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].

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 \( E_t \), consists of \( k \) models \( M_{t(1)}, \cdots, M_{t(k)} \) trained on the \( k \) corresponding data chunks \( D_{t(1)}, \cdots, D_{t(k)} \). \( \tau(i) \) maps the (relative) position \( i \) within \( E_t \) 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 \( E_{t+1} \) is obtained by replacing either the oldest model or the weakest model in \( E_t \) 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 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 \( f : \mathbb{D}^n \rightarrow \mathbb{R}^d \), denoted by \( \Delta f \), is \( \max_{D,D'} |f(D) - f(D')| \) over all neighboring pairs D and D’.

Definition 2 (Differential Privacy [12]). A randomized mechanism \( M : \mathbb{D}^n \rightarrow \mathbb{R}^d \) is \((\epsilon,\delta)\)-differentially private if for any pair of neighboring data sets \( D, D' \in \mathbb{D}^n \), and for all sets S of possible outputs:

\[
Pr[M(D) \in S] \leq e^\epsilon Pr[M(D') \in S] + \delta
\]

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

Theorem 1 (Parallel Composition [32]). Let \( M_i \) each provide \((\epsilon_i,\delta_i)\)-differential privacy. Let \( D_1 \) be arbitrary disjoint subsets of \( D \). The sequence of \( M_i(D_1) \) provides \((\max_i \epsilon_i, \max_i \delta_i)\)-differential privacy.

Theorem 2 (Sequential Composition [32][15]). Let \( M_i \) be \((\epsilon_i,\delta_i)\)-differentially private. The adaptive sequence of \( M_i \) is \((\Sigma_i \epsilon_i, \Sigma_i \delta_i)\)-differentially private.

Theorem 3 (Post Processing [15]). Let \( M : \mathbb{D}^n \rightarrow \mathbb{R}^d \) be a randomized mechanism that is \((\epsilon,\delta)\)-differentially private. Let \( f : \mathbb{R}^d \rightarrow \mathbb{R}^t \) be a deterministic function. Then \( f \circ M : \mathbb{D}^n \rightarrow \mathbb{R}^t \) is \((\epsilon,\delta)\)-differentially private.

Definition 3 (Laplace Mechanism [13]). Given any function \( f : \mathbb{D}^n \rightarrow \mathbb{R}^d \), the Laplace mechanism is defined as: \( M(D, f, \epsilon) = f(D) + (Y_1, \ldots, Y_d) \), where \( Y_i \) are i.i.d. random variables drawn from \( \text{Lap}(\Delta f / \epsilon) \).

Theorem 4 (DP of Laplace Mechanism [13]). The Laplace mechanism is \((\epsilon,0)\)-differentially private.

4 PROBLEM STATEMENT

We consider a data stream \( D_1, \ldots, D_t \) 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 \( D_{t+1} \) 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 \( t \), \( E_t = \{M_{\tau(1)}, \ldots, M_{\tau(k)}, \omega_{\tau(1)}, \ldots, \omega_{\tau(k)}\} \), consists of \( k \) prediction models \( M_i \) fs respectively trained on \( D_{\tau(1)}, \ldots, D_{\tau(k)} \), and their weights \( \omega_i \) where \( \tau(i) \) denotes the (absolute) time point corresponding to the (relative) position \( i \) within \( E_t \). We assume \( \tau(1), \ldots, \tau(k) \) 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 \( E_t \) is computed by

\[
E_{t,c}(x) = \frac{\sum_{i=1}^{k} \omega_{\tau(i)} \cdot M_{\tau(i),c}(x)}{\sum_{i=1}^{k} \omega_{\tau(i)}}
\]

The predicted class for \( x \) is the class \( c \) with the maximum \( E_{t,c}(x) \).

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 \( E_t \) is given by

\[
E_t(x) = \frac{\sum_{i=1}^{k} \omega_{\tau(i)} \cdot M_{\tau(i)}(x)}{\sum_{i=1}^{k} \omega_{\tau(i)}}
\]
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 \( \tilde{E}_t \) to \( \tilde{E}_{t+1} \) by replacing either the oldest model or the worst (i.e., with smallest weight) model in \( \tilde{E}_t \).

Problem of \((\epsilon, \delta)\)-DP Temporal Ensembles. The focus of this work is the scenario where each \( D_t \) contains sensitive, private information about the contributing users. We assume that each sample in \( \bigcup D_t \) has a unique identifier and all samples are independently generated. This independence assumption would allow us to treat all \( D_t \) as disjoint subsets of \( \bigcup D_t \). We want to ensure that, at any current time \( t \), the entire history of released ensembles up to \( t \), i.e., \( \tilde{E}'_t \) for all \( t' \leq t \), satisfy \((\epsilon, \delta)\)-DP for given \( \epsilon, \delta \). A formal definition of \((\epsilon, \delta)\)-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 \( \epsilon \) is split into \( \epsilon_1 \) for training models \( M_k \) and \( \epsilon_2 \) for computing noisy weights \( \tilde{w}_i \), where \( \epsilon = \epsilon_1 + \epsilon_2 \).

5 NOISY WEIGHT ESTIMATION

We assume that the labeled data \( D_t \) is split into training, validation, and testing subsets. For an ensemble \( E_t \) at time \( t \), the weights \( w_{t(1)}, \ldots, w_{t(k)} \) 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 \( E_t \) 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 \( w_{t(1)}, \ldots, w_{t(k)} \) 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 \( E_t \) 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]:

\[
\text{MSE}_i = \frac{1}{|V_t|} \text{Err}_t
\]

where

\[
\text{Err}_t = \sum_{x \in V_t} (1 - M_i.c(x))^2
\]

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

\[
\text{MSE}_r = \sum_{c} p(c)(1 - p(c))^2
\]

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

\[
w_t = \max(0, \text{MSE}_r - \text{MSE}_i)
\]

Lemma 1 (General Setting). Let \( \Delta \text{Err}_t \) denote the sensitivity of \( \text{Err}_t \) defined over all neighboring validation subsets \( V_t, V'_t \). \( \Delta \text{Err}_t = 1 \).

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 \( \text{Err}_t \) differs by at most 1 because \( M_i.c(x) \) is at most 1.

Computing Noisy Weight \( \tilde{w}_i' \): \( \tilde{w}_i' \) is computed from Equation (7) and (4) using the noisy \( \text{Err}_t' \):

\[
\text{Err}_t' = \text{Err}_t + \text{Lap}(\Delta\text{Err}_t/\epsilon)
\]

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):

\[
BA = a_1 \cdot TPR + a_2 \cdot TNR
\]

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 \( D_t \), 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 \( B Ai \) for \( M_i \) as follows:

\[
BA_i = a_1 \frac{TP_t}{|V_t|} + a_2 \frac{TN_t}{|V_t|}
\]

where \( TP_t \) (resp. \( TN_t \)) 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 \( \Delta BA_i \) denote the sensitivity of \( BA_i \) defined over neighboring pairs \( V_t, V'_t \). \( \Delta BA_i = \frac{a_1}{|V_t|} \max(\frac{a_1}{p}, \frac{a_2}{1-p}) \).

Proof. Consider neighboring validation subsets \( V_t, V'_t \) 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:

\[
|BA - BA'| \leq \frac{1}{|V_t|} \max(\frac{a_1}{p}, \frac{a_2}{1-p})
\]
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 \(V_t\)s is required (for the purpose of statistical significance), we can use the minimum size in \(\Delta A_t\) without referring to specific \(|V_t|\).

**Computing Noisy Weight \(w^*_i\):** We add the Laplace noise:

\[
w^*_i = BA_i + \text{Lap}(\Delta A_i/\varepsilon) \tag{11}
\]

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 \(M_t\). Then we get \(MSE_i\) as

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

and define the weight for \(M_t\) as

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

\(\mu\) 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 \(\Delta Err_i\) 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

\[
Err^*_i = Err_i + \text{Lap}(\Delta Err_i/\varepsilon) \tag{15}
\]

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 \(M_t\) as black-boxes. This is because the computation of the weights \(w_i\) only depends on the outputs, not the internal working of \(M_t\).

6 DP TEMPORAL ENSEMBLE

In Section 6.1, we provide the privacy analysis for a single \((\varepsilon, \delta)\)-DP ensemble \(E_t = \{M_{t(1)}, \ldots, M_{t(k)}, w^*_{t(1)}, \ldots, w^*_{t(k)}\}\), where each model \(M_t\) is trained using any method on the training subset \(S_t\) of \(D_t\) and its weight \(w^*_t\) is computed using the validation subset \(V_t\) of \(D_t\). In Section 6.2, we update \(E_t\) to \(E_{t+1}\) and present the privacy analysis for releasing all ensembles \(E_1, \ldots, E_t\) 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 \(E_t\). Let \(X_t = <S_{t(1)}, \ldots, S_{t(k)}, V_t>\), where \(S_{t(1)}, \ldots, S_{t(k)}\) are the training subsets for \(M_{t(1)}, \ldots, M_{t(k)}\) and \(V_t\) is the validation subset of \(D_t\) for computing the noisy weights \(w^*_{(1)}, \ldots, w^*_{t(k)}\).

**Definition 4 (Neighboring Datasets for Ensembles).** Consider \(X_t = <S_{t(1)}, \ldots, S_{t(k)}, V_t>\) and \(X'_t = <S'_{t(1)}, \ldots, S'_{t(k)}, V'_t>\). We say that \(X_t\) and \(X'_t\) are neighboring if \(\cup S_j \cup V_t\) and \(\cup S'_j \cup V'_t\) (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 \(i, S_t\) and \(S'_t\) 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 \(R_t\) to the range of \(E_t\) is \((\varepsilon, \delta)\)-differentially private if for all neighboring 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 \(M_j\) in \(E_t\) is produced by a model-agnostic \((\varepsilon, \delta)\)-DP mechanism \(A\) and that the noisy weight \(w^*_j\) is produced by the Laplace mechanism \(L\) in Section 5. Then the combined mechanism that produces \(E_t = \{M_{t(1)}, \ldots, M_{t(k)}, w^*_{t(1)}, \ldots, w^*_{t(k)}\}\) is \((\max(\varepsilon_j, k \cdot \varepsilon), \delta)\)-differentially private.

**Proof.** For simplicity of proof, we write \(\tau(1), \ldots, \tau(k)\) as \(1, \ldots, k\). The \((\varepsilon_j, \delta)\)-DP guarantee of \(A\) implies that for neighboring training subsets \(S_j, S'_j\) (Def. 4), and for any possible set \(M_j\) of outputs:

\[
\Pr[A(S_j) \in M_j] \leq e^{\varepsilon_j} \Pr[A(S'_j) \in M_j] + \delta \tag{17}
\]

For the weight calculation, the Laplace mechanism \(L\) provides \((\varepsilon_j, 0)\)-DP for releasing the noisy weights \(w^*_j\) following Theorem 4. Therefore, for any \(M_j \in M_t\), neighboring validation subsets \(V_j, V'_j\), and any possible set \(W^*_j\) of weights:

\[
\Pr[L(M_j(V_j)) \in W^*_j] \leq e^\varepsilon_j \Pr[L(M_j(V'_j)) \in W^*_j] \tag{18}
\]

Denoting our combined mechanism as \(C\), with input \(X_t = <S_1, \ldots, S_k, V_t>\), and any possible set of outputs \(O = \{M_1, \ldots, M_k, W^*_1, \ldots, W^*_k\}\), we get:

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

Now consider the only two possible cases of neighboring \(X_t\) as \(<S_1, \ldots, S_k, V_t>\) and \(<S'_1, \ldots, S'_k, V'_t>\): (I) change one arbitrary \(S_i\) or (II) change \(V_t\). For Case (I), all models should satisfy Equation (17), but since only \(S_i\) changes to reach a neighboring input, Equation (17) will be obtained on one \(M_i\) and for all others \(j \neq i\) we would get \(\Pr[A(S_j) \in M_j] = \Pr[A(S'_j) \in M_j]\). Additionally, since in this case we are \textbf{not} changing \(V_t, V'_t\), for all \(i \leq k, \Pr[L(M_i(V_t)) \in W^*_i] = \Pr[L(M_i(V'_t)) \in W^*_i]\). Combining these two facts we reach \(\Pr[C(X_t) \in O] \leq e^{\varepsilon} \Pr[C(X'_t) \in O] + \delta\) from Equation (19) above.

For Case (II), we do not change any of \(S_1, \ldots, S_k\), thus for all \(1 \leq i \leq k, \Pr[A(S_i) \in M_j] = \Pr[A(S'_i) \in M_j]\). 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 \(\Pr[C(X_t) \in O] \leq e^{k\varepsilon_j} \Pr[C(X'_t) \in O]\) from Equation (19) above.

Finally, since DP must hold for the worst-case guarantee, we take the minimum between the two cases defined above, which gives us the \((\max(\varepsilon_j, k \cdot \varepsilon), \delta)\)-DP. Note that combining the two cases is a tailored instantiation of the parallel composition (Theorem 1).
Discussion. The construction of $(\epsilon_1, \delta)$-DP mechanism $\mathcal{A}$ for training a single model $M_t$ 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 $(\epsilon, \delta)$-DP mechanisms $\mathcal{L}$ for building an ensemble $E_t = \{M_{r(1)}, \ldots, M_{r(k)}, w^*_{r(1)}, \ldots, w^*_{r(k)}\}$, 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 $E_t = \{M_{r(1)}, \ldots, M_{r(k)}, w^*_{r(1)}, \ldots, w^*_{r(k)}\}$, the training and validation subsets at $t+1$, i.e., $S_{t+1}$ and $V_{t+1}$; privacy parameters $(\epsilon_1, \epsilon_2, \delta)$; update_mode $(\text{oldest or worst})$.

Output: $E_{t+1}$

1: Train $(\epsilon_1, \delta)$-DP model $M_{t+1}$ on $S_{t+1}$
2: if update_mode = “oldest” then
3: Calculate $w^*_{r(1)}, \ldots, w^*_{r(i)}, w^*_{r(i+1)}$ for $M_{r(1)}, \ldots, M_{r(k)}, M_{t+1}$ using $V_{t+1}$ and $E_2$
4: $w^*_{r(1)}, \ldots, w^*_{r(k)}, w^*_{r(k-1)} \leftarrow w^*_{r(2)}, \ldots, w^*_{r(k)}, w^*_{r(k+1)}$
5: $M_{r(1)}, \ldots, M_{r(k)}, M_{t+1} \leftarrow M_{r(2)}, \ldots, M_{r(k)}, M_{t+1}$
6: $E_{t+1} = \{M_{r(1)}, \ldots, M_{r(k)}, w^*_{r(1)}, \ldots, w^*_{r(k)}\}$
7: else
8: Calculate $w^*_{r(1)}, \ldots, w^*_{r(i)}, w^*_{r(i+1)}$ for $M_{r(1)}, \ldots, M_{r(k)}, M_{t+1}$ using $V_{t+1}$ and $E_2$
9: $w^*_{r(1)}, \ldots, w^*_{r(k)}, w^*_{r(k+1)} \leftarrow w^*_{r(2)}, \ldots, w^*_{r(k)}, w^*_{r(k+1)}$
10: $i^* \leftarrow \text{argmin}_i \{\|w^*_{r(i)}\| \mid 1 \leq i \leq k + 1\}$
11: $E_{t+1} \leftarrow \{M_{r(i)}, w^*_{r(i)} \mid 1 \leq i \leq k + 1, i \neq i^*\}$
12: end if
13: return $E_{t+1}$

Algorithm 1 shows the steps for updating the ensemble $E_t$ 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 $w^*$). 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 $E_t$ satisfies $(\max\{\epsilon_1, k \cdot \epsilon_2\}, \delta)$-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 $(\max\{\epsilon_1, k \cdot \epsilon_2\}, \delta)$-DP remains to hold in the case of replacing oldest model and degrades to $(\max\{\epsilon_1, (k+1) \cdot \epsilon_2\}, \delta)$-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., $X = \langle X_1, \ldots, X_t \rangle$ where $X_i$ is the input data for the ensemble $E_i$ 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 $j \neq i$, $X_j = X'_j$. We consider the output consisting of all ensembles released up to the time $t$, i.e., $\mathcal{E} = \{E_1, \ldots, E_t\}$.

Definition 6 (Differential Privacy for History). A mechanism $\mathcal{C}$ from the domain of $X$ to the range of $\mathcal{E}$ is $(\epsilon, \delta)$-differentially private with respect to history if for any neighbouring pair $(X, X')$ and for all sets $O$ of possible outputs:

$$\Pr[C(X) \in O] \leq e^\epsilon \Pr[C(X') \in O] + \delta$$  (20)

Theorem 6. With update_mode=“oldest”, Algorithm 1 is $(\max\{\epsilon_1, k \cdot \epsilon_2\}, \delta)$-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 $(\epsilon_1, \delta)$-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 $E_t$), which ensures the $(k \cdot \epsilon_2, \delta)$-DP remains unchanged.

Theorem 7. With update_mode=“worst”, Algorithm 1 is $(\max\{\epsilon_1, (k+1) \cdot \epsilon_2\}, \delta)$-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 $(\max\{\epsilon_1, (k+1) \cdot \epsilon_2\}, \delta)$-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 $(\epsilon, \delta)$, we can set $\epsilon_1 = \epsilon$ and $\epsilon_2 = \epsilon/k$ when replacing oldest model (Theorem 6), and set $\epsilon_1 = \epsilon$ and $\epsilon_2 = \epsilon/k+1$ 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   | 30%   | 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 $M_t$ as a neural network using DPSGD [1] with privacy budget $(\epsilon_1, \delta)$. In each iteration, DPSGD adds the Gaussian noise $\mathcal{N}(0, \sigma^2C^2)$ to the clipped gradient $\frac{g(x_i)}{\max\{1, |\|x_i\|\sigma\}}$, where $C$ is the clipping factor. For a large number of model parameters, $\|g(x_i)\|\sigma$ 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 $\sigma$. To reduce the norm $\|g(x_i)\|\sigma$, we also consider the option of transfer learning for training $M_t$; 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 \( D_t \) via DP-SGD keeping the parameters for other layers unchanged. If no such public \( P \) is available, \( M_t \) will be fully trained using the sensitive \( D_t \) via DP-SGD.

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 \( D_i \)s 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 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 \( D_i \)s 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 \( D_i \)s 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 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 \( D_i \)s, 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 \( D_t \).

For all datasets: we standardize continuous features using StandardScaler from scikit-learn[34] and use the one-hot encoding for categorical features. We run DP-SGD with 30 epochs with the mini-batch size of 100. The labeled data \( D_t \) 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          | ✓       | ✓        | ✓        | \([D_{t(1)}, \ldots, D_{t(k)}]\) |
| EP           | ✓       | ✓        | ✗        | \([D_{t(1)}, \ldots, D_{t(k)}]\) |
| PT(1)        | ✓       | ✗        | ✓        | \([D_{t(1)}]\)         |
| PT(k)        | ✓       | ✗        | ✓        | \([D_{t(1)} \cup \ldots \cup D_{t(k)}]\) |
| ET           | ✗       | ✓        | ✓        | \([D_{t(1)}, \ldots, D_{t(k)}]\) |
| E            | ✗       | ✓        | ✗        | \([D_{t(1)}, \ldots, D_{t(k)}]\) |

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 (\( M_i \)s) trained on the \( k \) data chunks \( D_{t(1)}, \ldots, D_{t(k)} \) 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 \( D_{t(1)} \cup \cdots \cup D_{t(k)} \) of \( k \) chunks to train the model and advances to the next non-overlapping window covering times \( t + 1, \ldots, t + k \), 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., \( t + 1, \ldots, t + k \)).

All DP methods are evaluated under the same privacy budget \((\epsilon, \delta)\). The following default settings are used: \( \epsilon = 1, \delta = 0.0001, \) window size \( k = 5, \) drift type = “rapid”, and update_mode = “oldest”. We set \( \epsilon_1 = \epsilon \) and \( \epsilon_2 = \epsilon/k \) for update_mode = “oldest”, and set \( \epsilon_1 = \epsilon \) and \( \epsilon_2 = \epsilon/(k+1) \) 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.
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 $\epsilon = 1$. The utility gap increases as $\epsilon$ decreases, with the average drop of 9% at $\epsilon = 0.2$. Our privacy settings are much tighter than those in the DPN literature, for example, the minimum and maximum values of $\epsilon$ 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 $\epsilon$. As the privacy budget gets tighter ($\epsilon$ 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 $\epsilon$ and the second row varies $k$. With the varying $\epsilon$, 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 $\sigma$ 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 $\epsilon$ is $\epsilon/k$ 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.
We presented a practical DP solution to predictive modeling (both classification and regression) for data streams with concept drift. Our solution is model agnostic and can be used with any existing DP classification/regression method.

This research is in part supported by a CGS-D award for Lovedeep Gondara and a discovery grant for Ke Wang from Natural Sciences and Engineering Research Council of Canada.

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.
