Variance Reduced Training with Stratified Sampling for Forecasting Models

Yucheng Lu 1 2, Youngsuk Park 2, Lifan Chen 2, Yuyang Wang 2, Christopher De Sa 1, Dean Foster 3 4

Abstract

In large-scale time series forecasting, one often encounters the situation where the temporal patterns of time series, while drifting over time, differ from one another in the same dataset. In this paper, we provably show under such heterogeneity, training a forecasting model with commonly used stochastic optimizers (e.g., SGD) potentially suffers large variance on gradient estimation, and thus incurs long-time training. We show that this issue can be efficiently alleviated via stratification, which allows the optimizer to sample from pre-grouped time series strata. For better trading-off gradient variance and computation complexity, we further propose SCott (Stochastic Stratified Control Variate Gradient Descent), a variance reduced SGD-style optimizer that utilizes stratified sampling via control variate. In theory, we provide the convergence guarantee of SCott on smooth non-convex objectives. Empirically, we evaluate SCott and other baseline optimizers on both synthetic and real-world time series forecasting problems, and demonstrate SCott converges faster with respect to both iterations and wall clock time.

1. Introduction

Large-scale time series forecasting is prevalent in many real-world applications, such as traffic flow prediction (Vlahogianni et al., 2014), stock price monitoring (Box et al., 2011), weather forecasting (Xu et al., 2019), etc. Traditional forecasting models such as SSM (Durbin & Koopman, 2012), ARIMA (Zhang, 2003), ETS (De Livera et al., 2011) and Gaussian Processes (Brahim-Bellhouari & Bermak, 2004) are the folklore methods for modeling the dynamics of a single time series. Recently, deep forecasting models (Faloutsos et al., 2019) that leverage deep learning techniques have been proven to be particularly well-suited at modeling over an entire collection of time series (Rangapuram et al., 2018; Wang et al., 2019; Salinas et al., 2020). In such setting, multiple time series are jointly learned, which enables forecasting over a large scope.

In practice, a time series dataset can be heterogeneous with respect to a single forecasting model (Lee et al., 2018). The heterogeneity here specifically indicates the underlying distribution of interests may vary across different time series instances due to local effects (Wang et al., 2019; Sen et al., 2019); or is correlated to time in each time series individually – a phenomenon we refer to as concept drift (Gama et al., 2014). In light of this, a seemingly plausible solution is to maintain multiple forecasters. However, in most applications training a single model is inevitable since deploying multiple models incurs storage overhead and sometimes generalizes worse (Montero-Manso & Hynman, 2020; Oreshkin et al., 2019; Gasthaus et al., 2019). As a first investigation in this paper, we provably show the time series heterogeneity can induce arbitrarily large gradient estimation variance in many optimizers, including SGD (Bottou, 2010), Adam (Kingma & Ba, 2014), AdaGrad (Ward et al., 2019), etc.

Extensive study has been conducted on reducing gradient estimation variance in stochastic optimization such as using mini-batching (Gower et al., 2019), control variate (Johnson & Zhang, 2013) and importance sampling (Csiba & Richtárik, 2018). These methods are mostly motivated by optimization theory and do not consider time series heterogeneity at a finer-grained level. In this paper, we take a different perspective: observing that the distribution of interests in time series is usually recurring over time horizon or is correlated over instances (Liao, 2005; Aghabozorgi et al., 2015; Maharaj et al., 2019), we argue gradient variance induced by time series heterogeneity can be mitigated via stratification. Specifically, the intuition is that if we can somehow stratify the time series into multiple strata where each stratum contains homogeneous series, then the variance on the gradient estimation can be provably reduced via
weighted sampling over all the strata. Our paper concludes
with a specific algorithm named SCott (Stochastic Stratified
Control Variate Gradient Descent), an SGD-style optimizer
that utilizes this stratified sampling strategy with control
variate to balance variance-complexity trade off.

Our contributions can be summarized as follows:
1. We show in theory that even on a simple AutoRegressive
(AR) forecasting model, the gradient estimation variance
can be arbitrarily large and slows down training.
2. We conduct a comprehensive study on temporal time
series, and show how stratification over timestamps al-
 lows us to obtain homogeneous strata with negligible
computation overhead.
3. We propose a variance-reduced optimizer SCott based on
stratified sampling, and prove its convergence on smooth
non-convex objectives.
4. We empirically evaluate SCott on both synthetic and real-
world forecasting tasks. We show SCott is able to speed
up SGD, Adam and Adagrad without compromising the
computation overhead.

Notations. Throughout this paper, we use \( y_j \) to denote
the \( j \)-th coordinate of a vector \( y \). We use \( y_{i,a;b} \) to denote
\([y_{i,a}, y_{i,a+1}, \ldots, y_{i,b-1}, y_{i,b}]\). For two variables \( g_1 \) and
\( g_2 \), \( g_1 = \Omega(g_2) \) means there exists a numerical constant \( c \)
such that \( g_1 \geq cg_2 \). We use \(|S|\) to denote the cardinality
of a set \( S \). We use \( \mathbb{E}[X] \) and \( \text{Var}[X] \) to denote the expectation
and variance of a random variable \( X \), given their existence.

2. Related Work

Learning from Heterogeneous Time Series. Heterogeneity
in time series forecasting has been investigated in prior arts.
Previous works mostly address this from two aspects:
(1) Maintaining multiple models. A representative work
is ESRNN (Smyl, 2020), the M4 forecasting competition
winner that proposes using ensemble of experts (Hewa-
malage et al., 2021); (2) Modifying the model architecture
that characterizes the time series prior to training (Ban-
dara et al., 2020b; Chen et al., 2020; Bandara et al., 2020a;
Lara-Benitez et al., 2021). In this paper, we investigate an
orthogonal direction on variance-reduced gradients. Our
method does not require multiple models or modification
of model architectures.

Sampling in Stochastic Optimization. In the domain of
stochastic optimization, uniform sampling is the folklore
sampler used in many first-order optimizers, e.g. SGD
(Zhang, 2004). Based on that, Nagaraj et al. (2019)
discusses uniform sampler without replacement, Gao et al.
(2015) proposes adopting an active weighted sampler for
training and Park & Ryu (2020) discusses sampling with
 cyclic scanning. Several fairness-aware samplers are also
investigated in (Iosifidis et al., 2019; Wang et al., 2020;
Holstein et al., 2019). In other works, London (2017); Aber-
nethy et al. (2020) study the effect of adaptive sampling
on model generalization. A series of works extensively
discuss the importance sampling based on gradient norm
(Alain et al., 2015), gradient bound (Lee et al., 2019), loss
(Loshchilov & Hutter, 2015), etc, is able to accelerate train-
ing. Perhaps the closest works to this paper are (Zhao &
Zhang, 2014; Zhang et al., 2017), which propose using strati-
fied sampling for more diverse gradients. This, however,
is notably different from our investigation as we do not
use stratified sampling for mini-batching, and we focus on
efficient stratification on time series data.

Stratification in Machine Learning. Stratification is a
powerful technique for machine learning. For instance,
application-driven works like Liberty et al. (2016) proposes
using stratified sampling to solve a specialized regression
problem in databases whereas Yu et al. (2019) discusses
stratification in weakly supervised learning. In terms of
variance-reduced training, most of the previous works ex-
clusively focus on using stratification for diversifying mini-
batches. Concretely, with the basic proposition of strati-
ed mini-batching from (Zhao & Zhang, 2014), subsequent
works like Zhang et al. (2017) extends that to a sampling
framework; Liu et al. (2020) proposes using adaptive strata;
and Fu & Zhang (2017) discusses transferring stratification
framework to Bayesian learning.

3. Preliminaries

In this section we introduce the formulation of training
forecasting models and stochastic gradient optimizers.

Problem Statement. As in other machine learning tasks,
training forecasting models is often formulated into the
Empirical Risk Minimization (ERM) framework. Given \( N \)
different time series: \( \{ z_{i,t} \}_{t=1}^{N} \) where \( z_{i,t} \) denotes the value
of \( i \)-th time series at time \( t \), let \( x_{i,t} \) denote the (potentially)
available features of \( i \)-th time series at time \( t \), we aim to
train a forecasting model \( F \) with parameters \( \theta \) (Table 1).
The training is then formulated by connecting \( F \) with a loss
function \( \mathcal{L} \) to be minimized. For instance, given the notation
in Table 1, a deterministic model over loss function \( \mathcal{L} \) at
prediction time \( t_0 \) can be expressed as

\[
f_{i,t_0}(\theta) = \mathcal{L}(z_{i,t_0 + 1:t_0 + \tau_p}, \hat{z}_{i,t_0 + 1:t_0 + \tau_p}),
\]

where \( \hat{z}_{i,t_0 + 1:t_0 + \tau_p} = F(z_{i,t_0 - \tau_c:1:t_0}, x_{i,t:1}; \theta) \) and
\( f_{i,t_0}(\theta) : \mathbb{R}^d \to \mathbb{R} \) is the loss incurred on the \( i \)-th time se-
ries at time \( t_0 \). Popular options for loss functions \( \mathcal{L} \) include
Mean Square Error (MSE) Loss, Quantile Loss, Negative
Log Likelihood, KL Divergence, etc (Gneiting & Katzfuss,
2014). The training is then formulated as an optimization
Table 1. Quantity of interests to approximate in different forecasting types. Inside the table, $F$ denotes the model where it takes context and features as input, and then make predictions via model parameters $\theta$, $\tau_c$ and $\tau_p$ denote the context length and prediction length, $t_0$ is referred to as prediction time by convention.

<table>
<thead>
<tr>
<th>Forecasting Model Type</th>
<th>Mappings/Distributions to Approximate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deterministic</td>
<td>$\hat{z}<em>{i,t_0+1:t_0+\tau_p} = F(z</em>{i,t_0+1:t_0+\tau_p}; \theta)$</td>
</tr>
<tr>
<td>Probabilistic</td>
<td>$\mathbb{P}(z_{i,t_0+1:t_0+\tau_p}</td>
</tr>
</tbody>
</table>

We start from studying the variance of Stochastic Gradient (SG): $\text{Var}[\nabla f_\xi(\theta)]$. For any iterative stochastic gradient optimizer $A$, let $\theta^{(0)}$ and $\theta^{(t)}$ denote its initial parameters and its output model parameters after $t$ iterations respectively. Let $\xi^{(t)}$ denote the mini-batch sampled in its $t$-th iteration. We start from a mild assumption on $A$.

**Assumption 1.** If stochastic optimizer $A$ satisfies $[\nabla f_{\xi(k)}(\theta^{(k)})]_j = 0$ for every $t > 0$ and $k = 1, \ldots, t$, then $[\theta^{(t)}]_j = [\theta^{(0)}]_j$ holds for any index of parameter $1 \leq j \leq p$.

Assumption 1 is often referred to as "zero-respecting" in optimization theory (Carmon et al., 2019) and widely covers many popular optimizers (e.g. SGD, Adam, Adagrad, RMSProp, Momentum SGD, etc) under arbitrary hyperparameter settings. This states that the SG optimizer $A$ will not modify a certain parameter of the model unless a gradient updates it in the training. With Assumption 1, we obtain the following theorem.

**Theorem 1.** For any AR($p$) model ($p \geq 1, p \in \mathbb{N}$) defined in Equation (3), there exists a time series dataset $D$ with $\max_{i,t} |z_{i,t}| = \delta$, such that for any stochastic gradient optimizer $A$ with any $\theta^{(0)} \in \mathbb{R}^p$ and hyperparameters, for all $0 < \epsilon < \frac{\delta^2 p - \frac{5}{8}}{2}$, $A$ needs to compute at least

$$T = \Omega \left( \text{Var} \left[ \nabla f_{\xi^{(0)}}(\theta^{(0)}) \right] \right)$$

(5)

number of stochastic gradients to find a $\hat{\theta} \in \mathbb{R}^p$ achieving $\mathbb{E}[\|\nabla f(\hat{\theta})\|] \leq \epsilon$. Furthermore,

$$\text{Var} \left[ \nabla f_{\xi^{(0)}}(\theta^{(0)}) \right] = \Omega \left( \delta^4 p \right).$$

(6)

Theorem 1 provides important insights in two aspects. Specifically, Equation (5) shows if we wish to find a target model with small error, then the least number of stochastic gradients we need is lower bounded by the complexity of variance on the SG. And this conclusion holds for arbitrary hyperparameter scheduling (even with very small learning rate in gradient descent type optimizers). On the other hand, Equation (6) reveals that the variance of SG can be arbitrarily large in theory, even with advanced transforma-
tion/preprocessing on the dataset such as magnitude scaler\(^2\) (Salinas et al., 2020). As the magnitude of the dataset, or the order of AR model increases, the variance on the SG can increase to infinity in theory.

5. Training with Stratified Sampling

Our motivation study in the previous section reveals that to provably reduce the gradient estimation variance, optimizers beyond Assumption 1 should be considered. In this section we first illustrate comprehensively a notion of low-cost stratification over time series, and how it leads to stratified gradients. Then we introduce the stratified sampling, which provably reduces gradient variance induced from the entire heterogeneous time series instances to the inter-strata homogeneous ones. We conclude this section with a proposition of an algorithm SCott, which modifies an existing optimizer with stratified sampling while trading-off the variance and complexity.

5.1. Stratification over Time Series

**Toy examples.** While we defer the discussion on obtaining variance-reduced gradients, we start from a toy example demonstrating simple stratification over timestamps on a stationary time series instance leads to strata of the gradients. The toy example is shown in Figure 1 with a simple yet effective stratification policy: hashing the timestamp based on the temporal interval. The intuition behind this toy example is fairly straightforward: note that in Table 1, the gradients induced on a certain time series segment only relates to the input features/observations, output observations and model parameters. That means, with the identical parameters, close input and output leads to close gradients in the sense of Euclidean distance. Building on top of this insight, if the distribution of interest is recurring over the time horizon, we can simply stratify all the time series segments based on timestamps with negligible cost. Additionally, we provide another example in Section B.1 that provably illustrates the variance reduced effect through stratified sampling over timestamps.

**Generalized stratification.** Given the toy example, we proceed to discuss performing similar low-cost stratification over time series in general cases that leads to adequate clustering on the gradients. A natural extension is to consider the long- and short-term temporal patterns (Lai et al., 2018) and perform recursive stratification. For instance, if a time series instance is recurring both in terms of months and seasons, we can generalize the timestamp stratification from Figure 1 and perform two dimensional hashing on (months, seasons) tuples. Note that such stratification induces much smaller overhead than clustering on high-dimensional features, since we are performing hierarchical hashing and are able to determine the stratum for each time series within logarithm time complexity. As will be shown in the experimental section, this extended policy of stratification suffice in many settings.

On the other hand, however, finer-grained stratification can be adopted. A de facto method is to utilize features from the dataset or extracted from the time series (e.g., density, spectrum, etc), and run a clustering algorithms such as K-Means based on that feature space (Bandara et al., 2020b). Note that based on our intuition, the main objective here is to identify time series with similar input/output as homogeneous instances while leaving the others as heterogeneous ones. In light of this, we do not need accurate stratification in high-dimensional feature space as the previous works did (Aghabozorgi et al., 2015).

5.2. Stratified Sampling

Stratification outputs several strata, we next discuss how to perform low-variance gradient estimation from these strata. Without the loss of generality, suppose the time series dataset is stratified into \( B \in \mathbb{N}^+ \) strata, i.e., \( D = D_1 \cup \cdots \cup D_B \), such that each training example \((i, t)\) (recall Equation (1)) belongs to one unique stratum. Using the standard definition of stratified sampling, we obtain a new estimator as

\[
g(\theta) = \sum_{i=1}^{B} \frac{|D_i| \nabla f_{\xi_i}(\theta)}{|D|}, \tag{7}
\]

where \( \xi_i \) is a set of size \( b \) and each element in \( \xi_i \) is in the form of a tuple \((j, t)\) where \((j, t) \sim \text{Uniform}[D_j]\). Comparing Equation (7) with (2) we can see the stratified sampling essentially accumulates the examples from different strata and perform a weighted average. With simple derivation, the property of stratified sampling is summarized in the following lemma

**Lemma 1.** Stratified sampler at any point \( \theta \in \mathbb{R}^d \) satisfies

\[
\mathbb{E}[g(\theta)] = \nabla f(\theta), \quad \text{Var}[g(\theta)] = \sum_{i=1}^{B} \frac{|D_i|^2 \text{Var}[\nabla f_{\xi_i}(\theta)]}{|D|^2}.
\]

Lemma 1 reveals stratified sampling ensures the unbiased estimation of true gradient \( \nabla f \), and the variance on such sampler only depends on the variance of stochastic gradient sampled within each stratum instead of the entire dataset \( \text{Var}[\nabla f_{\xi}(\theta)] \). In other words, stratified sampling does not suffer additional noise even if the distribution among strata are significantly, or even adversarially different.
Algorithm 1 SCott (Stochastic Stratified Control Variate Gradient Descent)

Require: Total number of iterations \(T\), learning rate \(\{\alpha_i\}_{i\in[T]}\), initialized \(\theta^{(0,0)}\), strata \(\{D_i\}_{i\in[B]}\).

1: for \(t = 0, 1, \cdots, T - 1\) do
2:   Sample a \(\xi^{(t)}\) from stratum \(i\) and perform Stratified Sampling (with \(w_i = |D_i|/|D|\)):
3:      \[ g^{(t,0)} = \sum_{i=1}^{B} w_i \nabla f^{(t)}(\theta^{(t,0)}) \] (8)
4:   for \(k = 0, 1, \cdots, K - 1\) do
5:     Compute the update \(v^{(t,k)}\) as
6:     \[ \nabla f^{(t,k)}(\theta^{(t,k)}) - \nabla f^{(t,k)}(\theta^{(t,0)}) + g^{(t,0)} \] (9)
7:   Update the parameters as
8:      \[ \theta^{(t,k+1)} = \theta^{(t,k)} - \alpha_k v^{(t,k)} \] (10)
9: end for
10: end for
11: return Sample \(\theta^{(T)}\) from \(\{\theta^{(t,0)}\}_{t=0}^{T-1}\) with \(\mathbb{P}(\theta^{(T)} = \theta^{(t,0)}) \propto \alpha_t B\)

5.3. SCott: Trading-off Variance and Complexity

Despite stratified sampling mitigating the gradient variance, naively utilizing such sampler in an optimizer is suboptimal since a single sampling requires computation over \(O(B \cdot M)\) gradients comparing to the \(O(M)\) complexity as shown in Equation (2). To address this, we propose a control variate based design on top of stratified sampling. Our intuition is that by periodically performing a stratified sampling and computing some snapshot gradients over the training trajectory, we can use those gradients as estimation anchors to reduce variance while allowing the optimizers to adopt flexible mini-batch sizes in the effective iterations. In other words, we seek to achieve an intermediate solution between the plain stratified sampling and stochastic optimizers, so that we can benefit from both worlds.

The formal description of such algorithm, which we refer to as SCott, is shown in Algorithm 1. Note that SCott has separate outer and inner iteration loops. A stratified sampling is only performed in each outer loop in Equation (8) and the output of stratified sampling \(\theta^{(0,i)}\) is then used as control variate\(^3\) in inner loop as in Equation (9).

### Table 2. Different stratification policies map SCott to algorithms

<table>
<thead>
<tr>
<th>Stratification Policy</th>
<th>Complexity</th>
<th>Equivalence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arbitrary</td>
<td>Theorem 2</td>
<td>SCott</td>
</tr>
<tr>
<td>Random Hashing</td>
<td>(O(\Delta L/|D|)^{3/2} \epsilon^{-2}))</td>
<td>SCSG</td>
</tr>
<tr>
<td>Finest-Grained</td>
<td>(O(\Delta L/|D|^{3/2} \epsilon^{-2}))</td>
<td>SVRG</td>
</tr>
</tbody>
</table>

6. Convergence Analysis

In this section, we derive the convergence rate of SCott. We first start from several assumptions.

**Assumption 2.** The loss on each single training example \(f_{i,t}\), \(\forall i\), is \(L\)-smooth: for some constant \(L > 0\),

\[ \|\nabla f_{i,t}(\theta_1) - \nabla f_{i,t}(\theta_2)\| \leq L\|\theta_1 - \theta_2\|, \forall \theta_1, \theta_2 \in \mathbb{R}^d. \]

Assumption 2 is a standard assumption in optimization theory. Note that smooth function is not necessarily convex, which implies our theory works with non-convex models.

\(^3\)Refer to (Nelson, 1990) for principles on the control variate.
e.g. deep neural networks with Sigmoid activations. We also make the assumption on the sampling variance as follows.

**Assumption 3.** For all \( \theta \in \mathbb{R}^d, i \in \{1, \ldots, B\} \) and \( t = 0, \ldots, T - 1 \) in Equation (8), there exists a constant \( \sigma_i^2 \) s.t.

\[
\text{Var}_{\xi \sim \mathcal{D}}[\nabla f_{\xi}(\theta)] \leq \sigma_i^2.
\]

The constant \( \sigma_i^2 \) in Assumption 3 denotes the upper bound on the gradient variance when uniform sampling is performed inside the \( i \)-th stratum. For the convenience of later discussion, we further denote \( \sigma^2 \) as the upper bound on the variance when uniform sampling over the entire dataset: \( \text{Var}_{\xi \sim \mathcal{D}}[\nabla f_{\xi}(\theta)] \leq \sigma^2 \). Without the loss of generality, we let \( M = 1 \) in our theory. Based on the two assumptions, the convergence rate of Algorithm 1 is shown in the following theorem:

**Theorem 2.** Denote \( \Delta = f(0) - \inf_{\theta} f(\theta) \) and \( w_i = |D_i|/|D| \). For any \( \epsilon > 0 \), if we stratify the dataset \( D \) into \( \{D_i\}_{i=1}^{B} \) such that \( \sum_{i=1}^{B} w_i^2 \sigma_i^2 = O(\epsilon^2) \), and let inner loop iterations \( K_i \sim \text{Geo}(B/(B+1))^4 \), Algorithm 1 needs to compute at most

\[
T = O \left( \frac{\Delta L_B \sum_{i=1}^{B} w_i^2 \sigma_i^2}{\epsilon^2} + \frac{\Delta L |D|^2}{\epsilon^2} \right)
\]

number of stochastic gradients to ensure \( \mathbb{E} \left\| \nabla f(\tilde{\theta}^{(T)}) \right\| \leq \epsilon \), where \( \mathbb{I}\{\cdot\} \) denotes the Indicator function.

If we deliberately let all the strata maintain the same size, the convergence rate can be simplified as follows.

**Corollary 1.** Following Theorem 2, if all the strata are the same size, i.e., \( |D_i| = |D_j| > 1, \forall i, j \), Algorithm 1 needs to compute at most

\[
T = O \left( \frac{\Delta L_B \sum_{i=1}^{B} \sigma_i^2}{\epsilon^2} \right)
\]  \hspace{1cm} (11)

number of stochastic gradients to ensure output \( \tilde{\theta}^{(T)} \) fulfills \( \mathbb{E} \left\| \nabla f(\tilde{\theta}^{(T)}) \right\| \leq \epsilon \).

**Remark 2: Reduced Variance Dependency.** Note that \( \sum_{i=1}^{B} w_i^2 \sigma_i^2 = O(\epsilon^2) \) can always be fulfilled since we can at least select \( B = |D| \) and obtain \( \sigma_i^2 = 0 \), as in that case every stratum only contains one sample. If this precondition is somehow violated, it may only guarantee suboptimality in theory, converging to a noisy ball with \( \sum_{i=1}^{B} w_i^2 \sigma_i^2 \). However, comparing to other stochastic control variate type optimizers, including (Li & Li, 2018) and (Babanezhad et al., 2015), where noise ball is in the order of \( O(\sigma^2) \), SCott is able to reduce the dependency only on the inner stratum variance, i.e., from \( \sigma^2 \) to \( B \sum_{i=1}^{B} w_i^2 \sigma_i^2 \) (and \( B^{-1} \sum_{i=1}^{B} \sigma_i^2 \) with Corollary 1).

**Remark 3: Understanding the Selection of \( K_i \).** The number of inner loops per outer loop (\( K_i \)), i.e., the frequency of performing a stratified sampling is a crucial design choice. Theorem 2 show that a Geometric distributed selection helps with the convergence, which is a technique used in other analysis of control variate type algorithms (Li & Li, 2018; Horváth et al., 2020). In practice, we can optimize such selection via an additional hyperparameter: in the supplementary material, we discuss using \( \|v^{(t,k)}\|^2 \leq \gamma \|v^{(t,0)}\|^2 \) as an additional criterion to terminate the inner loop for some hyperparameter \( \gamma \).

**Remark 4: Improved Complexity Compared to Stochastic Optimizers.** Carmon et al. (2019) shows the theoretical lower bound on the complexity of stochastic optimizers is \( \Omega(\Delta L \sigma^2 \epsilon^{-4}) \). Comparing Corollary 1 with this bound, we can observe a complexity improvement of at least \( O(\epsilon^{-\frac{1}{2}}) \) from SCott compared to stochastic optimizers. On the other hand, as also shown in Table 2, the convergence rate of SCott improves upon SCSG and stochastic optimizers in the sense that its upper bound depends only on the inner strata variance.

### 7. Experiment

In this section we empirically evaluate our algorithms and investigate how SCott improves optimizers in practice. We implement SCott in PytorchTS, a time series forecasting library (Rasul et al., 2020). All the tasks run on a local machine configured with a 2.6GHz Inter (R) Xeon(R) CPU, 8GB memory and a NVIDIA GTX 1080 GPU.

In Section 7.1 and Section 7.2, we focus on comparing SCott with SGD and SCSG in different settings. These optimizers all have SGD-style update formulas, which helps us better understand the effect of variance reduction. Additionally, in Section 7.3, we modify the main step of SCott and let it follow the update rule of Adam and Adagrad. We rerun the experiments from previous sections on the two SCott variants.

**Models and Loss functions.** In this section, we focus on
Table 3. Performance with different algorithms given the same time budget for training. We compute the loss over the entire training and test dataset. We assign time budget of 0.5 and 3 hours for each setting, respectively, and present the mean and standard deviation among 5 different runs. All the optimizers are fine tuned in each task.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Optimizer</th>
<th>Exchange Rate</th>
<th>Traffic</th>
<th>Electricity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Training</td>
<td>Test</td>
<td>Training</td>
</tr>
<tr>
<td>MLP</td>
<td>SGD</td>
<td>-1.825 ± 0.013</td>
<td>-1.715 ± 0.017</td>
<td>-2.387 ± 0.015</td>
</tr>
<tr>
<td></td>
<td>SCSG</td>
<td>-2.037 ± 0.009</td>
<td>-1.732 ± 0.019</td>
<td>-2.612 ± 0.013</td>
</tr>
<tr>
<td></td>
<td>SCott</td>
<td>-2.145 ± 0.008</td>
<td>-1.685 ± 0.009</td>
<td>-2.867 ± 0.019</td>
</tr>
<tr>
<td></td>
<td>Adam</td>
<td>-2.762 ± 0.009</td>
<td>-2.945 ± 0.012</td>
<td>-2.597 ± 0.021</td>
</tr>
<tr>
<td></td>
<td>S-Adam</td>
<td>-3.917 ± 0.009</td>
<td>-3.032 ± 0.006</td>
<td>-3.038 ± 0.011</td>
</tr>
<tr>
<td></td>
<td>Adagrad</td>
<td>-3.488 ± 0.011</td>
<td>-3.218 ± 0.011</td>
<td>-2.692 ± 0.012</td>
</tr>
<tr>
<td></td>
<td>S-Adagrad</td>
<td>-3.886 ± 0.007</td>
<td>-3.239 ± 0.012</td>
<td>-2.864 ± 0.013</td>
</tr>
<tr>
<td>N-BEATS</td>
<td>SGD</td>
<td>1.224 ± 0.018</td>
<td>1.346 ± 0.021</td>
<td>2.798 ± 0.008</td>
</tr>
<tr>
<td></td>
<td>SCSG</td>
<td>1.034 ± 0.016</td>
<td>1.182 ± 0.019</td>
<td>2.024 ± 0.012</td>
</tr>
<tr>
<td></td>
<td>SCott</td>
<td>1.077 ± 0.022</td>
<td>1.222 ± 0.012</td>
<td>1.898 ± 0.013</td>
</tr>
<tr>
<td></td>
<td>Adam</td>
<td>0.695 ± 0.021</td>
<td>0.773 ± 0.018</td>
<td>1.013 ± 0.012</td>
</tr>
<tr>
<td></td>
<td>S-Adam</td>
<td>0.514 ± 0.012</td>
<td>0.593 ± 0.021</td>
<td>0.809 ± 0.021</td>
</tr>
<tr>
<td></td>
<td>Adagrad</td>
<td>0.764 ± 0.022</td>
<td>0.806 ± 0.012</td>
<td>2.068 ± 0.012</td>
</tr>
<tr>
<td></td>
<td>S-Adagrad</td>
<td>0.563 ± 0.013</td>
<td>0.692 ± 0.009</td>
<td>1.486 ± 0.014</td>
</tr>
</tbody>
</table>

7.1. Warm-up: Synthetic Dataset

We start from training VAR and LSTM on a synthetic dataset, where we know the ground-truth for stratification. We generate the synthetic dataset by repeatedly transforming a linear curve based on simple functions such as \( \sin \) and \( \text{polynomial} \). For brevity, the details for generating the dataset is discussed in the supplementary material. We plot the results of SGD and SCott from Figure 2(a) to 2(d). We show SGD and SCott with the fine-tuned learning rate while for SGD we show the top three curves with different learning rates. In Figure 2(a) and Figure 2(c). For SGD, if the learning rate is large, then the convergence curve is noisy and unstable. To ensure stable convergence, SGD needs to adopt small learning rate, and that results in more iterations. SCSG, on the other hand, slightly improve over SGD, while is still noisy in later iterations. By comparison, SCott contains less variance noise, and it allows us to use larger learning rate while keeping a stable convergence.

7.2. Real World Applications

We proceed to discuss the performance on real-world applications. In this experiment, we train the FeedForward Network (MLP) and N-BEATS. We use three public benchmark datasets: Traffic, Exchange-Rate and Electricity (Lai et al., 2018), where details are shown as below:

- **Traffic**: A collection of hourly data from the California Department of Transportation. The data describes the road occupancy rates (between 0 and 1) measured by different sensors on San Francisco Bay area free ways. For this dataset, we set \( \tau_c = 3 \) days (72 hours) and \( \tau_p = 1 \) day (24 hours).
- **Exchange-Rate**: the collection of the daily exchange rates of eight foreign countries including Australia, British, Canada, Switzerland, China, Japan, New Zealand and Singapore ranging from 1990 to 2016. For this dataset, we set \( \tau_c = 8 \) days and \( \tau_p = 1 \) day.
- **Electricity**: The electricity consumption in kWh was recorded hourly from 2012 to 2014, for \( n = 321 \) clients.

We converted the data to reflect hourly consumption. For this dataset, we set \( \tau_c = 3 \) days (72 hours) and \( \tau_p = 1 \) day (24 hours).

**Stratification.** As mentioned in Section 5, we adopt a simple timestamp-based stratification policy for all the datasets. For Traffic dataset: we stratify all the time series segments only based on its weekday and season, i.e., two time series segments are in the same stratum if and only if their weekday and season are the same. This results in total 49 subgroups, we repeat this process on Electricity dataset and
it results in 70 strata; For Exchange-Rate dataset, we first evenly divide the whole time series into 6 range based on the time stamps. And then within each range we further group the time series segments based on the time series instance, this results in total 32 strata.

Results Analysis. We first see from Table 3 that given same time budget, SCott achieves smaller loss on both training and test set. We further plot the training curves in Figure 2(e) to 2(f) and Figure 2(i) to 2(j). We can observe the results are mostly aligned with our results on synthetic dataset: with stratified sampling, SCott is able to adopt large learning rate which allows it to converge faster compared to SGD and SCSG. Additionally, we verify in Figure 2(g) and 2(k) the benefits of SCott does not compromise the validation error of the model. Finally, we plot in Figure 2(h) and 2(l) that the stratified sampling does induce smaller variance compared to uniform sampling, even with the simple stratification policy we adopt.

7.3. Variants of SCott
So far, we focus on comparing SCott with SGD and SCSG. Moreover, we investigate how SCott can be applicable to en-
hance other types of optimizers, Adam and Adagrad. To do so, we incorporate the main step of SCott in Equation (10) into the update rule of Adam and Adagrad, which we refer to as S-Adam and S-Adagrad. We rerun all the experiments on the real-world dataset. We first see from Table 3 S-Adam/S-Adagrad is able to achieve smaller loss given same time. Then we further plot the training curves in Figure 3. We find SCott is able to improve both Adam and Adagrad by a certain margin. As shown in Table 3, these SCott variants improve upon their non-SCott baselines without compromising the test loss. Additionally, comparing SGD-type with Adam- and Adagrad-type optimizers, SCott sometimes can outperform Adam and Adagrad (such as in MLP on traffic dataset). On the other hand, we find S-Adam and S-Adagrad consistently outperform SGD-type optimizers as well as Adam and Adagrad.

8. Conclusions

In this paper, we show that heterogeneity in large scale time series data is detrimental to the convergence of the stochastic optimizers. To address the challenge, we introduce SCott, a variance reduced optimizer that speeds up the training of forecasting models based on stratified time series data. A novel convergence analysis is provided for SCott, which by varying the stratification conditions, recovers the well-known results in stochastic optimization. Empirically, we show SCott converges faster compared to plain stochastic optimizer, with respect to both iterations and time on both synthetic and real-world dataset. We leave the future works of investigating the effect of stratification on SCott variants, applying SCott tasks beyond forecasting, and developing practical stepsize selection (Park et al., 2020; Yu et al., 2020).

Acknowledgement

The authors would like to thank Valentin Flunkert, Konstantinos Benidis, Tim Januschowski, Syama Sundar Rangapuram, Lorenzo Stella, Laurent Callot and anonymous reviewers from ICML 2021 for providing valuable feedbacks on the earlier versions of this paper.

References


Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., and Bouchachia, A. A survey on concept drift adaptation. ACM computing surveys (CSUR), 46(4):1–37, 2014.


Variance Reduced Training with Stratified Sampling for Forecasting Models


