# Communication-Efficient Distributed Online Prediction by Dynamic Model Synchronization*

Michael Kamp[1], Mario Boley[1], Daniel Keren[2], Assaf Schuster[3], and Izchak Sharfman[3]

[1] Fraunhofer IAIS & University Bonn, {surname.name}@iais.fraunhofer.de
[2] Haifa University, dkeren@cs.haifa.ac.il
[3] Technion, Israel Institute of Technology, {assaf,tsachis}@technion.ac.il

**Abstract.** We present the first protocol for distributed online prediction that aims to minimize online prediction loss and network communication at the same time. This protocol can be applied wherever a prediction-based service must be provided timely for each data point of a multitude of high frequency data streams, each of which is observed at a local node of some distributed system. Exemplary applications include social content recommendation and algorithmic trading. The challenge is to balance the joint predictive performance of the nodes by exchanging information between them, while not letting communication overhead deteriorate the responsiveness of the service. Technically, the proposed protocol is based on controlling the variance of the local models in a decentralized way. This approach retains the asymptotic optimal regret of previous algorithms. At the same time, it allows to substantially reduce network communication, and, in contrast to previous approaches, it remains applicable when the data is non-stationary and shows rapid concept drift. We demonstrate empirically that the protocol is able to hold up a high predictive performance using only a fraction of the communication required by benchmark methods.

## 1 Introduction

We consider distributed online prediction problems on multiple connected high-frequency data streams where one is interested in minimizing predictive error and communication at the same time. This situation abounds in a wide range of machine learning applications, in which communication induces a severe cost. Examples are parallel data mining [Zinkevich et al., 2009, Hsu et al., 2011] and M2M communication [Wang et al., 2013] where communication constitutes a performance bottleneck, learning with mobile sensors [Nguyen et al., 2004, Predd et al., 2006] where communication drains battery power, and, most centrally, prediction-based real-time services [Dekel et al., 2012] carried out by several servers, e.g., for social content promotion, ad placement, or algorithmic trading.

* A preliminary extended abstract of this paper was presented at the BD3 workshop at VLDB'13. This research has been supported by the EU FP7-ICT-2013-11 under grant 619491 (FERARI).

Here, due to network latency, the cost of communication can also be a loss of prediction quality itself, because, in order to avoid inconsistent system states some data points have to be discarded for learning whenever a communication event is triggered. In this paper, we abstract on all these various motivations and provide a protocol that aims to minimize communication as such. In particular, we provide the first protocol that dynamically adapts communication to exploit the communication reduction potential of well-behaved input sequences but at the same time retains the predictive performance of static communication schemes.

In contrast to work on the communication complexity of batch learning [Bar-Or et al., 2005, Ouyang et al., 2009, Balcan et al., 2012, Daumé III et al., 2012], we consider the online in-place performance of a streaming distributed prediction system. For this setting, earlier research focused on strategies that communicate periodically after a fixed number of data points have been processed [Mann et al., 2009, Dekel et al., 2012]. For these static communication schemes Dekel et al. [2012] shows that for smooth loss functions and stationary environments optimal asymptotic regret bounds can be retained by updating a global model only after observing a mini-batch of examples. While such a fixed periodic communication schedule reduces the communication, further reduction is desirable: the above mentioned costs of communication can have a severe impact on the practical performance—even if they are not reflected in asymptotic performance bounds. Moreover, distributed learning systems can experience periodical or singular target drifts. In these settings, a static schedule is bound to either provide only little to none communication reduction or to insufficiently react to changing data distributions.

In this work, we give the first data-dependent distributed prediction protocol that dynamically adjusts the amount of communication performed depending on the hardness of the prediction problem. It aims to provide a high online in-place prediction performance and, at the same time, explicitly tries to minimize communication. The underlying idea is to perform model synchronizations only in system states that show a high variance among the local models, which indicates that a synchronization would be most effective in terms of its correcting effect on future predictions. While the model variance is a non-linear function in the global system, we describe how it can be monitored locally in a communication-efficient way. The resulting protocol allows communicative quiescence in stable phases, while, in hard phases where variance reduction is crucial, the protocol will trigger a lot of model synchronizations. Thus, it remains applicable when the data is non-stationary and shows rapid concept drifts—cases in which a static scheme is doomed to either require a high communication frequency or suffer from low adaption. We show theoretically (Sec. 3.1), that, despite the communication reduction achieved by our dynamic protocol, it retains any shifting regret bounds provided by its static counterpart. We also demonstrate its properties empirically (Sec. 4) with controlled synthetic data and real-world datasets from stock markets and the short-message service Twitter.

## 2 Preliminaries

In this section we formally introduce the distributed online prediction task. We recall simple sequential learning algorithms and discuss a basic communication scheme to utilize them in the distributed scenario.

### 2.1 Distributed Online Prediction

Throughout this paper we consider a distributed online prediction system of $k$ **local learners** that maintain individual **linear models** $w_{t,1}, \ldots, w_{t,k} \in \mathbb{R}^n$ of some global environment through discrete time $t \in [T]$ where $T \in \mathbb{N}$ denotes the total time horizon with respect to which we analyze the system's performance. This environment is represented by a **target distribution** $\mathcal{D}_t \colon X \times Y \to [0,1]$ that describes the relation between an input space $X \subseteq \mathbb{R}^n$ and an output space $Y \subseteq \mathbb{R}$. The nature of $Y$ varies with the learning task at hand; $Y = \{-1, 1\}$ is used for binary classification, $Y = \mathbb{R}$ for regression. Generally, we assume that all training examples $x \in X$ are drawn from a ball of **radius** $R$ and also that $x_n = 1$ for all $x \in X$, i.e., $\|x\| \in [1/n, R]$—two common assumptions in online learning (the latter avoids to explicitly fit a bias term of the linear models). All learners sample from $\mathcal{D}_t$ independently in parallel using a constant and uniform sampling frequency, and we denote by $(x_{t,l}, y_{t,l}) \sim \mathcal{D}_t$ the **training example** received at node $l$ at time $t$. Note that, while the underlying environment can change over time, we assume that at any given moment $t$ there is one fixed distribution governing the points observed at all local nodes.

Conceptually, every learner first observes the input part $x_{t,l}$ and performs a real time service based on the linear **prediction score** $p_{t,l} = \langle w_{t,l}, x_{t,l} \rangle$, i.e., the inner product of $x_{t,l}$ and the learner's current model vector. Only then it receives as feedback the true label $y_{t,l}$, which it can use to locally update its model to $w_{t+1,l} = \varphi(w_{t,l}, x_{t,l}, y_{t,l})$ by some **update rule** $\varphi \colon \mathbb{R}^n \times X \times Y \to \mathbb{R}^n$. Let $W_t \in \mathbb{R}^{k \times n}$ denote the complete **model configuration** of all local models at time $t$ (denoting by $w_{t,l}$ the model at learner $l$ at time $t$ as above). The learners are connected by a communication infrastructure that allows them to jointly perform a **synchronization operation** $\sigma : \mathbb{R}^{k \times n} \to \mathbb{R}^{k \times n}$ that resets the whole model configuration to a new state after local updates have been performed. This operator may take into account the information of all local learners simultaneously. The two components $(\varphi, \sigma)$ define a **distributed learning protocol** that, given the inputs of the environment, produces a sequence of model configurations $\mathbf{W} = W_1, \ldots, W_T$. Its performance is measured by:

1. the in-place predictive performance $\sum_{t=1}^{T} \sum_{l=1}^{k} f(w_{t,l}, x_{t,l}, y_{t,l})$ measured by a **loss function** $f : \mathbb{R}^n \times \mathbb{R}^n \times Y \to \mathbb{R}_+$ that assigns positive penalties to prediction scores based on how (in-)appropriately they describe the true label; and
2. the amount of **communication** within the system that is measured by the number of bits send in-between learners in order to compute the synchronization operation $\sigma$.

Regarding the predictive performance, one is typically interested in bounding the average **regret** of the model configurations produced by the protocol with respect to a **reference sequence** $\mathbf{U} = U_1, \ldots, U_T$. For technical reasons, in this paper we focus on the squared regret, i.e.,

$$R(\mathbf{W}, \mathbf{U}) = \frac{1}{T} \sum_{t=1}^{T} \frac{1}{k} \sum_{l=1}^{k} (f(w_{t,l}, x_{t,l}, y_{t,l}) - f(u_{t,l}, x_{t,l}, y_{t,l}))^2 \enspace .$$

This type of regret is often referred to as shifting regret (see, Herbster and Warmuth [2001]) and typically bounds are given in the total shift per node of the reference sequence $\sum_{t=1}^{T} \sum_{l=1}^{k} \|u_{t,l} - u_{t-1,l}\|^2$. Traditional results often restrict regret analysis to the case of a static reference sequence, i.e., $u_{1,1} = u_{1,2} = \cdots = u_{t,l}$. This is particularly useful if we consider the **stationary scenario** where $\mathcal{D}_1 = \cdots = \mathcal{D}_T$.

### 2.2 Loss-Proportional Convex Update Rules

Principally, the protocol developed in this paper can be applied to a wide range of update rules for online learning (from, e.g., stochastic gradient descend [Zinkevich et al., 2010] to regularized dual averaging [Xiao, 2010]). For the formal analysis, however, we focus on update rules covered by the following definition.

**Definition 1** *We call an update rule $\varphi$ an **$f$-proportional convex update** for a loss function $f$ if there are a constant $\gamma > 0$, a closed convex set $\mathit{\Gamma}_{x,y} \subseteq \mathbb{R}^n$, and $\tau_{x,y} \in (0, 1]$ such that for all $w \in \mathbb{R}^n$, $x \in X$, and $y \in Y$ it holds that*

*(i) $\|w - \varphi(w, x, y)\| \geq \gamma f(w, x, y)$, i.e., the update magnitude is a true fraction of the loss incurred, and*
*(ii) $\varphi(w, x, y) = w + \tau_{x,y} (P_{x,y}(w) - w)$ where $P_{x,y}(w)$ denotes the projection of $w$ onto $\mathit{\Gamma}_{x,y}$, i.e., the update direction is identical to the direction of a convex projection that only depends on the training example.*

As a first example for update rules satisfying these conditions, consider the **passive aggressive update rules** [Crammer et al., 2006]. These rules are defined for a variety of learning tasks including classification, regression, and uni-class prediction and can be uniformly described by

$$\varphi(w, x, y) = \arg \min_{w' \in \mathbb{R}^n} \frac{1}{2} \|w - w'\|^2 \;\; s.t. \;\; f(w', x, y) = 0 \qquad (1)$$

where for classification $f$ is the **hinge loss**, i.e., $f(w, x, y) = \max(1 - y\langle w, x\rangle, 0)$, for regression the **$\epsilon$-insensitive loss**, i.e., $f(w, x, y) = \max(|\langle w, x\rangle - y| - \epsilon, 0)$, and for uni-class prediction (where no $x$ is observed and $Y = \mathbb{R}^n$) the loss is given by $f(w, y) = \max(|w - y| - \epsilon, 0)$. It can be observed immediately that, in all three cases, these update rules are an actual projection on the convex set $\mathit{\Gamma}_{x,y} = \{w \in \mathbb{R}^n : f(w, x, y) = 0\}$, which corresponds to a half-space, a $2\epsilon$-strip, and an $\epsilon$-ball, respectively. Hence, Cond. (ii) of the definition follows immediately with $\tau_{x,y} = 1$. Cond. (i) can then be verified from the closed form solution of Eq. 1, which in case of classification is given by

$$\varphi(w,x,y) = w + \frac{f(w,x,y)}{\|x\|^2} yx \ .$$

Using the data radius $R$, we can easily bound the update magnitude from below as $\|w - \varphi(w,x,y)\| \geqslant R^{-1} f(w,x,y)$, i.e., Cond. (i) holds with $\gamma = R^{-1}$. The other cases follow similarly. Crammer et al. [2006] also gives other variants of passive aggressive updates that have a reduced learning rate determined by an aggressiveness parameter $C > 0$. These rules also satisfy the conditions of Def. 1. For example the rule for classification then becomes

$$\varphi(w,x,y) = w_t + \frac{f(w,x,y)}{\|x\|^2 + \frac{1}{2C}} yx \ .$$

Using $\|x\| \in [1/n, R]$, one can show that this variant remains hinge-loss proportional with $\gamma = n^{-1}(R^2 + 1/(2C))^{-1}$, and the update direction is identical to the same convex projection as in the standard case.

Another popular family of update rules for differentiable loss functions is given by **stochastic gradient descent**, i.e., rules of the form

$$\varphi(w,x,y) = w - \eta \nabla_w f(w,x,y)$$

with a positive learning rate $\eta > 0$. If one uses the squared hinge loss, $f(w,x,y) = 1/2 \max(1 - y\langle w,x\rangle, 0)^2$, we have $\nabla_w f(w,x,y) = y(1 - y\langle w,x\rangle)x$. Hence, this update rule is hinge loss proportional with $\gamma = \eta/n$, and the update direction is identical to the passive aggressive update rule for classification—that is, in the direction of a convex projection. The same can be checked for regression using the squared $\epsilon$-insensitive loss and many other variants of gradient descent.

In the following we will define a static averaging protocol that reduces the communication cost in a distributed online learning scenario and serves as baseline to our dynamic synchronization protocol.

### 2.3 Static Averaging

In terms of cost, every synchronization operator lies between two extreme baselines—constant broadcast of all training examples and quiescence, i.e., no communication at all. The predictive performance of these two extremes in terms of static regret lies between $O(\sqrt{kT})$ for serial learning (which is optimal for the stationary setting, see Cesa-Bianchi and Lugosi [2006] and Abernethy et al. [2009]) and $O(k\sqrt{T})$ for no communication, which corresponds to solving $k$ separate online learning problems in parallel.

An intermediate solution is to only reset all local models to their joint average every $b$ rounds where $b \in \mathbb{N}$ is referred to as **batch size** (see Mann et al. [2009] and Dekel et al. [2012]). Formally, this **static averaging operator** is given by $\sigma(W_t) = \left(\overline{W}_t, \ldots, \overline{W}_t\right)$ if $t \mod b = 0$ and $\sigma(W_t) = W_t$, otherwise. Here,

**Algorithm 1** Static Averaging Protocol

```
Initialization:
  local models $w_{1,1}, \dots, w_{1,k} \leftarrow (0, \dots, 0)$
Round $t$ at node $l$:
  observe $x_{t,l}$ and provide service based on $p_{t,l}$
  observe $y_{t,l}$ and update $w_{t+1,l} \leftarrow \varphi(w_{t,l}, x_t, y_t)$
  if $t \mod b = 0$ then
    send $w_{t,l}$ to coordinator
At coordinator every $b$ rounds:
  receive local models $\{w_{t,l} \colon l \in [k]\}$
  For all $l \in [k]$ set $w_{t,l} \leftarrow \sigma(w_{t,1}, \dots, w_{t,k})_l$
```

$\overline{W}_t = 1/k \sum_{l=1}^{k} w_{t,l}$ denotes the mean model. This choice of a (uniform) model mixture is often used for combining linear models that have been learned in parallel on independent training data (see also McDonald et al. [2010], Zinkevich et al. [2010]). The motivation is that the mean of $k$ models provides a variance reduction of $\sqrt{k}$ over an individual random model (recall that all learners sample from the same distribution, hence their models are identically distributed). For certain learning problems in the stationary setting, it can even be shown that this protocol retains the asymptotically optimal regret of $O(\sqrt{kT})$ [Dekel et al., 2012] for small enough batch sizes[4].

For assessing the communication cost of this operation, we use a simplified cost model that only counts the number of model vectors sent between the learners: independently of the exact communication infrastructure, the number of model messages asymptotically determines the true bit-based communication cost. Using a designated coordinator note as in Alg. 1, $\sigma$ can be applied to a configuration of the distributed prediction system simply by all nodes sending their current model to the coordinator, who in turn computes the mean model and sends it back to all the nodes. Hence, the **communication cost** of static averaging over $k$ nodes with batch size $b$ is $O(kT/b)$.

While this is less than the naive baseline by a factor of $b$, in many scenarios the achieved reduction might still be insufficient. In particular, for non-stationary settings the batch size has to be chosen small enough for the protocol to remain adaptive to changes in the environment so that the communication reduction effect can be marginal. A big weakness of the scheme is that it is oblivious to the actual model configuration observed, so that it also induces a lot of communication in situations where all models are approximately identical. In the

[4] Dekel et al. [2012] consider a slightly modified algorithm, which accumulates updates and then only applies them delayed at the end of a batch. However, the expected loss of eager updates (as used in Alg. 1) is bounded by the expected loss of delayed updates in the stationary setting (as used in Dekel et al. [2012]) as long as the updates reduce the distance to a loss minimizer on average (which is the case for sufficient regularization; see again Zhang [2004, Eq. 5]).

following section, we present a data-dependent dynamic averaging operator that can substantially reduce the communication cost while approximately retaining the performance of static averaging.

## 3 Dynamic Synchronization

In this section, we develop a dynamic protocol for synchronizations based on quantifying their effect. In order to assess the performance of this protocol from a learning perspective, we compare it to the static protocol as described in Alg. 1. After showing that this approach is sound from a learning perspective, we discuss how it can be implemented in a distributed prediction system in a communication-efficient way.

### 3.1 Partial Averaging

Intuitively, the communication for performing model averaging is not well invested in situations where all models are already approximately equal. A simple measure to quantify the effect of synchronizations is given by the **variance** of the current local model configuration space, i.e., $\delta(W) = \frac{1}{k}\sum_{l=1}^{k} \|\overline{W} - W_l\|^2$ . In the following definition we provide a relaxation of the static averaging operation that allows to omit synchronization in cases where the variance of a model configuration is low.

**Definition 2** *A* ***partial averaging operator*** *with positive variance threshold $\Delta \in \mathbb{R}$ and batch size $b \in \mathbb{N}$ is a synchronization operator $\sigma_\Delta$ such that $\sigma_\Delta(W_t) = W_t$ if $t \mod b \neq 0$ and otherwise: (i) $\overline{W_t} = \overline{\sigma_\Delta(W_t)}$, i.e., it leaves the mean model invariant, and (ii) $\delta(\sigma_\Delta(W)) \leq \Delta$, i.e., after its application the model variance is bounded by $\Delta$.*

An operator adhering to this definition does not generally put all nodes into sync (albeit the fact that we still refer to it as *synchronization* operator). In particular it allows to leave all models untouched as long as the variance remains below the threshold $\Delta$ or to only average a subset of models in order to satisfy the variance constraint. This is the basis for our dynamic averaging protocol. In the following, we analyze the impact on the learning performance of using partial averaging instead of static averaging. We start with showing that, given two model configurations $D$ and $S$, applying the partial averaging operator $\sigma_\Delta$ to $D$ and the static averaging operator $\sigma$ to $S$ increases their average squared pairwise model distances by at most $\Delta$.

**Lemma 3** *Let $D_t, S_t \in \mathbb{R}^{k\times n}$ be model configurations at time $t \in \mathbb{N}$. Then*

$$\frac{1}{k}\sum_{l=1}^{k} \|\sigma_\Delta(D_t)_l - \sigma(S_t)_l\|^2 \leq \frac{1}{k}\sum_{l=1}^{k} \|d_{t,l} - s_{t,l}\|^2 + \Delta \ .$$

*Proof.* We consider the case $t \mod b = 0$ (otherwise the claim follows immediately). Expressing the pairwise squared distances via the difference to $\overline{D_t}$ and using the definitions of $\sigma$ and $\sigma_\Delta$ we can bound

$$\begin{aligned}
&\frac{1}{k}\sum_{l=1}^{k}\|\sigma_\Delta(D_t)_l-\sigma(S_t)_l\|^2=\frac{1}{k}\sum_{l=1}^{k}\|\sigma_\Delta(D_t)_l-\overline{D_t}+\overline{D_t}-\overline{S_t}\|^2\\
=&\underbrace{\frac{1}{k}\sum_{l=1}^{k}\|\sigma_\Delta(S_t)_l-\overline{S_t}\|^2}_{\leq\Delta,\text{ by (ii) of Def. 2}}+2\langle\underbrace{\frac{1}{k}\sum_{l=1}^{k}\sigma_\Delta(D_t)_l-\overline{D_t}}_{=0,\text{ by (i) of Def. 2}},\overline{D_t}-\overline{S_t}\rangle+\|\overline{D_t}-\overline{S_t}\|^2\\
\leq&\Delta+\|\frac{1}{k}\sum_{l=1}^{k}(d_{t,l}-s_{t,l})\|^2=\Delta+\frac{1}{k}\sum_{l=1}^{k}\|d_{t,l}-s_{t,l}\|^2 \ .
\end{aligned}$$

□

In order to prove a regret bound of partial over static averaging it remains to show that this increase in distance cannot separate model configurations too far during the learning process. For this we show that $f$-propotional convex updates on the same training example reduce the distance between a pair of models proportional to their loss difference.

**Lemma 4** *Let $\varphi$ be an $f$-proportional convex update rule with constant $\gamma > 0$. Then for all models $d, s \in \mathbb{R}^n$ it holds that*

$$\|\varphi(d,x,y)-\varphi(s,x,y)\|^2\leq\|d-s\|^2-\gamma^2\left(f(d,x,y)-f(s,x,y)\right)^2 .$$

*Proof.* For $w \in \mathbb{R}^n$ we write $P_{x,y}(w) = P(w)$ for the projection of $w$ on $\Gamma_{x,y}$ and $w' = \varphi(w, x, y)$. Since $P(\cdot)$ is a projection on a convex set, it holds for all $v, w \in \mathbb{R}^n$ that

$$\|P(v)-P(w)\,\|^2\leq\|v-w\|^2-\|v-P(v)-w+P(w)\,\|^2 \tag{2}$$

(e.g., by lemma 3.1.4 in Nesterov [2003]). Also since $w' = \tau_{x,y}P(w)+(1-\tau_{x,y})w$ by (ii) of the definition of $f$-proportional convex updates, the idempotence of $P(\cdot)$ implies that $P(w) = P(w')$. Applying (2) to the models $d, s$ and to the updated models $d', s'$, respectively, and subtracting the two inequalities gives

$$0\leq\|d-s\|^2-\|d'-s'\|^2-\|d-P(d)-s+P(s)\,\|^2+\|d'-P(d)-s'+P(s)\,\|^2 \ .$$

By inserting $w' = w + \tau_{x,y}(P(w) - w)$ and using $\tau_{x,y} \in (0, 1]$ it follows that

$$\begin{aligned}
\|d'-s'\|^2\leq&\|d-s\|^2-\|(d-P(d))-s+P(s)\,\|^2\\
&+(1-\tau_{x,y})^2\|(d-P(d))-s+P(s)\,\|^2\\
\leq&\|d-s\|^2-\tau_{x,y}\left(\|d-P(d)\,\|-\|s-P(s)\,\|\right)^2\\
\leq&\|d-s\|^2-\gamma^2\left(f(d,x,y)-f(s,x,y)\right)^2
\end{aligned}\tag{3}$$

as required, where the last inequality follows from $\tau_{x,y} \in (0,1]$ and (i) of the definition of $f$-proportionality by noting that

$$\|w - P(w)\| = \frac{1}{\tau_{x,y}} \|w - (w + \tau_{x,y}(P(w) - w))\| = \frac{\|w - w'\|}{\tau_{x,y}} \geq \frac{\gamma}{\tau_{x,y}} f(w,x,y) \ .$$

□

From the two lemmas above we see that, while each synchronization increases the distance between the static and the dynamic model by at most $\Delta$, with each update step, the distance is decreased proportional to the loss difference. In the following theorem, we state that the average squared regret of using a partial averaging operator $\sigma_\Delta$ over a static averaging operator $\sigma$ with batch size $b$ is bounded by $\Delta/(b\gamma^2)$. We use the notion $\varphi(W_t) = (\varphi(w_{t,1}, x_{t,1}, y_{t,1}), \ldots, \varphi(w_{t,k}, x_{t,k}, y_{t,k}))$.

**Theorem 5** *Let $\mathbf{D} = D_0, \ldots, D_T$ and $\mathbf{S} = S_0, \ldots, S_T$ be two sequences of model configurations such that $D_0 = S_0$ and for $t = 1, \ldots, T$ defined by $D_{t+1} = \sigma_\Delta(\varphi(D_t))$ and $S_{t+1} = \sigma(\varphi(S_t))$, respectively (with an identical batch size $b \in N$). Then it holds that $R(\mathbf{D}, \mathbf{S}) \leq \Delta/(b\gamma^2)$.*

*Proof.* Let $\beta_t = 1$ if $t \mod b = 0$ and $\beta_t = 0$ otherwise. By combining Lm. 3 and 4 we have for all $t \in [T]$ that

$$\frac{1}{k}\sum_{l=1}^{k} \|d_{t+1,l} - s_{t+1,l}\|^2 \leq \frac{1}{k}\sum_{l=1}^{k} \|d_{t,l} - s_{t,l}\|^2 - \frac{\gamma^2}{k}\sum_{l=1}^{k}(f(d_{t,l}) - f(s_{t,l}))^2 + \beta_t \Delta \ .$$

Applying this inequality recursively for $t = 0, \ldots, T$, it follows that

$$\begin{aligned}\frac{1}{k}\sum_{l=1}^{k} \|d_{T+1,l} - s_{T+1,l}\|^2 \leq & \frac{1}{k}\sum_{l=1}^{k} \|d_{0,l} - s_{0,l}\|^2 + \left\lfloor \frac{T}{b} \right\rfloor \Delta \\ & - \sum_{t=1}^{T} \frac{\gamma^2}{k} \sum_{l=1}^{k} (f(d_{t,l}) - f(s_{t,l}))^2 .\end{aligned}$$

Using $D_0 = S_0$ we can conclude

$$\sum_{t=1}^{T} \frac{1}{k} \sum_{l=1}^{k} (f(d_{t,l}) - f(s_{t,l}))^2 \leq \frac{1}{\gamma^2}\left( \left\lfloor \frac{T}{b} \right\rfloor \Delta - \frac{1}{k}\sum_{l=1}^{k} \|d_{T+1,l} - s_{T+1,l}\|^2 \right) \leq \frac{T}{b\gamma^2}\Delta$$

which yields the result after dividing both sides by $T$. □

We remark that Thm. 5 implies that partial averaging retains the optimality of the static mini-batch algorithm of Dekel et al. [2012] for the case of stationary targets: by using a time-dependent variance threshold based on $\Delta_t \in O(1/\sqrt{t})$ the bound of $O(\sqrt{T})$ follows. From Thm. 5 it follows that if a shifting bound exists for the static protocol then this bound also applies to the dynamic protocol. Formally, suppose the shifting regret $R(\mathbf{S}, \mathbf{U})$ of using the static averaging operator is bounded by $c_1 \sum_{t=1}^{T} \sum_{l=1}^{k} \|u_{t,l} - u_{t-1,l}\|_2^2 + c_2$, for a reference sequence $\mathbf{U}$

and positive constants $c_1, c_2 \in \mathbb{R}_+$ (as, e.g., in [Herbster and Warmuth, 2001]). Then the shifting regret of using dynamic averaging is bounded by

$$R(\mathbf{D}, \mathbf{U}) \leq c_1 \sum_{t=1}^{T} \sum_{l=1}^{k} \|u_{t,l} - u_{t-1,l}\|_2^2 + c_2 + \frac{1}{\gamma^2} \Delta \ ,$$

where $\mathbf{D}$ denotes the sequence of model configurations produced by $\sigma_\Delta$. For the proof let furthermore $\mathbf{S}$ denote the sequence of model configurations produced by $\sigma$. With this we can directly derive the bound by using the definition of shifting regret, i.e.,

$$\begin{aligned}
R(\mathbf{D}, \mathbf{U}) &= \frac{1}{T} \sum_{t=1}^{T} \frac{1}{k} \sum_{l=1}^{k} (f(d_{t,l}) - f(u_{t,l}))^2 \\
&= \frac{1}{T} \sum_{t=1}^{T} \frac{1}{k} \sum_{l=1}^{k} ((f(d_{t,l}) - f(s_{t,l})) + (f(s_{t,l}) - f(u_{t,l})))^2 \\
&\overbrace{\leq}^{Thm.5} \frac{1}{\gamma^2} \Delta + \frac{1}{T} \sum_{t=1}^{T} \frac{1}{k} \sum_{l=1}^{k} (f(s_{t,l}) - f(u_{t,l}))^2 \\
&\leq \frac{1}{\gamma^2} \Delta + R(\mathbf{S}, \mathbf{U}) = \frac{1}{\gamma^2} \Delta + c_1 \sum_{t=1}^{T} \sum_{l=1}^{k} \|u_{t,l} - u_{t-1,l}\|_2^2 + c_2 \ .
\end{aligned}$$

Intuitively, this means that the dynamic protocol only adds a constant to any shifting bound of static averaging.

### 3.2 Communication-efficient Protocol

After seeing that partial averaging operators are sound from the learning perspective, we now turn to how they can be implemented in a communication-efficient way. Every distributed learning algorithm that implements a partial averaging operator has to implicitly control the variance of the model configuration. However, we cannot simply compute the variance by centralizing all local models, because this would incur just as much communication as static full synchronization. Our strategy to overcome this problem is to first decompose the global condition $\delta(W) \leq \Delta$ into a set of local conditions that can be monitored at their respective nodes without communication (see, e.g., Sharfman et al. [2007]). Secondly, we define a resolution protocol that transfers the system back into a valid state whenever one or more of these local conditions are violated. This includes carrying out a sufficient amount of synchronization to reduce the variance to be less or equal than $\Delta$.

For deriving local conditions we consider the domain of the variance function restricted to an individual model vector. Here, we identify a condition similar to a safe-zone (see Keren et al. [2012]) such that the global variance can not cross the $\Delta$-threshold as long as all local models satisfy that condition.[5]

[5] Note that a direct distribution of the threshold across the local nodes (as in, e.g., Keralapura et al. [2006]) is in-feasible, because the variance function is non-linear.

**Algorithm 2** Dynamic Synchronization Protocol

```
Initialization:
  local models $w_{1,1}, \dots, w_{1,k} \leftarrow (0, \dots, 0)$
  reference vector $r \leftarrow (0, ..., 0)$
  violation counter $v \leftarrow 0$
Round $t$ at node $l$:
  observe $x_{t,l}$ and provide service based on $p_{t,l}$
  observe $y_{t,l}$ and update $w_{t+1,l} \leftarrow \varphi(w_{t,l}, x_t, y_t)$
  if $t \mod b = 0$ and $\|w_{t,l} - r\|^2 > \Delta$ then
    send $w_{t,l}$ to coordinator (violation)
At coordinator on violation:
  let $B$ be the set of nodes with violation
  $v \leftarrow v + |B|$
  if $v = k$ then $B \leftarrow [k]$, $v \leftarrow 0$
  while $B \neq [k]$ and $\frac{1}{B} \sum_{l \in B} \|w_{t,l} - r\|^2 > \Delta$ do
    augment $B$ by augmentation strategy
    receive models from nodes added to $B$
  send model $\overline{\mathbf{w}} = \frac{1}{B} \sum_{l \in B} w_{t,l}$ to nodes in $B$
  if $B = [k]$ also set new reference vector $r \leftarrow \overline{\mathbf{w}}$
```

**Theorem 6** *Let $D_t = d_{t,1}, ..., d_{t,k} \in \mathbb{R}^n$ be the model configuration at time t and $r \in \mathbb{R}^n$ be some **reference vector**. If for all $l \in [k]$ the **local condition** $\|d_{t,l} - r\|^2 \leq \Delta$ holds, then the global variance is bounded by $\Delta$, i.e.,*

$$\frac{1}{k} \sum_{l=1}^{k} \|d_{t,l} - \overline{D}_t\|^2 \leq \Delta \ .$$

*Proof.* The theorem follows directly from the fact that the current average vector $\overline{D}_t$ minimizes the squared distances to all $d_{t,i}$, i.e.,

$$\frac{1}{k} \sum_{i=1}^{k} \|d_{t,l} - \overline{D}_t\|^2 \leq \frac{1}{k} \sum_{i=1}^{k} \|d_{t,l} - r\|^2 \leq \Delta$$

□

We now incorporate these local conditions into a distributed prediction algorithm. As a first step, we have to guarantee that at any time all nodes use the same reference vector $r$, for which a natural choice is the last average model that has been set to all local nodes. If the reference vector is known to all local learners a local learners $l$ can then monitor its local condition $\|d_{t,l} - r\|^2 \leq \Delta$ in a decentralized manner.

It remains to design a resolution protocol that specifies how to react when one or several of the local conditions are violated. A direct solution is to trigger a full synchronization in that case. This approach, however, does not scale well with a high number of nodes in cases where model updates have a non-zero

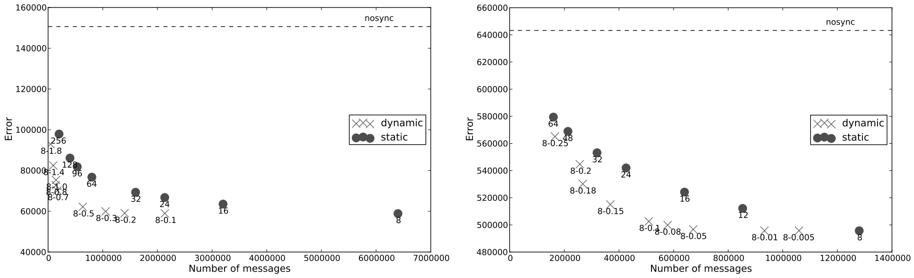

**Fig. 1.** Performance of static and dynamic model synchronization that track (left) a rapidly drifting disjunction over 100-dim. data with 512 nodes; and (right) a neural network with one hidden layer and 150 output vars. with 1024 nodes.

probability even in the asymptotic regime of the learning process. When, e.g., PAC models for the current target distribution are present at all local nodes, the probability of one local violation, albeit very low for an individual node, increases exponentially with the number of nodes. An alternative approach that can keep the amount of communication low relative to the number of nodes is to perform a local balancing procedure: on a violation, the respective node sends his model to a designated node we refer to as coordinator. The coordinator then tries to balance this violation by incrementally querying other nodes for their models. If the mean of all received models lies within the safe zone, it is transferred back as new model to all participating nodes, and the resolution is finished. If all nodes have been queried, the result is equal to a full synchronization and the reference vector can be updated. In both cases, the variance of the model configuration is bounded by $\Delta$ at the end of the balancing process, because all local conditions hold. Also, it is easy to check that this protocol leaves the global mean model unchanged. Hence, it is complying to Def. 2.

While balancing can achieve a high communication reduction over direct resolution particularly for a large number of nodes, it potentially degenerates in certain special situations. We can end up in a stable regime in which local violations are likely to be balanced by a subset of the nodes; however a full synchronization would strongly reduce the expected number of violations in future rounds. In other words: balancing can delay crucial reference point updates indefinitely. A simple hedging mechanism for online optimization can be employed in order to avoid this situation: we count the number of local violations using the current reference point and trigger a full synchronization whenever this number exceeds the total number of nodes. This concludes our dynamic protocol for distributed prediction. All components are summarized in Alg. 2

## 4 Empirical Evaluation

In this section we investigate the practical performance of the dynamic learning protocol for settings ranging from clean linearly separable data, over unseparable data with a reasonable linear approximation, up to real-world data without any

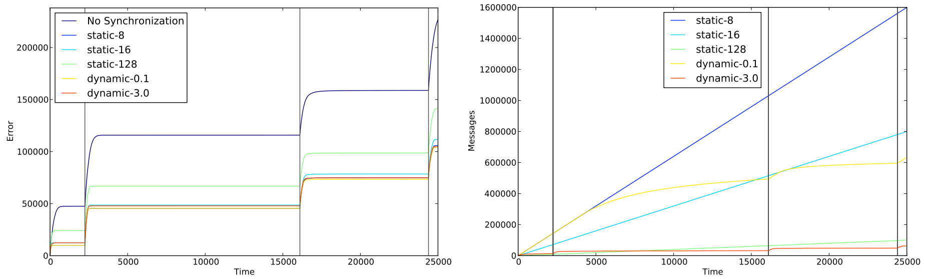

**Fig. 2.** Cumulative error (left) and communication (right) over time for tracking a rapidly drifting disjunction for different synchronization protocols; vertical lines depict drifts.

guarantee. Our main goal is to empirically confirm that the predictive gain of static full synchronizations (using a batch size of 8) over no synchronization can be approximately preserved for small enough thresholds, and to assess the amount of communication reduction achieved by these thresholds.

### 4.1 Linearly Separable Data

We start with the problem of tracking a rapidly drifting random disjunction. In this case the target distribution produces data that is episode-wise linearly separable. Hence, we can set up the individual learning processes so that they converge to a linear model with zero classification error within each episode. Formally, we identify a target disjunction with a binary vector $z \in \{0,1\}^n$. A data point $x \in X = \{0,1\}^n$ is labeled positively $y = 1$ if $\langle x, z \rangle \geq 1$ and otherwise receives a negative label $y = -1$. The target disjunction is drawn randomly at the beginning of the learning process and is randomly re-set after each round with a fixed drift probability of 0.0001. In order to have balanced classes, the disjunctions as well as the data points are generated such that each coordinate is set independently to 1 with probability $\sqrt{1 - 2^{-1/n}}$. We use the unregularized passive aggressive update rule with hinge loss.

In Fig. 1 (left) we present the result for dimensionality $n = 100$, with $k = 512$ nodes, processing $m = 12.8M$ data points through $T = 100000$ rounds. For divergence thresholds up to 0.3, dynamic synchronization can retain the error number of statically synchronizing every 8 rounds. At the same time the communication is reduced to 9.8% of the original number of messages. An approximately similar amount of communication reduction can also be achieved using static synchronization by increasing the batch size to 96. This approach, however, only retains 61.0% of the accuracy of statically synchronizing every 8 rounds.

Fig. 2 provides some insight into how the two evaluation metrics develop over time. Target drifts are marked with vertical lines that frame episodes of a stable target disjunction. At the beginning of each episode there is a relatively short phase in which additional errors are accumulated and the communicative protocols acquire an advantage over the baseline of never synchronizing. This is

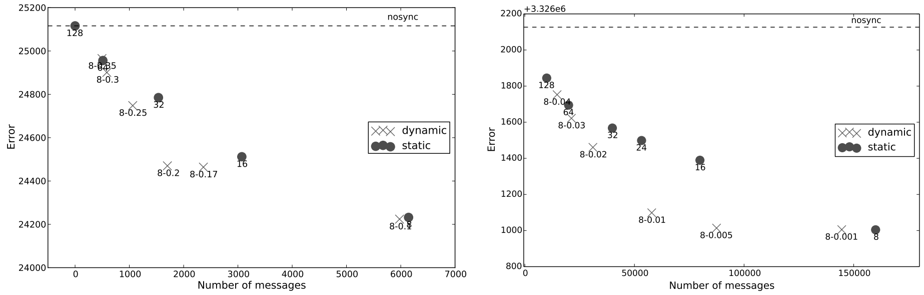

**Fig. 3.** Performance of static and dynamic synchronization with 256 nodes that predict (left) Twitter retweets over 1000 textual features and (right) stock prices based on 400 prices and sliding averages.

followed by a phase during which no additional error is made. Here, the communication curve of the dynamic protocols remain constant acquiring a gain over the static protocols in terms of communication.

### 4.2 Non-separable Data with Noise

We now turn to a harder experimental setting, in which the target distribution is given by a rapidly drifting two-layer neural network. For this target even the Bayes optimal classifier per episode has a non-zero error, and, in particular, the generated data is not linearly separable. Intuitively, it is harder in this setting to save communication, because a non-zero residual error can cause the linear models to periodically fluctuate around a local loss minimizer—resulting in crossings of the variance threshold even when the learning processes have reached their asymptotic regime. We choose the network structure and parameter ranges in a way that allow for a relatively good approximation by linear models (see Bshouty and Long [2012]). The process for generating a single labeled data point is as follows: First, the label $y \in Y = \{-1, 1\}$ is drawn uniformly from $Y$. Then, values are determined for hidden variables $H_i$ with $1 \leq i \leq \lceil \log n \rceil$ based on a Bernoulli distribution $P[H_i = \cdot \,|Y = y] = \mathrm{Ber}(p^h_{i,y})$. Finally, $x \in X = \{-1, 1\}^n$ is determined by drawing $x_i$ for $1 \leq i \leq n$ according to $P[X_i = x_i, |H_{p(i)} = h] = \mathrm{Ber}(p^o_{i,h})$ where $p(i)$ denotes the unique hidden layer parent of $x_i$. In order to ensure linear approximability, the parameters of the output layer are drawn such that $|p^o_{i,-1} - p^o_{i,1}| \geq 0.9$, i.e., their values have a high *relevance* in determining the hidden values. As in the disjunction case all parameters are re-set randomly after each round with a fixed drift probability (here, 0.01). For this non-separable setting we choose again to optimize the hinge loss, this time with regularized passive aggressive updates with $C = 10.0$ and a batch size of $b = 8$.

Fig. 1 (right) contains the results for dimensionality 150, with $k = 1024$ nodes, processing $m = 2.56M$ data points through $T = 10000$ rounds. For variance thresholds up to 0.08, dynamic synchronization can retain the error of the baseline. At the same time, the communication is reduced to 45% of the original number of messages. Moreover, even for thresholds up to 0.2, the dynamic protocol retains more than 90% of the accuracy of static synchronization with only 20% of its communication.

### 4.3 Real-world Data

We conclude our experimental section with tests on two real-world datasets containing stock prices and Twitter short messages, respectively.

The data from Twitter has been gathered via its streaming API (`https://dev.twitter.com/docs/streaming-apis`) during a period of 3 weeks (Sep 26 through Oct 15 2012). Inspired by the content recommendation task, we consider the problem of predicting whether a given tweet will be re-tweeted within one hour after its posting—for a number of times that lies below or above the median hourly re-tweet number of the specific Twitter user. The feature space are the top-1000 textual features (stemmed 1-gram, 2-gram) ranked by information gain, i.e., $X = \{0,1\}^{1000}$. Learning is performed with $C = 0.25$. The stock price data is gathered from Google Finance (`http://www.google.com/finance`) and contains the daily closing stock prices of the S&P100 stocks between 2004 and 2012. Inspired by algorithmic trading, we consider the problem of predicting tomorrow's closing price, i.e., $Y = \mathbb{R}$, of a single target stock based on all stock prices and their moving averages (11, 50, and 200 days) of today, i.e., $X = \mathbb{R}^{400}$. The target stock is switched with probability 0.001. Here, we use the epsilon insensitive loss, $\epsilon = 0.1$, and a regression parameter of $C = 1.0$ for regularized passive aggressive updates.

The results for 1.28M data points distributed to $k = 256$ nodes are presented in Fig. 3. Again, the gap between no synchronization and the baseline is well preserved by partial synchronizations. For Twitter (left), a threshold of 0.1 performs even better then the static baseline with less communication (0.97%). With a threshold of 0.2 the dynamic protocol still preserves 74% of predictive gain using only 27% communication. For the stock prices (right), a threshold of 0.005 preserves 99% of the predictive gain using 54% of the communication. The trade-off is even more beneficial for threshold 0.01 which preserves 92% of the gain using only 36% communication.

## 5 Conclusion

We presented a protocol for distributed online prediction that can save communication by dynamically omitting synchronizations in sufficiently stable phases of a modeling task, while at the same time being adaptive in phases of concept drifts. The protocol has a controlled predictive regret over its static counterpart and experiments show that it can indeed reduce the communication substantially—up to 90% in settings where the linear learning processes are suitable to model the data well and converge reasonably fast. Generally, the effectiveness of the approach appears to correspond to the effectivity of linear modeling with $f$-proportional convex update rules in the given setting.

For future research a theoretical characterization of this behavior is desirable. A practically even more important direction is to extend the approach to other model classes that can tackle a wider range of learning problems. In principle, the approach of controlling model variance remains applicable, as long as the variance is measured with respect to a distance function that induces a useful loss bound between two models. For probabilistic models this can for instance be the KL-divergence. However, more complex distance functions constitute more challenging distributed monitoring tasks, which currently are open problems.

## Bibliography

Jacob Abernethy, Alekh Agarwal, Peter L. Bartlett, and Alexander Rakhlin. A stochastic view of optimal regret through minimax duality. In *COLT 2009*, 2009.

Maria-Florina Balcan, Avrim Blum, Shai Fine, and Yishay Mansour. Distributed learning, communication complexity and privacy. *CoRR*, abs/1204.3514, 2012.

Amir Bar-Or, Ran Wolff, Assaf Schuster, and Daniel Keren. Decision tree induction in high dimensional, hierarchically distributed databases. In *SDM*, 2005.

Nader H. Bshouty and Philip M. Long. Linear classifiers are nearly optimal when hidden variables have diverse effects. *Machine Learning*, 86(2):209–231, 2012.

Nicolò Cesa-Bianchi and Gábor Lugosi. *Prediction, learning, and games.* Cambridge University Press, 2006. ISBN 978-0-521-84108-5.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585, 2006.

Hal Daumé III, Jeff M. Phillips, Avishek Saha, and Suresh Venkatasubramanian. Efficient protocols for distributed classification and optimization. *CoRR*, abs/1204.3523, 2012.

Ofer Dekel, Ran Gilad-Bachrach, Ohad Shamir, and Lin Xiao. Optimal distributed online prediction using mini-batches. *Journal of Machine Learning Research*, 13:165–202, 2012.

Mark Herbster and Manfred K Warmuth. Tracking the best linear predictor. *The Journal of Machine Learning Research*, 1:281–309, 2001.

Daniel Hsu, Nikos Karampatziakis, John Langford, and Alexander J. Smola. Parallel online learning. *CoRR*, abs/1103.4204, 2011.

Ram Keralapura, Graham Cormode, and Jeyashankher Ramamirtham. Communication-efficient distributed monitoring of thresholded counts. In *SIGMOD*, pages 289–300, 2006.

Daniel Keren, Izchak Sharfman, Assaf Schuster, and Avishay Livne. Shape sensitive geometric monitoring. *Knowledge and Data Engineering, IEEE Transactions on*, 24(8):1520–1535, 2012.

G. Mann, R. McDonald, M. Mohri, N. Silberman, and D. Walker. Efficient large-scale distributed training of conditional maximum entropy models. In *NIPS*, volume 22, pages 1231–1239, 2009.

Ryan T. McDonald, Keith Hall, and Gideon Mann. Distributed training strategies for the structured perceptron. In *HLT-NAACL*, pages 456–464, 2010.

Yurii Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*, volume 87. Kluwer Academic Publisher, 2003.

XuanLong Nguyen, Martin J Wainwright, and Michael I Jordan. Decentralized detection and classification using kernel methods. In *ICML*, page 80. ACM, 2004.

Jie Ouyang, Nilesh Patel, and Ishwar Sethi. Induction of multiclass multifeature split decision trees from distributed data. *Pattern Recognition*, 42(9):1786–1794, 2009.

Joel B Predd, SB Kulkarni, and H Vincent Poor. Distributed learning in wireless sensor networks. *Signal Processing Magazine, IEEE*, 23(4):56–69, 2006.

Izchak Sharfman, Assaf Schuster, and Daniel Keren. A geometric approach to monitoring threshold functions over distributed data streams. *ACM Trans. Database Syst.*, 32(4), 2007.

Jui-Pin Wang, Yu-Chen Lu, Mi-Yen Yeh, Shou-De Lin, and Phillip B Gibbons. Communication-efficient distributed multiple reference pattern matching for m2m systems. In *ICDM*. IEEE, 2013.

Lin Xiao. Dual averaging methods for regularized stochastic learning and online optimization. *The Journal of Machine Learning Research*, 11:2543–2596, 2010.

Tong Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *ICML 2004*, 2004.

Martin Zinkevich, Alex J. Smola, and John Langford. Slow learners are fast. In *NIPS 2009*, pages 2331–2339, 2009.

Martin Zinkevich, Markus Weimer, Alexander J. Smola, and Lihong Li. Parallelized stochastic gradient descent. In *NIPS 2010*, pages 2595–2603, 2010.