of D_t to calculate the weights for all models in $\mathcal{E}_t$ because t is closest to the next time point $t + 1$ that the ensemble at time t aims to predict. In this section, we present the noisy estimation of $w_{\tau(1)}, \cdots, w_{\tau(k)}$ and we present the privacy analysis in Section 6.

For classification, we consider two settings. In the **general setting**, we measure the classification accuracy for all classes. In the **focused setting**, we consider the accuracy of a chosen class called the positive class, which is commonly used in class-imbalanced problems such as fake news/malware/disease detection.

5.1 Classification - General Setting

Consider a model M_i in $\mathcal{E}_t$ and the validation subset V_t. In the general setting, we consider the classification error of M_i defined by the **Mean Squared Error** (MSE), as in [6, 41]:

$$MSE_i = \frac{1}{|V_t|} Err_i \tag{4}$$

where

$$Err_i = \sum_{x \in V_t} (1 - M_{i,c}(x))^2 \tag{5}$$

and $M_{i,c}(x)$ is the score given by M_i for the instance x and its true class c. For a random predictor, the MSE is given by

$$MSE_r = \sum_c p(c)(1 - p(c))^2 \tag{6}$$

where $p(c)$ is the proportion of class c in V_t. The weight w_i for M_i is defined as the hinge loss:

$$w_i = \max(0, MSE_r - MSE_i) \tag{7}$$

LEMMA 1 (GENERAL SETTING). *Let ΔErr_i denote the sensitivity of Err_i defined over all neighboring validation subsets V_t, V'_t. $\Delta Err_i = 1$.*

PROOF. Consider computing Equation (5) for neighboring validation sets V_t, V'_t. All $M_{i,c}(x)$ are same except for one instance x, so Err_i differs by at most 1 because $M_{i,c}(x)$ is at most 1. □

Computing Noisy Weight w_i^*: w_i^* is computed from Equation (7) and (4) using the noisy Err_i^*:

$$Err_i^* = Err_i + Lap(\Delta Err_i / \varepsilon_2) \tag{8}$$

5.2 Classification - Focused Setting

The focused setting is concerned with prediction performance of the positive class. Typically the positive class has a small proportion compared to other classes called the negative class and the general classification accuracy does not reflect the accuracy of the positive class. In this case, we consider the **balanced accuracy** (BA):

$$BA = a_1 \cdot TPR + a_2 \cdot TNR \tag{9}$$

where a_1 and a_2 are constants and $a_1 + a_2 = 1$. TPR and TNR are the true positive rate (the proportion of predicted positives that are actually positive) and the true negative rate (the proportion of predicted negatives that are actually negative). BA is in the range $[0, 1]$. Since $TNR = 1 - FPR$, where FPR is the false positive rate (the proportion of negatives that are predicted as positives), BA is related to (TPR,FPR) commonly used for generating AUC. The above BA generalizes the balanced accuracy in [5] that assumes $a_1 = a_2 = 1/2$.

To obtain the noisy weight, we assume that some estimates of the proportions of positive samples and negative samples in D_i, denoted by p and n with $p + n = 1$, are public. These are *not* necessarily the exact proportions in the sensitive data, but rather are estimates from general knowledge (for example, p and n come from the general knowledge about the entire data stream). These estimates allow us to estimate BA_i for M_i as follows:

$$BA_i = a_1 \frac{TP_i}{p \cdot |V_t|} + a_2 \frac{TN_i}{n \cdot |V_t|} \tag{10}$$

where TP_i (resp. TN_i) is the number of positive instances (resp. negative instances) in V_t that are predicted by M_i as positive (negative).

LEMMA 2 (FOCUSED SETTING). *Let ΔBA_i denote the sensitivity of BA_i defined over neighboring pairs V_t, V'_t. $\Delta BA_i = \frac{1}{|V_t|} max(\frac{a_1}{p}, \frac{a_2}{1-p})$.*

PROOF. Consider neighboring validation subsets V_t, V'_t. For simplicity, we drop the index i below.

$$BA - BA' = \frac{a_1}{p} \times \left(\frac{TP}{|V_t|} - \frac{TP'}{|V_t|} \right) + \frac{a_2}{n} \times \left(\frac{TN}{|V_t|} - \frac{TN'}{|V_t|} \right)$$

$$= \frac{1}{|V_t|} \left(\frac{a_1}{p} \times (TP - TP') + \frac{a_2}{n} \times (TN - TN') \right)$$

For neighboring V_t, V'_t where only one sample is different, there are four possible cases: (i) both $TP - TP'$ and $TN - TN'$ are 0, (ii) one of $|TP - TP'|$ and $|TN - TN'|$ is 1 and the other is 0, (iii) $TP - TP' = 1$ and $TN - TN' = -1$, (iv) $TP - TP' = -1$ and $TN - TN' = 1$. Noting $p + n = 1$, we have:

$$|BA - BA'| \le \frac{1}{|V_t|} \times max(\frac{a_1}{p}, \frac{a_2}{1-p})$$

□

In the above lemma we assume that the validation size $|V_t|$ is public. The same assumption was made in [1, 4, 25] (e.g. see Remark 2.2 in [25]). Alternatively, if a minimum validation size $|V_t|$ for all V_ts is required (for the purpose of statistical significance), we can use the minimum size in ΔBA_i without referring to specific $|V_t|$.

Computing Noisy Weight w_i^*: We add the Laplace noise:

$$w_i^* = BA_i + Lap(\Delta BA/\varepsilon_2) \tag{11}$$

5.3 Regression

For a continuous class variable, we define Err_i as

$$Err_i = \sum_{x \in V_t} (y_x - \hat{y}_x)^2 \tag{12}$$

where y_x is the true class value of x and $\hat{y}_x$ is the predicted class value by the regression model M_i. We then get MSE_i as

$$MSE_i = \frac{1}{|V_t|} Err_i \tag{13}$$

and define the weight for M_i as

$$w_i = \frac{1}{MSE_i + \mu} \tag{14}$$

μ is a small constant to allow weight calculation in rare situations when $MSE_i = 0$. $\mu = 10^{-5}$ is used in our experiments.

LEMMA 3 (REGRESSION). *Let ΔErr_i denote the sensitivity of Err_i defined over all neighboring validation subsets V_t, V_t'. $\Delta Err_i = 1$.*

PROOF. Recall that the true class value and the predicted class value are in the range $[0, 1]$. So, for neighboring validation sets V_t, V_t', Err_i differs by at most 1 because $(y_x - \hat{y}_x)^2$ is at most 1. □

Computing Noisy Weight w_i^*: We add the Laplace noise

$$Err_i^* = Err_i + Lap(\Delta Err_i/\varepsilon_2) \tag{15}$$

and compute w_i^* using Err_i^*, Eqn. (13) and Eqn. (14).

Discussion. The weighting scheme for both classification and regression is model agnostic, that is, it treats the DP models M_i as black-boxes. This is because the computation of the weights w_is only depends on the outputs, not the internal working of M_i.

6 DP TEMPORAL ENSEMBLE

In Section 6.1, we provide the privacy analysis for a single (ε, δ)-DP ensemble $\mathcal{E}_t = \{M_{\tau(1)}, \cdots, M_{\tau(k)}, w_{\tau(1)}^*, \cdots, w_{\tau(k)}^*\}$, where each model M_i is trained using any method on the training subset S_i of D_i and its weight w_i^* is computed using the validation subset V_t of D_t. In Section 6.2, we update $\mathcal{E}_t$ to $\mathcal{E}_{t+1}$ and present the privacy analysis for releasing all ensembles $\mathcal{E}_1, \cdots, \mathcal{E}_t$ up to the time t.

6.1 Releasing A Single Ensemble

First, we extend the notion of DP in Definition 1 to releasing an ensemble $\mathcal{E}_t$. Let $X_t = < S_{\tau(1)}, \cdots, S_{\tau(k)}, V_t >$, where $S_{\tau(1)}, \cdots, S_{\tau(k)}$ are the training subsets for $M_{\tau(1)}, \cdots, M_{\tau(k)}$ and V_t is the validation subset of D_t for computing the noisy weights $w_{\tau(1)}^*, \cdots, w_{\tau(k)}^*$.

DEFINITION 4 (NEIGHBORING DATASETS FOR ENSEMBLES). *Consider $X_t = < S_{\tau(1)}, \cdots, S_{\tau(k)}, V_t >$ and $X_t' = < S_{\tau(1)}', \cdots, S_{\tau(k)}', V_t' >$.*

We say that X_t and X_t' are neighboring if $\cup_i S_i \cup V_t$ and $\cup_i S_i' \cup V_t'$ (duplicates preserved) are neighboring in the sense of Definition 1.

Note that X_t and X_t' are neighboring if and only if either for one i, S_i and S_i' are neighboring and $S_j = S_j'$ for all $j \neq i$, or V_t and V_t' are neighboring and $S_i = S_i'$ for all i.

DEFINITION 5 (DIFFERENTIAL PRIVACY FOR ENSEMBLES). *A mechanism C from the domain of X_t to the range of $\mathcal{E}_t$ is (ε, δ)-differentially private if for all neighbouring pairs (X_t, X_t') and for all sets O of possible outputs:*

$$\Pr[C(X_t) \in O] \leq e^\varepsilon \Pr[C(X_t') \in O] + \delta \tag{16}$$

THEOREM 5. *Assume that each M_i in $\mathcal{E}_t$ is produced by a model-agnostic (ε_1, δ)-DP mechanism $\mathcal{A}$ and that the noisy weight w_i^* is produced by the Laplace mechanism $\mathcal{L}$ in Section 5. Then the combined mechanism that produces $\mathcal{E}_t = \{M_{\tau(1)}, \cdots, M_{\tau(k)}, w_{\tau(1)}^*, \cdots, w_{\tau(k)}^*\}$ is $(max\{\varepsilon_1, k \cdot \varepsilon_2\}, \delta)$-differentially private.*

PROOF. For simplicity of proof, we write $\tau(1), \cdots, \tau(k)$ as $1, \cdots, k$. The (ε_1, δ)-DP guarantee of $\mathcal{A}$ implies that for neighboring training subsets S_i, S_i' (Def. 4), and for any possible set $\mathcal{M}_i$ of outputs:

$$\Pr[\mathcal{A}(S_i) \in \mathcal{M}_i] \leq e^{\varepsilon_1} \Pr[\mathcal{A}(S_i') \in \mathcal{M}_i] + \delta \tag{17}$$

For the weight calculation, the Laplace mechanism $\mathcal{L}$ provides $(\varepsilon_2, 0)$-DP for releasing the noisy weights w_i^* following Theorem 4. therefore, for any $M_i \in \mathcal{M}_i$, neighboring validation subsets V_t and V_t', and any possible set $\mathcal{W}_i^*$ of weights:

$$\Pr[\mathcal{L}(M_i(V_t)) \in \mathcal{W}_i^*] \leq e^{\varepsilon_2} \Pr[\mathcal{L}(M_i(V_t')) \in \mathcal{W}_i^*] \tag{18}$$

Denoting our combined mechanism as C, with input $X_t = < S_1, \cdots, S_k, V_t >$, and any possible set of outputs $O = \{\mathcal{M}_1, \cdots, \mathcal{M}_k, \mathcal{W}_1^*, \cdots, \mathcal{W}_k^*\}$, we get:

$$\Pr[C(X_t) \in O] =$$
$$\Pr[\mathcal{A}(S_1) \in \mathcal{M}_1] \cdot \ldots \cdot \Pr[\mathcal{A}(S_k) \in \mathcal{M}_k] \cdot$$
$$\Pr[\mathcal{L}(M_1(V_t)) \in \mathcal{W}_1^*] \cdot \ldots \cdot \Pr[\mathcal{L}(M_k(V_t)) \in \mathcal{W}_k^*] \tag{19}$$

Now consider the only two possible cases of neighboring $X_t = < S_1, \cdots, S_k, V_t >$ and $X_t' = < S_1', \cdots, S_k', V_t' >$: (I) change one arbitrary S_i or (II) change V_t.

For Case (I), all models should satisfy Equation (17), but since only **one** S_i changes to reach a neighboring input, Equation (17) will be obtained on one M_i and for all the others $j \neq i$ we would get $\Pr[\mathcal{A}(S_j) \in \mathcal{M}_j] = \Pr[\mathcal{A}(S_j') \in \mathcal{M}_j]$ as $S_j = S_j'$. Additionally, since in this case we are **not** changing V_t, $V_t = V_t'$, so for all $1 \leq i \leq k$, $\Pr[\mathcal{L}(M_i(V_t)) \in \mathcal{W}_i^*] = \Pr[\mathcal{L}(M_i(V_t')) \in \mathcal{W}_i^*]$. Combining these two facts we reach $\Pr[C(X_t) \in O] \leq e^{\varepsilon_1} \Pr[C(X_t') \in O] + \delta$ from Equation (19) above.

For Case (II), we do not change any of $S_1, \cdots, S_k$, thus for all $1 \leq i \leq k$, $\Pr[\mathcal{A}(S_i) \in \mathcal{M}_i] = \Pr[\mathcal{A}(S_i') \in \mathcal{M}_i]$. Additionally, changing V_t for this scenario, every application of the Laplace mechanism satisfies Equation (18), which is done k times for M_i, $1 \leq i \leq k$. Combining these two facts we reach $\Pr[C(X_t) \in O] \leq e^{k\varepsilon_2} \Pr[C(X_t') \in O]$ from Equation (19) above.

Finally, since DP must hold for the worst-case guarantee, we take the maximum between the two cases defined above, which gives us the $(max\{\varepsilon_1, k \cdot \varepsilon_2\}, \delta)$-DP. Note that combining the two cases is a tailored instantiation of the parallel composition (Theorem 1). □

Discussion. The construction of (ε_1, δ)-DP mechanism $\mathcal{A}$ for training a single model M_i has been studied in the literature, for example, DP neural networks [1], DP random forest [36], and DP support-vector machine [37]. Our focus is on the construction of (ε, δ)-DP mechanisms $\mathcal{L}$ for building an ensemble $\mathcal{E}_t = \{M_{\tau(1)}, \cdots, M_{\tau(k)}, w^*_{\tau(1)}, \cdots, w^*_{\tau(k)}\}$, using the single model mechanism $\mathcal{A}$ as a black-box.

6.2 Releasing the History of Ensembles

Algorithm 1 Update The Ensemble

Input: The ensemble $\mathcal{E}_t = \{M_{\tau(1)}, \cdots, M_{\tau(k)}, w^*_{\tau(1)}, \cdots, w^*_{\tau(k)}\}$;
  the training and validation subsets at $t+1$, i.e., S_{t+1} and V_{t+1};
  privacy parameters $(\varepsilon_1, \varepsilon_2, \delta)$; update_mode (oldest or worst).
Output: $\mathcal{E}_{t+1}$
1: Train (ε_1, δ)-DP model M_{t+1} on S_{t+1}
2: **if** update_mode = "oldest" **then**
3:   Calculate $w^*_{\tau(2)}, \cdots, w^*_{\tau(k)}, w^*_{t+1}$ for $M_{\tau(2)}, \cdots, M_{\tau(k)}, M_{t+1}$
      using V_{t+1} and ε_2
4:   $w^*_{\tau(1)}, \cdots, w^*_{\tau(k-1)}, w^*_{\tau(k)} \leftarrow w^*_{\tau(2)}, \cdots, w^*_{\tau(k)}, w^*_{t+1}$
5:   $M_{\tau(1)}, \cdots, M_{\tau(k-1)}, M_{\tau(k)} \leftarrow M_{\tau(2)}, \cdots, M_{\tau(k)}, M_{t+1}$
6:   $\mathcal{E}_{t+1} = \{M_{\tau(1)}, \cdots, M_{\tau(k)}, w^*_{\tau(1)}, \cdots, w^*_{\tau(k)}\}$
7: **else**
8:   Calculate $w^*_{\tau(1)}, \cdots, w^*_{\tau(k)}, w^*_{t+1}$ for $M_{\tau(1)}, \cdots, M_{\tau(k)}, M_{t+1}$
      using V_{t+1} and ε_2
9:   $w^*_{\tau(1)}, \cdots, w^*_{\tau(k)}, w^*_{\tau(k+1)} \leftarrow w^*_{\tau(1)}, \cdots, w^*_{\tau(k)}, w^*_{t+1}$
10:   $i^* \leftarrow argmin_i\{w^*_{\tau(i)} \mid 1 \le i \le k+1\}$
11:   $\mathcal{E}_{t+1} \leftarrow \{M_{\tau(i)}, w^*_{\tau(i)} \mid 1 \le i \le k+1, i \ne i^*\}$
12: **end if**
13: return $\mathcal{E}_{t+1}$

Algorithm 1 shows the steps for updating the ensemble $\mathcal{E}_t$ to adapt the new chunk D_{t+1} for two update modes, indicated by the input variable update_mode: replace the oldest model and replace the worst model (i.e., the model having smallest w^*_i). In the former case V_{t+1} is used k times (Step 3), and in the latter case V_{t+1} is used $k+1$ times (Step 8). Theorem 5 shows that releasing a single ensemble $\mathcal{E}_t$ satisfies $(\max\{\varepsilon_1, k \cdot \varepsilon_2\}, \delta)$-DP. With the repeated update at each time t, the adversary is able to access the history of all released ensembles up to the current time. We show that, with the access to the history, the $(\max\{\varepsilon_1, k \cdot \varepsilon_2\}, \delta)$-DP remains to hold in the case of replacing oldest model, and degrades to $(\max\{\varepsilon_1, (k+1) \cdot \varepsilon_2\}, \delta)$-DP in the case of replacing worst model.

First, we extend the notion of DP to the global input data X from time 1 to time t, i.e., $X = <X_1, \cdots, X_t>$ where X_i is the input data for the ensemble $\mathcal{E}_i$ defined in Definition 4. We say that X and X' are *neighboring* if exactly one pair (X_i, X'_i) is neighboring, as defined in Definition 4, and for all other $j \ne i, X_j = X'_j$. We consider the output consisting of all ensembles released up to the time t, i.e., $\mathcal{E} = <\mathcal{E}_1, \cdots, \mathcal{E}_t>$.

DEFINITION 6 (DIFFERENTIAL PRIVACY FOR HISTORY). *A mechanism C from the domain of X to the range of $\mathcal{E}$ is (ε, δ)-differentially private with respect to history if for any neighbouring pair (X, X')*

and for all sets O of possible outputs:

$$\Pr[C(X) \in O] \le e^\varepsilon \Pr[C(X') \in O] + \delta \qquad (20)$$

THEOREM 6. *With update_mode="oldest", Algorithm 1 is $(\max\{\varepsilon_1, k \cdot \varepsilon_2\}, \delta)$-DP with respect to history.*

PROOF. The proof is basically the same as for Theorem 5, noting that X and X' differ only in a single sample either in the training subset or in the validation subset, for one ensemble. □

Therefore, even if the adversary has access to all released ensembles, the privacy loss does not accumulate compared to releasing a single ensemble. This is due to the two facts. (i) each model in an ensemble is trained on a *disjoint* training subset, which ensures that accessing more models does not change the (ε_1, δ)-DP (i.e., parallel composition, Theorem 1), (ii) each validation subset is used exactly k *times* (that is, V_t is used for the k models in $\mathcal{E}_t$), which ensures the $(k \cdot \varepsilon_2, 0)$-DP remains unchanged.

THEOREM 7. *With update_mode="worst", Algorithm 1 is $(\max\{\varepsilon_1, (k+1) \cdot \varepsilon_2\}, \delta)$-DP with respect to history.*

PROOF. The proof follows the same idea as Theorem 6, but for the case of replacing the worst model, we have to calculate the weights for all k models already in the ensemble *plus* the additional new model in order to find the worst model, so each validation subset is used $k+1$ times. Therefore, now we have the overall privacy guarantee of $(\max\{\varepsilon_1, (k+1) \cdot \varepsilon_2\}, \delta)$-DP. □

Discussion. Therefore, replacing worst model incurs a *slightly* larger privacy loss, compared to replacing oldest model. Importantly, in both cases the privacy loss depends on the size of an ensemble, k, but not on the number of ensembles released. This property is essential for practical use because the number of ensemble updates is potentially unbounded for data streams. To optimize the given privacy budget (ε, δ), we can set $\varepsilon_1 = \varepsilon$ and $\varepsilon_2 = \varepsilon/k$ when replacing oldest model (Theorem 6), and set $\varepsilon_1 = \varepsilon$ and $\varepsilon_2 = \varepsilon/k+1$ when replacing worst model (Theorem 7).

Dataset	Attr.	Obs.	C/R	Prop.	Type
Hyperplane	20	Variable	C	50%	Synthetic
EMBER-B	2381	2,100,000	C	50%	Real
EMBER-U	2381	1,365,000	C	30%	Real
Housing Market	292	30,473	R	NA	Real

Table 1: C/R for classification/regression and Prop. for the proportion of positive class.

7 EVALUATION

This section evaluates the proposed DP temporal ensemble method. We train each M_i as a neural network using DPSGD [1] with privacy budget (ε_1, δ). In each iteration, DPSGD adds the Gaussian noise $\mathcal{N}(0, \sigma^2 C^2)$ to the clipped gradient $\frac{g(x_i)}{\max(1, \|g(x_i)\|_2/C)}$ where C is the clipping factor. For a large number of model parameters, $\|g(x_i)\|_2$ is large, leading to a noisy gradient. This effect is compounded for typically small data chunk sizes where the sampling ratio for a fixed minibatch size becomes relatively large, which increases σ. To reduce the norm $\|g(x_i)\|_2$, we also consider the option of *transfer learning* for training M_i: first, we pre-train a model using a

public dataset P without privacy concerns (for example, obsolete dataset, anonymized dataset, dataset obtained with data owners' explicit consent, or dataset from related but public domain) and then, we train only the last few layers using the sensitive D_i via DPSGD keeping the parameters for other layers unchanged. If no such public P is available, M_i will be fully trained using the sensitive D_i via DPSGD.

7.1 Data and Model Details

Table 1 shows the data summary.

7.1.1 Hyperplane. Hyperplane is a *synthetic* dataset used extensively in the concept drift literature [22, 41] to classify points separated by a hyperplane. We simulate time-evolving concepts by changing the orientation and the position of the hyperplane in a smooth manner. As in [21], we use the `HyperplaneGenerator()` function from [33] to create the simulated points, and use four parameters (`n_drift_features`, `mag_change` , `noise_percentage`, and `sigma_percentage`) to generate four drift types: *gradual* drift (concept changes slowly over time)[2], *rapid* drift (change happens at a rapid pace)[3], *recurrent* drift (concepts reappear at future times, every fifth time for our case)[4], and *abrupt* drift (concept changes suddenly at a time instance, every fifth time for our case)[5]. We evaluate using the classification Accuracy for the general setting. For all drift types, we generate a total of 20 chunks D_is with the default size of 1000, and use a fully connected neural network with two hidden layers of sizes 20 and 10 respectively with ReLU as the activation function for the hidden layers and softmax for the output layer. The default drift type is rapid. We do not use any public data or transfer learning for this dataset.

7.1.2 EMBER-(B & U). EMBER [3] contains features for Windows executable files for the years of 2017 and 2018 with the goal to classify malicious vs benign files, and the dataset has a natural concept drift [43]. We remove unlabelled observations. **EMBER-B** is the original class-balanced version and **EMBER-U** is obtained by under-sampling the positive class to 30%. For EMBER-B, we evaluate using classification Accuracy, and for EMBER-U we evaluate using Balanced Accuracy (BA) with $a_1 = 0.7$ and $a_2 = 0.3$ (Eqn. (9)). The data chunks D_is are created as bi-weekly observations, leading to an average chunk size of 33,333 for EMBER-B and 21,666 for EMBER-U. A fully connected neural network is selected via hyperparameter search [6]. For transfer learning, we use the first six months of 2017 as the public data P, leave the last six months of 2017 as the *time buffer*, and retrain the last two layers of the pre-trained model (preserving the layer sizes) using D_is for 2018.

7.1.3 Housing Market. Housing market [38] contains the property information from August 2011 to June 2015, with the goal of predicting the continuous property price (i.e., regression). We evaluate using 1-MSE where MSE is defined by Eqn. (13). We normalize the property price to within [0,1]. A fully connected neural network is selected via hyperparameter search [7]. For transfer learning, we use

[2]Parameter values: <10, 0.1, 0.05, 0.1>
[3]Parameter values: <20, 0.4, 0.1, 0.4>
[4]Same parameters as in rapid drift, use restart() argument every fifth time
[5]Parameter values: <0, 0, 0, 0>, switch labels every fifth time
[6]Four hidden layers (1400,2000,1100,250,2), ReLU for hidden, softmax for output
[7]Five hidden layers (500,350,250,150,50,1), ReLU for hidden, sigmoid for output

the data from 2011 as public data P to pre-train a model, leave out the data from 2012 as the time buffer, and use the months starting from January 2013 as our monthly data chunks D_is, leading to an average chunk size of 859. M_i is obtained by retraining the last two layers of the pre-trained model (preserving the sizes) using D_i.

For *all* datasets: we standardize continuous features using StandardScaler from scikit-learn[34] and use the one-hot encoding for categorical features. We run DPSGD with 30 epochs with the mini-batch size of 100. The labeled data D_i is split into train-validation-test using 70%-20%-10% and we use the training subset for training the model, validation subset for weight estimation, and the test subset to report the performance. We report the average result of 10 runs with standard errors.

Method	Private	Ensemble	Transfer	Data
EPT	✓	✓	✓	$[D_{\tau(1)}, \cdots, D_{\tau(k)}]$
EP	✓	✓	✗	$[D_{\tau(1)}, \cdots, D_{\tau(k)}]$
PT(1)	✓	✗	✓	$[D_{\tau(1)}]$
PT(k)	✓	✗	✓	$[D_{\tau(1)} \cup \cdots \cup D_{\tau(k)}]$
ET	✗	✓	✓	$[D_{\tau(1)}, \cdots, D_{\tau(k)}]$
E	✗	✓	✗	$[D_{\tau(1)}, \cdots, D_{\tau(k)}]$

Table 2: Competitor methods. ✓ signifies the presence of a trait whereas ✗ signifies its absence. Data is the data for training an ensemble or training a model for non-ensembles.

7.2 Competitor Methods

As discussed in Section 2, existing works on data streams either deal with summary statistics or do not consider privacy, or cannot deal with an unbounded number of updates. Table 2 lists the methods evaluated and their characteristics. We use the following naming convention: **"E"** denotes ensemble classifiers, **"P"** denotes DP, and **"T"** denotes transfer learning.

EPT is the DP temporal ensemble proposed in Section 6 consisting of k models (M_is) trained on the k data chunks $D_{\tau(1)}, \cdots, D_{\tau(k)}$s with transfer learning, whereas **EP** does not use transfer learning. **ET** and **E** are the non-private versions of EPT and EP and they serve as an upper bound for the performance of EPT and EP. We also compare our methods with two *non-ensemble* solutions, **PT(1)** and **PT(k)**, where a *single* model is used for prediction. PT(k) uses the union $D_{\tau(1)} \cup \cdots \cup D_{\tau(k)}$ of k chunks to train the model and advances to the *next non-overlapping* window covering times $t + 1, \cdots, t + k$, and PT(1) is the special case of $k = 1$, i.e., building a new model using each new chunk. With a single model trained using non-overlapping chunks, these methods do not need weight estimation and will spend the whole privacy budget on training the model. For prediction, EPT, EP, ET, E, and PT(1) are used to predict in the *next time* (i.e., $t + 1$) whereas PT(k) predicts in its next window (i.e., $t + 1, \cdots, t + k$).

All DP methods are evaluated under the same privacy budget (ε, δ). The following default settings are used: ($\varepsilon = 1$, $\delta = 0.0001$), window size $k = 5$, drift type = "rapid", and update_mode = "oldest". We set $\varepsilon_1 = \varepsilon$ and $\varepsilon_2 = \varepsilon/k$ for update_mode = "oldest", and set $\varepsilon_1 = \varepsilon$ and $\varepsilon_2 = \varepsilon/k+1$ for update_mode = "worst". We begin training for *all* methods once we have the *first* k data chunks. This delay is only for evaluation purposes.
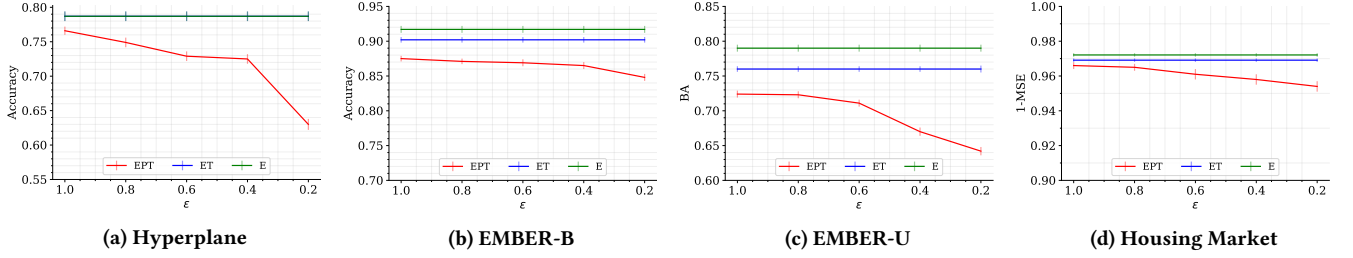
Figure 1: *Impact of privacy* (Comparing EPT with ET and E). The vertical bars represent the standard errors. For Hyperplane, there is no transfer learning, hence, EPT is same as EP and ET is same as E.
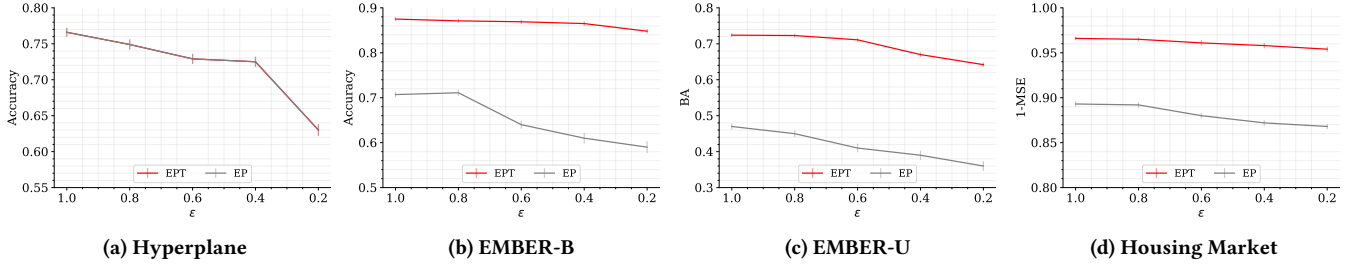

Figure 2: *Impact of transfer learning* (Comparing EPT vs EP). EPT and EP are identical for Hyperplane.

Section 7.3 studies the utility loss of our DP method compared to *non-private* counterparts, followed by the ablation studies evaluating the impact of transfer learning (Section 7.4), ensemble (Section 7.5), drift dynamics and chunk sizes (Section 7.6), and update mode (Section 7.7).

7.3 Impact of Privacy Preservation

The first question is how privacy preservation impacts the performance. To answer this question, we compare the performance of EPT with the non-private counterparts ET and E in Figure 1. The main finding is that EPT provides *close* utility (average difference of $< 3\%$) to ET for $\varepsilon = 1$. The utility gap increases as ε decreases, with the average drop of 9% at $\varepsilon = 0.2$. Our privacy settings are much tighter than those in the DPNN literature, for example, the minimum and maximum values of ε are 2 and 100 according to the survey [24]. Comparing the non-private models E and ET, transfer learning does not help. However, as we will show later, transfer learning significantly boosts the utility in the case of private models.

From now, we consider only the privacy preserving methods, i.e., EPT, EP, PT(1), and PT(k).

7.4 Impact of Transfer Learning

Figure 2 studies the effect of transfer learning by comparing EPT against the non-transfer learning counterpart EP (Note that EPT and EP are same for Hyperplane that has no transfer learning). EPT outperforms EP by a *significant* margin, with the average boost $> 10\%$ for EMBER-B and EMBER-U, and 7% for Housing Market, for all settings of ε. As the privacy budget gets tighter (ε decreases), EP decays in performance faster than EPT. This study supports our claim at the beginning of this section that transfer learning can boost the utility for training private models by reducing the number of trainable parameters for DPSGD.

From now, we consider only DP methods with transfer learning, i.e., EPT, PT(1), and PT(k).

7.5 Impact of Ensemble

To investigate how the ensemble approach helps, in Figure 3 we compare EPT with the non-ensemble counterparts PT(1) and PT(k). The first row varies ε and the second row varies k. With the varying ε, EPT outperforms the non-ensemble competitors consistently because our novel DP weight mechanisms diminish the weights for outdated models. PT(k), in general, performs better than PT(1), except for EMBER-B, because PT(1) uses a single data chunk, which leads to a larger σ for the Gaussian noise, as discussed at the beginning of this section. When $k \geq 5$, there is some performance decline for EPT because more outdated data chunks are used in an ensemble and because the budget $\varepsilon_2 = \varepsilon/k$ for weight estimation gets tighter, but this decline is smaller than that for PT(k) because of the "auto-correction" due to the weighting scheme in EPT. In our evaluation, we observed $3 \geq k \leq 7$ perform the best.

7.6 Impact of Concept Drifts

Figure 4 shows the impact of four simulated drift types using Hyperplane. When drift is rapid, as the chunk size $|D_i|$ increases, the performance of EPT and PT(1) initially increases and then decreases due to increasing drift introduced within a data chunk. This decline trend is especially observed for PT(k) that uses the union of k chunks to train the model. So the chunk size is a double-edged sword for rapid drift: too small or too large will hurt. There is a similar trend for recurring drift. When drift is gradual, all methods benefit as $|D_i|$ increases because drift is introduced slowly. When drift is abrupt, both EPT and PT(1) adapt well, but PT(k) fails to learn in this case as it uses a *stale* model, i.e., an abrupt change occurs after the model training.
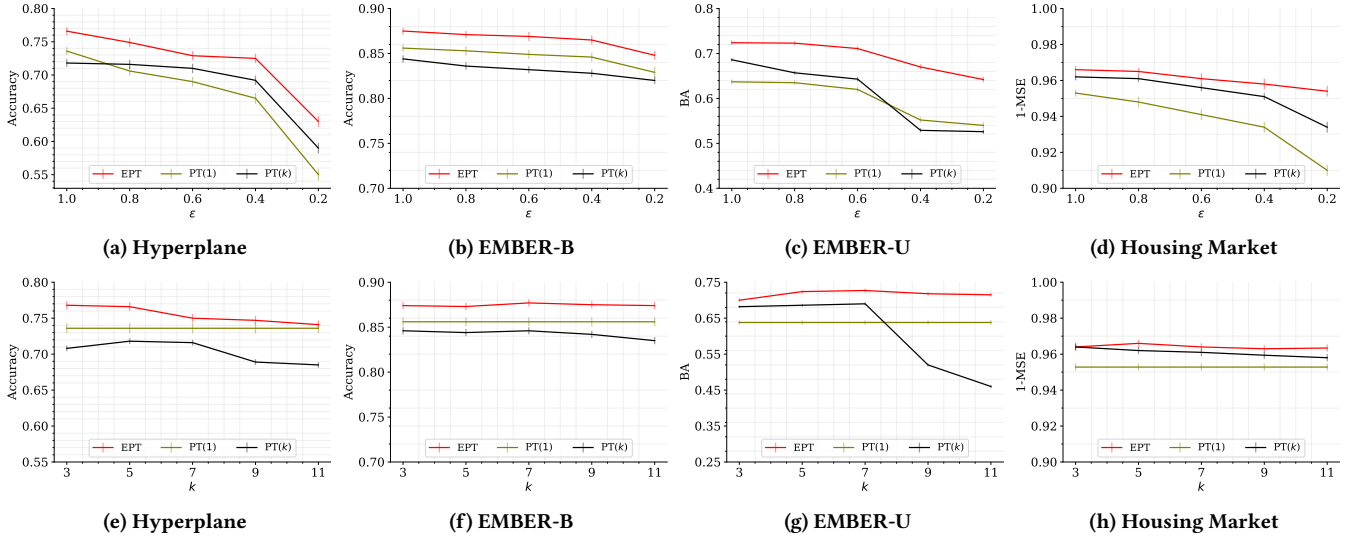
Figure 3: *Impact of ensemble* (Comparing EPT vs PT(1) and PT(k)). First row shows the comparison with varying ε while the second row shows the comparison with varying k.
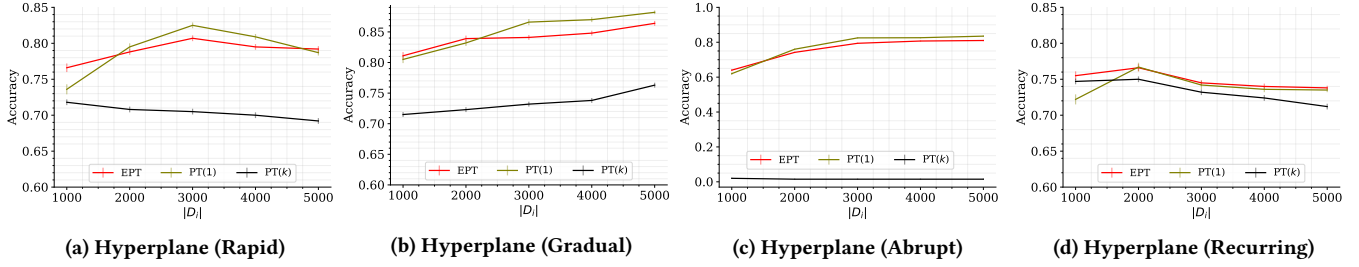

Figure 4: *Impact of drifts* (Comparing ensemble approaches for various drift types and chunk sizes.)
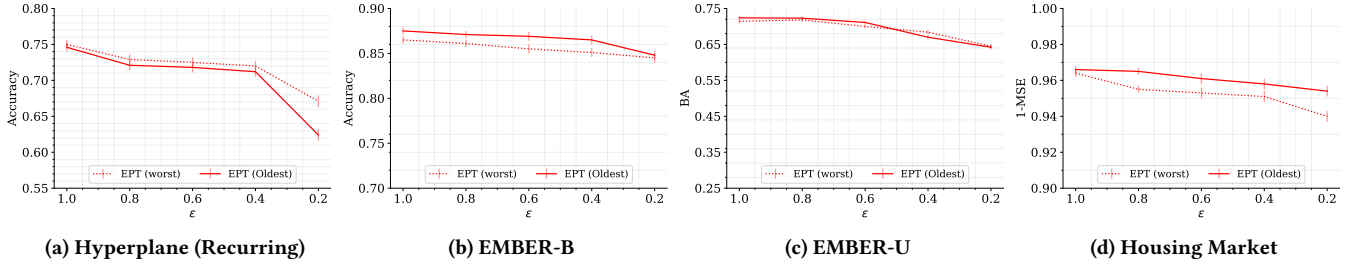

Figure 5: *Impact of model replacement* (Comparing replacing oldest model vs replacing worst model).

7.7 Impact of Model Replacement

Figure 5 shows the impact of replacing old model vs replacing worst model during the ensemble update of EPT. For Hyperplane, where we choose recurring drift, replacing the worst model is better than replacing the oldest model. For the other datasets, however, replacing the oldest model gives slightly better performance due to adding a smaller noise in the weight estimation, i.e., $\varepsilon_2 = \varepsilon/k$ vs $\varepsilon_2 = \varepsilon/k+1$. See Theorem 6 and Theorem 7.

8 CONCLUSION

We presented a practical DP solution to predictive modeling (both classification and regression) for data streams with concept drift.

To the best of our knowledge, this is the first work that allows an unbounded number of updates under a fixed privacy budget. The key component is a novel DP weighting mechanism for integrating the models in an ensemble. Our solution is model agnostic and can be used with any existing DP classification/regression method.

REFERENCES

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 308–318.

[2] Kevin Allix, Tegawendé F. Bissyandé, Jacques Klein, and Yves Le Traon. 2016. AndroZoo: Collecting Millions of Android Apps for the Research Community. In *Proceedings of the 13th International Conference on Mining Software Repositories* (Austin, Texas) *(MSR '16)*. ACM, New York, NY, USA, 468–471. https://doi.org/10.1145/2901739.2903508

[3] H. S. Anderson and P. Roth. 2018. EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models. *ArXiv e-prints* (April 2018). arXiv:1804.04637 [cs.CR]

[4] Gilles Barthe, Gian Pietro Farina, Marco Gaboardi, Emilio Jesús Gallego Arias, Andy Gordon, Justin Hsu, and Pierre-Yves Strub. 2016. Differentially private bayesian programming. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. 68–79.

[5] Kay Henning Brodersen, Cheng Soon Ong, Klaas Enno Stephan, and Joachim M Buhmann. 2010. The balanced accuracy and its posterior distribution. In *2010 20th international conference on pattern recognition*. IEEE, 3121–3124.

[6] Dariusz Brzeziński and Jerzy Stefanowski. 2011. Accuracy updated ensemble for data streams with concept drift. In *International conference on hybrid artificial intelligence systems*. Springer, 155–163.

[7] Alberto Cano and Bartosz Krawczyk. 2020. Kappa updated ensemble for drifting data stream mining. *Machine Learning* 109, 1 (2020), 175–218.

[8] CIHI. 2020. Discharge Abstract Database. https://www.cihi.ca/en/discharge-abstract-database-metadata.

[9] Andrea Dal Pozzolo, Giacomo Boracchi, Olivier Caelen, Cesare Alippi, and Gianluca Bontempi. 2015. Credit card fraud detection and concept-drift adaptation with delayed supervised information. In *2015 international joint conference on Neural networks (IJCNN)*. IEEE, 1–8.

[10] Gianmarco De Francisci Morales, Albert Bifet, Latifur Khan, Joao Gama, and Wei Fan. 2016. Iot big data stream mining. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 2119–2120.

[11] Danilo Rafael de Lima Cabral and Roberto Souto Maior de Barros. 2018. Concept drift detection based on Fisher's Exact test. *Information Sciences* 442 (2018), 220–234.

[12] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. 2006. Our data, ourselves: privacy via distributed noise generation. *EUROCRYPT* (2006), 486–503.

[13] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating Noise to Sensitivity in Private Data Analysis. In *Proceedings of the Third Conference on Theory of Cryptography* (New York, NY) *(TCC'06)*. Springer-Verlag, Berlin, Heidelberg, 265–284. https://doi.org/10.1007/11681878_14

[14] Cynthia Dwork, Moni Naor, Toniann Pitassi, Guy N Rothblum, and Sergey Yekhanin. 2010. Pan-Private Streaming Algorithms.. In *ICS*. 66–80.

[15] Cynthia Dwork, Aaron Roth, et al. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science* (2014).

[16] Ahmed Ismail Ebada, Samir Abdelrazek, and Ibrahim Elhenawy. 2020. Applying Cloud Based Machine Learning on Biosensors Streaming Data for Health Status Prediction. In *2020 11th International Conference on Information, Intelligence, Systems and Applications (IISA*. IEEE, 1–8.

[17] Ryan Elwell and Robi Polikar. 2011. Incremental learning of concept drift in nonstationary environments. *IEEE Transactions on Neural Networks* 22, 10 (2011), 1517–1531.

[18] Maryam Fanaeepour and Ashwin Machanavajjhala. 2019. PrivStream: differentially private event detection on data streams. In *Proceedings of the Ninth ACM Conference on Data and Application Security and Privacy*. 145–147.

[19] Xianjin Fang, Qingkui Zeng, and Gaoming Yang. 2020. Local Differential Privacy for Data Streams. In *International Conference on Security and Privacy in Digital Economy*. Springer, 143–160.

[20] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. 2014. A survey on concept drift adaptation. *ACM computing surveys (CSUR)* 46, 4 (2014), 1–37.

[21] Ömer Gözüaçık, Alican Büyükçakır, Hamed Bonab, and Fazli Can. 2019. Unsupervised concept drift detection with a discriminative classifier. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 2365–2368.

[22] Geoff Hulten, Laurie Spencer, and Pedro Domingos. 2001. Mining time-changing data streams. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. 97–106.

[23] Geetha Jagannathan, Krishnan Pillaipakkamnatt, and Rebecca N Wright. 2009. A practical differentially private random decision tree classifier. In *2009 IEEE International Conference on Data Mining Workshops*. IEEE, 114–121.

[24] Bargav Jayaraman and David Evans. 2019. Evaluating differentially private machine learning in practice. In *28th {USENIX} Security Symposium ({USENIX} Security 19)*. 1895–1912.

[25] Gautam Kamath and Jonathan Ullman. 2020. A primer on private statistics. *arXiv preprint arXiv:2005.00010* (2020).

[26] Georgios Kellaris, Stavros Papadopoulos, Xiaokui Xiao, and Dimitris Papadias. 2014. Differentially private event sequences over infinite streams. *Proceedings of the VLDB Endowment* 7, 12 (2014), 1155–1166.

[27] Michael Khavkin and Mark Last. 2018. Preserving Differential Privacy and Utility of Non-stationary Data Streams. In *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE, 29–34.

[28] J Zico Kolter and Marcus A Maloof. 2007. Dynamic weighted majority: An ensemble method for drifting concepts. *Journal of Machine Learning Research* 8, Dec (2007), 2755–2790.

[29] Paweł Ksieniewicz, Paweł Zyblewski, Michał Choraś, Rafał Kozik, Agata Giełczyk, and Michał Woźniak. 2020. Fake news detection from data streams. In *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–8.

[30] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, Joao Gama, and Guangquan Zhang. 2018. Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering* 31, 12 (2018), 2346–2363.

[31] Jan Lukes and Anders Søgaard. 2018. Sentiment analysis under temporal shift. In *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*. 65–71.

[32] Frank McSherry. 2009. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *SIGMOD*. ADM.

[33] Jacob Montiel, Jesse Read, Albert Bifet, and Talel Abdessalem. 2018. Scikit-Multiflow: A Multi-Output Streaming Framework. *J. Mach. Learn. Res.* 19, 1 (Jan. 2018), 2915–2914.

[34] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[35] Lena Pietruczuk, Leszek Rutkowski, Maciej Jaworski, and Piotr Duda. 2017. How to adjust an ensemble size in stream data mining? *Information Sciences* 381 (2017), 46–54.

[36] Santu Rana, Sunil Kumar Gupta, and Svetha Venkatesh. 2015. Differentially private random forest with high utility. In *2015 IEEE International Conference on Data Mining*. IEEE, 955–960.

[37] Benjamin IP Rubinstein, Peter L Bartlett, Ling Huang, and Nina Taft. 2009. Learning in a large function space: Privacy-preserving mechanisms for SVM learning. *arXiv preprint arXiv:0911.5708* (2009).

[38] Sberbank. 2017. *Sberbank Russian Housing Market Dataset*. https://www.kaggle.com/c/sberbank-russian-housing-market/data

[39] Martin Scholz and Ralf Klinkenberg. 2007. Boosting classifiers for drifting concepts. *Intelligent Data Analysis* 11, 1 (2007), 3–28.

[40] Reza Shokri and Vitaly Shmatikov. 2015. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*. ACM, 1310–1321.

[41] Haixun Wang, Wei Fan, Philip S Yu, and Jiawei Han. 2003. Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. 226–235.

[42] Tao Xiang, Yang Li, Xiaoguo Li, Shigang Zhong, and Shui Yu. 2018. Collaborative ensemble learning under differential privacy. In *Web Intelligence*, Vol. 16. IOS Press, 73–87.

[43] Limin Yang, Arridhana Ciptadi, Ihar Laziuk, Ali Ahmadzadeh, and Gang Wang. 2021. BODMAS: An Open Dataset for Learning based Temporal Analysis of PE Malware. In *Proceedings of Deep Learning and Security Workshop (DLS), in conjunction with IEEE Symposium on Security and Privacy (IEEE SP)*.

[44] Shujian Yu and Zubin Abraham. 2017. Concept drift detection with hierarchical hypothesis testing. In *Proceedings of the 2017 SIAM International Conference on Data Mining*. SIAM, 768–776.

[45] Guozheng Zhang and Shuyu Li. 2019. Research on differentially private bayesian classification algorithm for data streams. In *2019 IEEE 4th International Conference on Big Data Analytics (ICBDA)*. IEEE, 14–20.