Communication-Efficient Distributed Online Prediction using Dynamic Model Synchronizations

[Extended Abstract]

Mario Boley and Michael Kamp
Fraunhofer IAIS & University Bonn
{mario.boley,michael.kamp}@iais.fraunhofer.de

Daniel Keren
Haifa University
dkeren@cs.haifa.ac.il

Assaf Schuster and Izchak Sharfman
Technion, Israel Institute of Technology
assaf@technion.ac.il & 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. Applications include social content recommendation, algorithmic trading, and other scenarios where a configuration of local prediction models of high-frequency streams is used to provide a real-time service. For stationary data, the proposed protocol 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. The protocol is based on controlling the divergence of the local models in a decentralized way. Its beneficial properties are also confirmed empirically.

1. INTRODUCTION

We consider online prediction problems where data points are observed at local nodes in a distributed environment and there is a trade-off between maximizing prediction accuracy and minimizing network communication. 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.] 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. In addition to the above, here the cost of communication can also be a loss of prediction quality itself when training examples have to be discarded due to network latency.

1.1 Setting and Related Work

Despite the communication costs it induces, decentralization is inevitable in many modern scale applications. Hence, recent articles [Balcan et al., 2012, Daumé III et al.] explicitly investigate the communication complexity of learning with decentralized data. They consider, however, the *offline* task of finding a good global model over the union of all data as a final computation result. The same applies to some work on parallel machine learning (e.g., Zinkevich et al. [2010], McDonald et al. [2010]) where data shards are distributed among several processors and then all computation is carried out independently in parallel except for one final model merging step. While these approaches avoid communication for performance reasons, they do not intend to optimize the predictive performance during the computation. In contrast, we are interested in the *online* in-place performance, i.e., for every data point performance is assessed locally when and where it is received or sampled.

To this end, research focused so far on specific environments with fixed communication constraints. Correspondingly, the learning strategies that are proposed and analyzed for these settings, do not aim to minimize communication beyond the level that is enforced by these constraints. Zinkevich et al. [2009] considers a shared-memory model, in which all local nodes can update a global model in a round-robin fashion as they process their training examples. Since this approach is problematic if there is a notable communication latency, strategies have been investigated [Mann et al., 2009, Dekel et al., 2012] that communicate only periodically after a statically fixed number of data points have been processed. 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 *mini-batches* of $O(\sqrt[3]{m})$ data points. Here, m denotes the total number of data points observed throughout the lifetime of the system. For large values of m, the effect of bounded latency values is asymptotically outgrown by the increasing mini-batch size.

While a fixed periodic communication schedule reduces the communication by some fixed amount, 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. This is further amplified because a large number of modeling

tasks are performed simultaneously sharing the same limited bandwidth. Moreover, distributed learning systems that are deployed for a long lifetime relative to their data throughput can experience periodical or singular target drifts (e.g., corresponding to micro-trends in social networks). In these settings, a static schedule is bound to either provide only little to no communication reduction or to insufficiently react to changing data distributions.

1.2 Contributions and Outline

In this work, we give the first distributed prediction protocol for linear models that, at the same time, aims to provide a high online in-place prediction performance and explicitly tries to minimize communication. In terms of predictive power, as shown Sec. 3.1, the protocol retains the asymptotic optimal regret of the distributed mini-batch algorithm of Dekel et al. [2012] for stationary data. In addition, it allows to reduce the communication among the local nodes substantially. This is achieved by a dynamic data dependent communication schedule, which, in contrast to previous algorithms, remains applicable when the data is non-stationary and shows rapid concept drifts. The main idea is to synchronize the local models to their mean model in order to reduce their variance, but to do so only in system states that show a high divergence among the models. This divergence, measured by the average model distance to the mean model, indicates the synchronizations that are most important in terms of their correcting effects on the predictions. In stable phases this allows communicative quiescence, while, in hard phases where variance reduction is crucial, the protocol will trigger a lot of model synchronizations. In order to efficiently implement this strategy one has to monitor the non-linear divergence function without communication overhead. We propose a solution to this problem that adapts recent ideas from distributed systems research based on local safe-zones in the function domain (Sec. 3.2). Experiments confirm the beneficial properties of the protocol (Sec. 4).

2. PRELIMINARIES

In this section we formally introduce the distributed online prediction task. As simple local learning tool we recall stochastic gradient descent for linear models. Finally, we review the state-of-the-art communication protocol as a departure point for developing a more communication-efficient solution in subsequent sections.

2.1 Distributed Online Prediction

Throughout this paper we consider a distributed online prediction system of k **local learners** that maintain individual **linear models** $w_{t,1}, \dots, w_{t,k} \in \mathbb{R}^n$ 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** $\mathcal{D}_t : X \times Y \to [0,1]$ 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. While we allow $\mathcal{D}_t$ to vary with time, we assume that it remains constant most of the time and only experiences a small number of rapid drifts. That is, there are **drift points** $0 = d_0 < d_1 < \dots < d_p = T$ such that for all $i \in [p]$ and $t, t' \in [T]$ with $d_{i-1} \leq t \leq t' < d_i$ it holds that $\mathcal{D}_t = \mathcal{D}_{t'}$. Hence, there are identically distributed **episodes** $E_i = \{d_i, \dots, d_{i+1} - 1\}$ between any two drift points. We assume that all learners sample from $\mathcal{D}$ 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. Generally, we assume that all training examples are bounded by a ball with **radius** R.

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** $\varphi : \mathbb{R}^n \times X \times Y \to \mathbb{R}^n$. Finally, 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 and that may take into account the information of all local learners simultaneously. The performance of such a distributed online prediction system is measured by two quantities: 1) the predictive performance $\sum_{t=1}^{T} \sum_{l=1}^{k} f(p_{t,l}, y_{t,l})$ measured by a **loss function** $f : \mathbb{R} \times Y \to \mathbb{R}_+$ that assigns positive penalties to prediction scores; and 2) the amount of **communication** within the system that is measured by the number of bits sent in-between learners to compute the sync operation. Next, specify possible choices for the update rule, the loss function, and the synchronization operator.

2.2 Losses and Gradient Descent

Generally, the communication protocol developed in this paper is applicable to a wide range of online update rules for linear models from, e.g., the passive aggressive rule [Crammer and Singer, 2001] to regularized dual averaging [Xiao, 2010]. However, the regret bound given in Theorem 2 assumes that the updates are **contractions**. That is, there is some constant $c < 1$ such that for all $w, w' \in \mathbb{R}^n$, and $x, y \in X \times Y$ it holds that $\|\varphi(w, x, y) - \varphi(w', x, y)\| \leq c\|w - w'\|$. For the sake of simplicity, in this paper, we focus on rules based on l_2-regularized stochastic gradient descent, for which this contraction property is readily available. We note that by considering *expected* contractions the result can be extended to rules that reduce on average the distance to a (regularized) loss minimizer.

Before we can define gradient descent updates, we have to introduce the underlying loss functions measuring predictive performance. Again for convenience, we restrict ourselves to functions that are differentiable, convex, and globally **Lipschitz continuous** in the prediction score, i.e., there is some constant L such that for all $p, p', y \in \mathbb{R}^{2n} \times Y$ it holds that $|f(p, y) - f(p', y)| \leq L|p - p'|$. While these assumptions can be relaxed by spending some technical effort, they already include loss functions for all standard predictions tasks such as the **logistic loss** $f_{\text{lg}}(p, y) = \ln(1 + \exp(-yp))$ for binary classification (case $Y = \{-1, 1\}$) or the **Huber loss** for regression (in the case $Y = \mathbb{R}$)

$$f_{\text{hu}}(p, y) = \begin{cases} \frac{1}{2}(p - y)^2 & , \text{ for } |p - y| \leq 1 \\ |p - y| - \frac{1}{2} & \end{cases} .$$

See, e.g., Zhang [2004] for further possible choices. In both of these cases the (best) Lipschitz constant is $L = 1$.

Algorithm 1 Static Synchronization Protocol

Initialization:
  local models $w_{1,1}, \ldots, w_{1,k} \leftarrow (0, \ldots, 0)$

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 $\{w_{t,l} \colon l \in [k]\}$
  send $w_{t,1}, \ldots, w_{t,k} \leftarrow \frac{1}{k} \sum_{l \in [k]} w_l$

With this we can define **stochastic gradient descent (SGD)** rules with l_2-regularization, i.e., rules of the form

$$\varphi(w, x, y) = w - \eta_t \nabla_w \left(\frac{\lambda}{2} \|w\|^2 + f(\langle w, x \rangle, y) \right)$$

where $\lambda \in \mathbb{R}_+$ is a strictly positive **regularization parameter** and $\eta_t \in \mathbb{R}_+$ are strictly positive **learning rates** for $t \in \mathbb{N}$. For stationary target distributions, one often chooses a decreasing learning rate such as $\eta_t = 1/\sqrt{t}$ in order to guarantee convergence of the learning process. For non-stationary targets this is infeasible, because for large t it would prevent sufficient model adaption to target changes. However, one can show [Zinkevich et al., 2010] that stochastic gradient descent is a contraction for sufficiently small constant learning rates. Namely, for $\eta \leq (RL + \lambda)^{-1}$ the updates do contract with constant $c = 1 - \eta\lambda$. This can be used to show that the stochastic learning process converges to a distribution centered close to a regularized loss minimizer even when the process is distributed among k nodes (see the analysis of Zinkevich et al. [2010]). This refers to the stochastic learning process defined by the mean of independent local models that result from SGD with iid samples from (episodes of) the target distribution. In this paper, the contraction property is used for the regret bound of Thm. 2.

2.3 Communication and Mini-batches

For every episode E_i, the predictive performance of a distributed prediction system lies between two baselines that correspond to the two extremes in terms of communication behavior—complete centralization and no communication. Let $T_i = |E_i|$ denote the length of episode E_i and by $R = \sum_{t \in E_i,\, l \in [k]} f(p_{t,l}, y_{t,l}) - f^*$ the regret with respect to the optimal expected loss $f^* = \operatorname{argmin}_{w \in \mathbb{R}^n} \mathbb{E}_{(x,y) \sim \mathcal{D}_i}[f(\langle w, x \rangle, y)]$. When all data points are centrally processed by one online learner, for long enough episodes one can achieve an expected regret of $O(\sqrt{kT_i})$ which is optimal (see Cesa-Bianchi and Lugosi [2006] and Abernethy et al. [2009]). In contrast, when the k nodes perform their learning processes in parallel without any communication this results in an expected regret of $O(k\sqrt{T_i})$, which is worse than the centralized performance by a factor of $\sqrt{k}$. Therefore, we are interested in algorithms that lie between these two extremes and that show a beneficial trade-off between predictive performance and the amount communication.

Mann et al. [2009] and Dekel et al. [2012] give algorithms where information between nodes is only exchanged every b rounds where $b \in \mathbb{N}$ is referred to as **batch size**. These algorithms can be written as static model synchronization protocol similar to Alg. 1. Here, after a batch of kb examples has been processed globally in the system, all local models are re-set to the **mean model** of the configuration $\mathbf{w}$ defined as $\overline{\mathbf{w}} = 1/k \sum_{l=1}^{k} w_l$. Formally, the synchronization operator that is implicitly employed in these algorithms is given by $\sigma(\mathbf{w}_t) = (\overline{\mathbf{w}}_t, \ldots, \overline{\mathbf{w}}_t)$. We refer to this operation as **full mean synchronization**. The choice of a (uniform) model mixture is often used for combining linear models that have been learned in parallel on independent training data (see Mann et al. [2009], 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). Dekel et al. [2012] shows that when the gradient variance is bounded then the optimal regret can be asymptotically retained by setting $b = O(\sqrt[3]{T_i})$ even if a constant number of examples have to be discarded during each synchronization due to network latency. Note that this reference considers a slightly modified algorithm based on delayed gradient descent, which only applies (accumulated) updates at synchronization points. However, the expected loss of eager updates (as used in Alg. 1) is bounded by the expected loss of delayed updates (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 sufficiently small learning rates and regularization parameters; see again Zhang [2004, Eq. 5]).

Closing this section, let us analyze the communication cost of this protocol.Using a designated coordinator note as in Alg. 1, σ can computed 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. For assessing the communication cost of this operation, we only count the number of model vectors sent between the learners. This is feasible because, independently of the exact communication infrastructure, the number of model messages asymptotically determines the true bit-based cost. Hence, asymptotically the **communication cost** of static model synchronization over k nodes with batch size b is $O(kT/b)$. Dekel et al. [2012] assumes that the data distribution is stationary over all rounds and b can therefore be set to $O(\sqrt[3]{T})$. This results in an automatic communication reduction that increases with a longer system lifetime. However, this strategy is not applicable when we want to stay adaptive towards changing data distributions. In this case, we have to set the batch size with respect to the expected episode length and not with respect to the overall system lifetime. This number can be much smaller than T resulting in batch sizes that are too small to meet our communication reduction goal. In the following section, we therefore design a synchronization protocol that can substantially reduce this cost based on a data-dependent dynamic schedule.

3. DYNAMIC SYNCHRONIZATION

The synchronization protocol of Alg. 1 is static because it synchronizes after a fixed number of rounds independently of the sampled data and its effect on the local models. Consequently, it incurs the communication cost of a full synchronization round even if the models are (almost) identical and thus only receive little to none correction. In this section, we develop a dynamic protocol for synchronizations based on quantifying their effect. After showing that this approach

is sound from a learning perspective, we discuss how it can be implemented in a communication-efficient way.

3.1 Partial Synchronizations

A simple measure to quantify the correcting effect of synchronizations is given by the average Euclidean distance between the current local models and the result model. We refer to this quantity as the **divergence** of a model configuration, denoted by $\delta(\cdot)$, i.e., $\delta(\mathbf{w}) = \frac{1}{k}\sum_{l=1}^{k} \|\overline{\mathbf{w}} - \mathbf{w}_l\|_2$. In the following definition we provide a relaxation of the full mean synchronization operation that introduces some leeway in terms of this divergence.

DEFINITION 1. *A **partial synchronization operator** with a positive divergence threshold $\Delta \in \mathbb{R}$ is an operator $\sigma_\Delta : \mathbb{R}^{k\times n} \to \mathbb{R}^{k\times n}$ that 1) leaves the mean model invariant and 2) after its application the model divergence is bounded by Δ. That is, for all model configurations $\mathbf{w} \in \mathbb{R}^{k\times n}$ it holds that $\overline{\mathbf{w}} = \overline{\sigma_\Delta \mathbf{w}}$ and $\delta(\sigma_\Delta \mathbf{w}) \leq \Delta$.*

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 divergence remains below the threshold Δ. The following theorem notes that partial synchronization has a controlled regret over full synchronization if the batch size is sufficiently large and the divergence threshold is set proportional to the Lipschitz constant L of the losses and the data radius R.

THEOREM 2. *Suppose the update rule φ is a contraction with constant c. Then, for batch sizes $b \geq \log_2^{-1} c^{-1}$ and divergence thresholds $\Delta \leq \epsilon/(2RL)$, the average regret of using a partial synchronization operator σ_Δ instead of σ is bounded by ϵ, i.e., for all rounds $t \in \mathbb{N}$ it holds that the average regret $1/k \sum_{l=1}^{k} |f(p^{\Delta}_{t,l}, y_{t,l}) - f(p_{t,l}, y_{t,l})|$ is bounded by ϵ where $p_{t,l}$ and $p^{\Delta}_{t,l}$ denote the prediction scores at learner l and time t resulting from σ and σ_Δ, respectively.*

We omit the proof here referring to the full version of this paper. While the contraction assumption is readily available for regularized SGD, as mentioned in Sec. 2, it can be relaxed: by requiring the updates to only contract on *expectation* it is possible to extend the theorem to unregularized SGD updates as well as to other rules. Moreover, we remark that Thm. 2 implies that partial synchronizations retain the optimality of the static mini-batch algorithm of Dekel et al. [2012] for the case of stationary targets: By using a time-dependent divergence threshold based on $\epsilon_t \in O(1/\sqrt{t})$ the bound of $O(\sqrt{T})$ follows.

3.2 Communication-efficient Protocol

After seeing that partial synchronization operators are sound from the learning perspective, we now turn to how they can be implemented in a communication-efficient way. Every distributed learning protocol that implements a partial synchronization operator has to implicitly control the divergence of the model configuration. However, we cannot simply compute the divergence 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 $\delta(\mathbf{w}) \leq \Delta$ 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

Algorithm 2 Dynamic Synchronization Protocol

Initialization:
  local models $w_{1,1}, \ldots, w_{1,k} \leftarrow (0, \ldots, 0)$
  reference point $r \leftarrow (0, \ldots, 0)$
  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** $\|r - w_{t,l}\| > \Delta/2$ **then**
    send $w_{t,l}$ to coordinator

At coordinator on violation:
  let B be set of nodes with violation
  $v \leftarrow v + |B|$
  if $v = k$ **then** $B \leftarrow [k]$, $v \leftarrow 0$
  while $B \neq [k]$ **and** $\|r - \frac{1}{B}\sum_{l\in B} w_l\| > \Delta$ **do**
    augment B by augmentation strategy
    receive models from nodes added to B
  send to nodes in B model $w = \frac{1}{B}\sum_{l\in B} w_l$
  if $B = [k]$ also set new reference model $r \leftarrow w$

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 divergence to be less or equal than Δ.

For deriving local conditions we consider the domain of the divergence function restricted to an individual model vector. Here, we identify a *safe-zone* S (see Keren et al. [2012]) such that the global divergence can not cross the Δ-threshold as long as all local models remain in S.[1] The following statement, which we give again without proof, provides a valid spherical safe zone S_r that is centered around some global reference point r.

THEOREM 3. *Let $r \in \mathbb{R}^d$ be some reference point. If for all nodes $l \in \{1, \ldots, k\}$ it holds that $\|r - w_l\| \leq \Delta/2$ then we have for the model divergence that $\delta(\mathbf{w}) \leq \Delta$.*

We now incorporate these local conditions into a distributed prediction protocol. As a first step, we have to guarantee that at all times all nodes use the same reference point. For a prediction t, let us denote by t' the last time prior to t when a full model synchronization was performed (resp. $t' = 0$ in case no full synchronization has happened until round t). The mean model $\overline{\mathbf{w}}_{t'}$ is known to all local learners. We use this model as the reference model and set $r = \overline{\mathbf{w}}_{t'}$. A local learners l can then monitor their local condition $\|r - w_l\| \leq \Delta/2$ 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 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

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

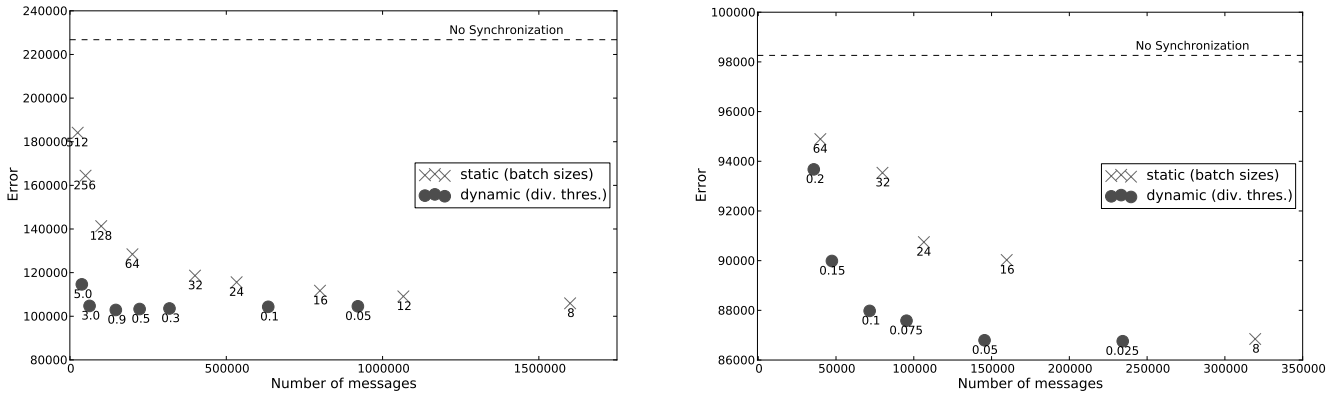

Figure 1: Performance of static and dynamic model synchronization that track (left) a rapidly drifting disjunction over 100-dimensional data with 512 nodes; and (right) a neural network with one hidden layer and 150 output variables. with 1024 nodes.

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 note 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 point can be updated. In both cases, the divergence of the model configuration is bounded by Δ at the end of the balancing process, because all local conditions hold. Also this protocol leaves the global mean model unchanged. Hence, it is complying to Def. 1.

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 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 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 two controlled settings: one with linearly separable data and one with unseparable data. 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.

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 0.0002. 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 $\sqrt{1 - 2^{-1/n}}$. As loss function for the stochastic gradient descent we use the logistic loss. Corresponding to our setting of noise-free linearly separable data, we choose the regularization parameter $\lambda = 0$ and the learning rate $\eta = 1$.

In Fig. 1 (left) we present the result for dimensionality $n = 100$, with $k = 512$ nodes, processing $m = 12.8M$ data points through $T = 25000$ rounds. For divergence thresholds up to 3.0, dynamic synchronization can retain the error number of statically synchronizing every 8 rounds. At the same time the communication is reduced to 3.9% 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 128. This approach, however, only retains 51.5% of the error reduction over no communication. Analyzing the development of the evaluation metrics over time reveals: 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 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.

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 divergence 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, 0.005). For this non-separable setting we choose again to optimize the logistic loss, this time with parameters $\lambda = 0.5$ and $\eta = 0.05$ respectively. Also, in order to increase the stability of the learning process, we apply averaged updates over mini-batches of size 8.

Figure 1 (right) contains the results for dimensionality 150, with $k = 1024$ nodes, processing $m = 2.56M$ data points through $T = 2500$ rounds. For divergence thresholds up to 0.05, dynamic synchronization can retain the error of the baseline. At the same time the communication is reduced to 46% of the original number of messages.

5. CONCLUSION

We presented a protocol for distributed online prediction that aims to dynamically save on network communications in sufficiently easy phases of the modeling task. The protocol has a controlled predictive regret over its static counterpart and experiments show that it can indeed reduce the communication substantially—up to 95% in settings where the linear learning processes are suitable to model the data well and converge reasonably fast. Generally, the effectivity of the approach appears to correspond to the effectivity of linear modeling by SGD 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 divergence remains applicable, as long as the divergence 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.

References

Jacob Abernethy, Alekh Agarwal, Peter L. Bartlett, and Alexander Rakhlin. A stochastic view of optimal regret through minimax duality. In *COLT 2009 - The 22nd Conference on Learning Theory*, 2009.

Maria-Florina Balcan, Avrim Blum, Shai Fine, and Yishay Mansour. Distributed learning, communication complexity and privacy. *Journal of Machine Learning Research - Proceedings Track*, 23:26.1–26.22, 2012.

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 and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2001.

Hal Daumé III, Jeff M. Phillips, Avishek Saha, and Suresh Venkatasubramanian. Efficient protocols for distributed classification and optimization. In *ALT 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.

Daniel Hsu, Nikos Karampatziakis, John Langford, and Alexander J. Smola. Parallel online learning. In *Scaling up machine learning: Parallel and distributed approaches*. Cambridge University Press.

Ram Keralapura, Graham Cormode, and Jeyashankher Ramamirtham. Communication-efficient distributed monitoring of thresholded counts. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data (SIGMOD 2006)*, 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 *Advances in Neural Information Processing Systems (NIPS 2009)*, volume 22, pages 1231–1239, 2009.

Ryan T. McDonald, Keith Hall, and Gideon Mann. Distributed training strategies for the structured perceptron. In *Human Language Technologies: Conf. of the North American Chapter of the Association of Computational Linguistics, Proceedings (HLT-NAACL)*, pages 456–464, 2010.

XuanLong Nguyen, Martin J Wainwright, and Michael I Jordan. Decentralized detection and classification using kernel methods. In *Proceedings of the twenty-first international conference on Machine learning*, page 80. ACM, 2004.

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.

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 *Proceedings of the 21st int. conf. on Machine learning (ICML 2004)*, 2004.

Martin Zinkevich, Alex J. Smola, and John Langford. Slow learners are fast. In *Proc. of 23rd Annual Conference on Neural Information Processing Systems (NIPS 2009)*, pages 2331–2339, 2009.

Martin Zinkevich, Markus Weimer, Alexander J. Smola, and Lihong Li. Parallelized stochastic gradient descent. In *Proc. of 24th Annual Conference on Neural Information Processing Systems (NIPS 2010)*, pages 2595–2603, 2010.