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 (**S**tochastic Stratified **Co**ntrol Variate Gradien**t** Descen**t**), 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-Belhouari & 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 & Hyndman, 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

This work was done during Yucheng Lu's internship at Amazon. Lifan Chen participated in this work while he was with Amazon. [1]Department of Computer Science, Cornell University, Ithaca, NY, USA. [2]Amazon Web Services (AWS) AI Labs, Palo Alto, CA, USA. [3]Amazon Research, New York, NY, USA. [4]University of Pennsylvania, Philadelphia, PA, USA. Correspondence to: Yucheng Lu <yl2967@cornell.edu>.

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021.

weighted sampling over all the strata. Our paper concludes with a specific algorithm named SCott (**S**tochastic Stratified **Co**ntrol Variate Gradien**t** Descen**t**), 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 allows 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 generalization of forecasting models.

Notations. Throughout this paper, we use $\boldsymbol{y}_j$ to denote the j-th coordinate of a vector $\boldsymbol{y}$. We use $\boldsymbol{y}_{i,a:b}$ to denote $[\boldsymbol{y}_{i,a}, \boldsymbol{y}_{i,a+1}, \cdots, \boldsymbol{y}_{i,b-1}, \boldsymbol{y}_{i,b}]$. 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 $|\mathcal{S}|$ to denote the cardinality of a set $\mathcal{S}$. We use $\mathbb{E}[X]$ and $\mathrm{Var}[X]$ 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 (Hewamalage et al., 2021); (2) Modifying the model architecture that characterizes the time series prior to training (Bandara 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); Abernethy 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 training. Perhaps the closest works to this paper are (Zhao & Zhang, 2014; Zhang et al., 2017), which propose using stratified 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 exclusively focus on using stratification for diversifying mini-batches. Concretely, with the basic proposition of stratified 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 trasferring 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: $\{\boldsymbol{z}_i\}_{i=1}^{N}$ where $\boldsymbol{z}_{i,t}$ denotes the value of i-th time series at time t, let $\boldsymbol{x}_{i,t}$ denote the (potentially) available features of i-th time series at time t, we aim to train a forecasting model F with parameters $\boldsymbol{\theta}$ (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

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

where $\hat{\boldsymbol{z}}_{i,t_0+1:t_0+\tau_p} = F(\boldsymbol{z}_{i,t_0-\tau_c+1:t_0}, \boldsymbol{x}_{i,1:T_i}; \boldsymbol{\theta})$ and $f_{i,t_0}(\boldsymbol{\theta}) : \mathbb{R}^d \to \mathbb{R}$ is the loss incurred on the i-th time series 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 $\boldsymbol{\theta}$. τ_c and τ_p denote the context length and prediction length, t_0 is referred to as prediction time by convention.

Forecasting Model Type	Mappings/Distributions to Approximate
Deterministic	$\hat{\boldsymbol{z}}_{i,t_0+1:t_0+\tau_p} = F(\boldsymbol{z}_{i,t_0-\tau_c+1:t_0}, \boldsymbol{x}_{i,1:T_i}; \boldsymbol{\theta})$
Probabilistic	$\hat{\mathbb{P}}(\boldsymbol{z}_{i,t_0+1:t_0+\tau_p} \vert \boldsymbol{z}_{i,t_0-\tau_c+1:t_0}, \boldsymbol{x}_{i,1:T_i}; \boldsymbol{\theta}) = F(\boldsymbol{z}_{i,t_0-\tau_c+1:t_0}, \boldsymbol{x}_{i,1:T_i}; \boldsymbol{\theta})$

problem as

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}\in\mathbb{R}^d} \left[f(\boldsymbol{\theta}) = \frac{1}{|\mathcal{D}|} \sum_{i=1}^{N} \sum_{t=1}^{T_i} f_{i,t}(\boldsymbol{\theta}) \right], \tag{1}$$

where T_i denotes the maximum prediction time in the i-th time series. We denote $\mathcal{D}$ as the set of all the training examples indexed by $(i,t), \forall 1 \le i \le N, \forall 1 \le t \le T_i$. A time series segment with certain length is then used as a single training example[1].

Stochastic Gradient Optimizers. A stochastic gradient optimizer refers to an iterative algorithm that repeatedly updates model parameters with a uniformly sampled mini-batch of training examples. Concretely, this stochastic gradient with mini-batch size M is computed as follows:

$$\nabla f_\xi(\boldsymbol{\theta}) = \frac{1}{M} \sum\nolimits_{(i,t)\in\xi} \nabla f_{i,t}(\boldsymbol{\theta}), \tag{2}$$

where $|\xi| = M$ and each element in ξ is in the form of a tuple (i,t) where $i \sim \text{Uniform}[1,N]$ and $t \sim \text{Uniform}[1,T_i]$. Popular optimizers includes SGD, Adam, Adagrad, etc.

4. Motivation

Based on the preliminaries, in this section we discuss our motivation that gradient variance in training forecasting models can be arbitrarily large and slow down training. Our motivation study is done with showing a complexity lower bound on the classic AutoRegressive (AR) model.

Motivation example: AutoRegression with MSE Loss. Throughout this section, we consider p-th order AutoRegressive, or AR(p) model, which can be expressed as follows:

$$\hat{\boldsymbol{z}}_{i,t} = \epsilon_t + \sum\nolimits_{j=1}^{p} \boldsymbol{\theta}_{t-j} \boldsymbol{z}_{i,t-j}, \tag{3}$$

where ϵ_t is a Gaussian noise term at time t. We further consider adopting the Mean Squared Error (MSE) loss function, which is

$$f_{i,t}(\boldsymbol{\theta}) = \left(\hat{\boldsymbol{z}}_{i,t} - \boldsymbol{z}_{i,t}\right)^2. \tag{4}$$

[1] In practice, this slice is of length that relates to the context and prediction length as shown in Table 1. A commonly used method to obtain all such segments is to use a sliding window strategy.

We start from studying the variance of Stochastic Gradient (SG): Var $[\nabla f_\xi(\boldsymbol{\theta})]$. For any iterative stochastic gradient optimizer $\mathcal{A}$, let $\boldsymbol{\theta}^{(0)}$ and $\boldsymbol{\theta}^{(t)}$ 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 $\mathcal{A}$.

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

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

$$T = \Omega\left(\mathit{Var}\left[\nabla f_{\xi^{(0)}}(\boldsymbol{\theta}^{(0)})\right]\right) \tag{5}$$

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

$$\mathit{Var}\left[\nabla f_{\xi^{(0)}}(\boldsymbol{\theta}^{(0)})\right] = \Omega\left(\delta^4 p\right). \tag{6}$$

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 transformation/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.

[2] With proper scaling, the magnitude δ can become a limited number, such as 1, making δ^4 bounded by a constant. However, the lower bound is still proportional to p, which can be arbitrarily large.

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., $\mathcal{D} = \mathcal{D}_1 \cup \cdots \cup \mathcal{D}_B$, 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

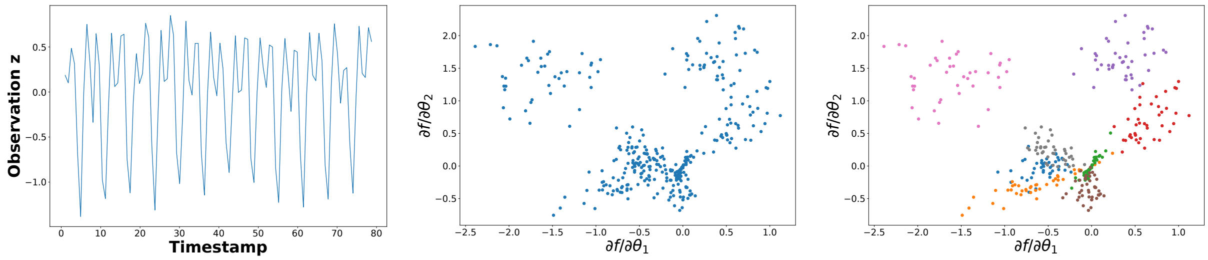
$$\boldsymbol{g}(\boldsymbol{\theta}) = \sum_{i=1}^{B} \frac{|\mathcal{D}_i| \nabla f_{\xi_i}(\boldsymbol{\theta})}{|\mathcal{D}|}, \tag{7}$$

where ξ_i is a set of size b and each element in ξ_i is in the form of a tuple (j, t) where $(j, t) \sim$ Uniform $[\mathcal{D}_i]$. 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 $\boldsymbol{\theta} \in \mathbb{R}^d$ satisfies*

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

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



(a) The Original Time Series (b) Gradient Distribution (c) Simple Stratification over Timestamps

Figure 1. A toy example illustrating simple stratification over the gradients via grouping the timestamps associated with each time series segment, based on estimated temporal interval in a recurring series. (Left) A time series follows the dynamics of $\sin(t) + \cos(2t) + 0.2\varepsilon_t$ with respect to time t where ε_t is a Gaussian noise. (Middle) The gradient distribution computed on an AR(2) model. (Right) The stratified gradients where the time series segment at time t_0 is grouped into the $(t_0 \bmod 8)$-th stratum with 8 strata in total.

Algorithm 1 SCott (**S**tochastic Stratified **Co**ntrol Variate Gradien**t** Descen**t**)

Require: Total number of iterations T, learning rate $\{\alpha_t\}_{1\leq t\leq T}$, initialized $\boldsymbol{\theta}^{(0,0)}$, strata $\{\mathcal{D}_i\}_{1\leq i\leq B}$.

for $t = 0, 1, \cdots, T-1$ **do**

  Sample a $\xi_i^{(t)}$ from stratum i and perform Stratified Sampling (with $w_i = |\mathcal{D}_i|/|\mathcal{D}|$):

$$\boldsymbol{g}^{(t,0)} = \sum\nolimits_{i=1}^{B} w_i \nabla f_{\xi_i^{(t)}}(\boldsymbol{\theta}^{(t,0)}) \tag{8}$$

  for $k = 0, 1, \cdots, K_t - 1$ **do**

    Sample $\xi^{(t,k)}$ from $\mathcal{D}$.

    Compute the update $\boldsymbol{v}^{(t,k)}$ as

$$\nabla f_{\xi^{(t,k)}}(\boldsymbol{\theta}^{(t,k)}) - \nabla f_{\xi^{(t,k)}}(\boldsymbol{\theta}^{(t,0)}) + \boldsymbol{g}^{(t,0)} \tag{9}$$

    Update the parameters as

$$\boldsymbol{\theta}^{(t,k+1)} = \boldsymbol{\theta}^{(t,k)} - \alpha_t \boldsymbol{v}^{(t,k)} \tag{10}$$

  end for

  Set $\boldsymbol{\theta}^{(t+1,0)} = \boldsymbol{\theta}^{(t,k_t)}$.

end for

return Sample $\hat{\boldsymbol{\theta}}^{(T)}$ from $\{\boldsymbol{\theta}^{(t,0)}\}_{t=0}^{T-1}$ with $\mathbb{P}(\hat{\boldsymbol{\theta}}^{(T)} = \boldsymbol{\theta}^{(t,0)}) \propto \alpha_t B$

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.

Table 2. Different stratification policies map SCott to algorithms SCSG (Li & Li, 2018) and SVRG (Johnson & Zhang, 2013) as illustrated in Section 6. Notations are defined in Section 6.

Stratification Policy	Complexity	Equivalence
Arbitrary	Theorem 2	SCott
Random Hashing	$O\left(\Delta L\sigma^{\frac{4}{3}}\epsilon^{-\frac{10}{3}}\right)$	SCSG
Finest-Grained	$O\left(\Delta L\lvert\mathcal{D}\rvert^{\frac{2}{3}}\epsilon^{-2}\right)$	SVRG

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

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 $f_{i,t}, \forall i, t$ is L-smooth:for some constant $L > 0$,*

$$\|\nabla f_{i,t}(\boldsymbol{\theta}_1) - \nabla f_{i,t}(\boldsymbol{\theta}_2)\| \leq L\|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|, \forall \boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \in \mathbb{R}^d.$$

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 $\boldsymbol{\theta} \in \mathbb{R}^d$, $i \in \{1, \cdots, B\}$ and $t = 0, \cdots, T-1$ in Equation (8), there exists a constant σ_i^2 s.t.*

$$Var_{\xi_i \sim \mathcal{D}_i}[\nabla f_{\xi_i}(\boldsymbol{\theta})] \leq \sigma_i^2.$$

The constant σ_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 σ^2 as the upper bound on the variance when uniform sampling over the entire dataset*: $\mathrm{Var}_{\xi \sim \mathcal{D}}[\nabla f_\xi(\boldsymbol{\theta})] \leq \sigma^2$. 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 $\Delta = f(\mathbf{0}) - \inf_{\boldsymbol{\theta}} f(\boldsymbol{\theta})$ and $w_i = |\mathcal{D}_i|/|\mathcal{D}|$. For any $\epsilon > 0$, if we stratify the dataset $\mathcal{D}$ into $\{\mathcal{D}_i\}_{i=1}^B$ such that $\sum_{i=1}^B w_i^2 \sigma_i^2 = O\left(\epsilon^2\right)$, and let inner loop iterations $K_t \sim Geo(B/(B+1))$[4], Algorithm 1 needs to compute at most*

$$T = O\left(\frac{\Delta L \left(B \sum_{i=1}^B w_i^2 \sigma_i^2\right)^{\frac{2}{3}}}{\epsilon^{\frac{10}{3}}} + \frac{\Delta L |\mathcal{D}|^{\frac{2}{3}} \mathbb{I}\{B = |\mathcal{D}|\}}{\epsilon^2}\right)$$

number of stochastic gradients to ensure $\mathbb{E}\left\|\nabla f\left(\hat{\boldsymbol{\theta}}^{(T)}\right)\right\| \leq \epsilon$, 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., $|\mathcal{D}_i| = |\mathcal{D}_j| > 1, \forall i, j$, Algorithm 1 needs to compute at most*

$$T = O\left(\frac{\Delta L \left(B^{-1} \sum_{i=1}^B \sigma_i^2\right)^{\frac{2}{3}}}{\epsilon^{\frac{10}{3}}}\right) \quad (11)$$

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

Remark 1: Consistent with Other Algorithms. Theorem 2 demonstrates SCott can be seen as a general form of other control variate type algorithms. Specifically, when we stratify the time series by random hashing, i.e., cyclically assign each example into B strata, then SCott matches with SCSG (Li & Li, 2018). On the other hand, if we adopt finest-grained stratification, i.e., $B = |\mathcal{D}|$, then SCott will be aligned with SVRG (Johnson & Zhang, 2013).

[4] $\mathrm{Geo}(p)$ is the geometric distribution with p, i.e., $\mathbb{P}(K_t = K) = p^K(1-p), \forall K = 0, 1, \cdots$

Remark 2: Reduced Variance Dependency. Note that $\sum_{i=1}^B w_i^2 \sigma_i^2 = O\left(\epsilon^2\right)$ can always be fulfilled since we can at least select $B = |\mathcal{D}|$ 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 σ^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_t. The number of inner loops per outer loop (K_t), 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 $\|\boldsymbol{v}^{(t,k)}\|^2 \leq \gamma \|\boldsymbol{v}^{(t,0)}\|^2$ as an additional criterion to terminate the inner loop for some hyperparameter γ.

Remark 4: Improved Complexity Compared to Stochastic Optimizers. Carmon et al. (2019) shows the theoretical lower bound on the complexity of stochastic optimizers is $\Omega\left(\Delta L \sigma^2 \epsilon^{-4}\right)$, Comparing Corollary 1 with this bound, we can observe a complexity improvement of at least $O\left(\epsilon^{-\frac{2}{3}}\right)$ 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.

Setting	Optimizer	Exchange Rate		Traffic		Electricity	
		Training	Test	Training	Test	Training	Test
MLP NLL	SGD	-1.825 $\pm$ 0.013	-1.715 $\pm$ 0.017	-2.387 $\pm$ 0.015	-2.443 $\pm$ 0.021	6.512 $\pm$ 0.028	5.558 $\pm$ 0.021
	SCSG	-2.037 $\pm$ 0.009	**-1.732** $\pm$ 0.019	-2.612 $\pm$ 0.013	-2.674 $\pm$ 0.024	6.424 $\pm$ 0.017	5.477 $\pm$ 0.022
	SCott	**-2.145 $\pm$ 0.008**	-1.685 $\pm$ 0.009	**-2.867 $\pm$ 0.019**	**-2.812 $\pm$ 0.020**	**6.303 $\pm$ 0.018**	**5.348 $\pm$ 0.016**
	Adam	-2.762 $\pm$ 0.009	-2.945 $\pm$ 0.012	-2.597 $\pm$ 0.021	-2.593 $\pm$ 0.015	6.388 $\pm$ 0.025	5.483 $\pm$ 0.021
	S-Adam	**-3.917 $\pm$ 0.009**	**-3.032 $\pm$ 0.006**	**-3.038 $\pm$ 0.011**	**-3.012 $\pm$ 0.018**	**5.737 $\pm$ 0.023**	**5.015 $\pm$ 0.014**
	Adagrad	-3.488 $\pm$ 0.011	-3.218 $\pm$ 0.011	-2.692 $\pm$ 0.012	-2.763 $\pm$ 0.021	6.185 $\pm$ 0.019	5.295 $\pm$ 0.017
	S-Adagrad	**-3.886 $\pm$ 0.007**	**-3.239 $\pm$ 0.012**	**-2.864 $\pm$ 0.013**	**-2.895 $\pm$ 0.018**	**6.022 $\pm$ 0.020**	**5.182 $\pm$ 0.029**
N-BEATS MAPE	SGD	1.224 $\pm$ 0.018	1.346 $\pm$ 0.021	2.798 $\pm$ 0.008	3.024 $\pm$ 0.019	0.628 $\pm$ 0.023	0.676 $\pm$ 0.030
	SCSG	**1.034 $\pm$ 0.016**	**1.182 $\pm$ 0.019**	2.024 $\pm$ 0.012	2.644 $\pm$ 0.017	0.610 $\pm$ 0.021	0.635 $\pm$ 0.029
	SCott	1.077 $\pm$ 0.022	1.222 $\pm$ 0.012	**1.898 $\pm$ 0.013**	**2.373 $\pm$ 0.011**	**0.516 $\pm$ 0.020**	**0.546 $\pm$ 0.021**
	Adam	0.695 $\pm$ 0.021	0.773 $\pm$ 0.018	1.013 $\pm$ 0.012	1.036 $\pm$ 0.019	0.589 $\pm$ 0.017	0.697 $\pm$ 0.021
	S-Adam	**0.514 $\pm$ 0.012**	**0.593 $\pm$ 0.021**	**0.809 $\pm$ 0.021**	**0.813 $\pm$ 0.021**	**0.445 $\pm$ 0.025**	**0.528 $\pm$ 0.011**
	Adagrad	0.764 $\pm$ 0.022	0.806 $\pm$ 0.012	2.068 $\pm$ 0.012	1.997 $\pm$ 0.018	0.537 $\pm$ 0.011	0.631 $\pm$ 0.025
	S-Adagrad	**0.563 $\pm$ 0.013**	**0.692 $\pm$ 0.009**	**1.486 $\pm$ 0.014**	**1.724 $\pm$ 0.018**	**0.428 $\pm$ 0.020**	**0.512 $\pm$ 0.016**

four representative settings in forecasting problems:

- Vector AutoRegressive Model (VAR) with MSE loss (Lai et al., 2018).
- LSTM with MSE loss (Maharaj et al., 2019). We set the hidden layer size to be 100 and the depth to be 2.
- Simple FeedForward Network (MLP) with Negative Log Likelihood (NLL) loss (Alexandrov et al., 2019). We set the hidden layer size to be 80 and the depth to be 4.
- N-BEATS with MAPE loss (Oreshkin et al., 2019). We set the number of stacks to be 30.

7.1. Warm-up: Synthetic Dataset

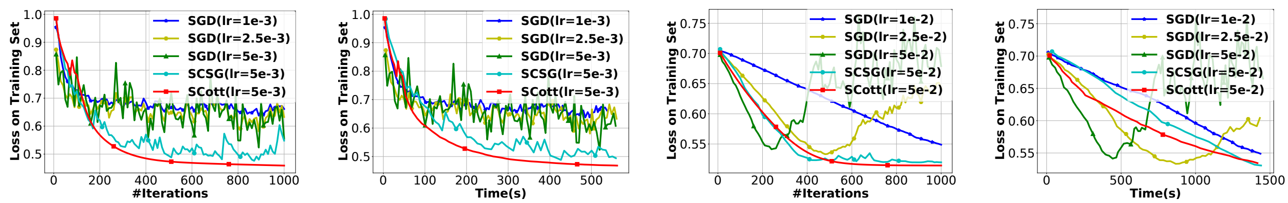
We start from training VAR and LSTM on a synthetic dataset, where we know the ground-truth for stratification. We generate the synthetic dataset by repeatedly transforming a linear curve based on simple functions such as *sin* and *polynomial*. For brevity, the details for generating the dataset is discussed 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 rate. 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

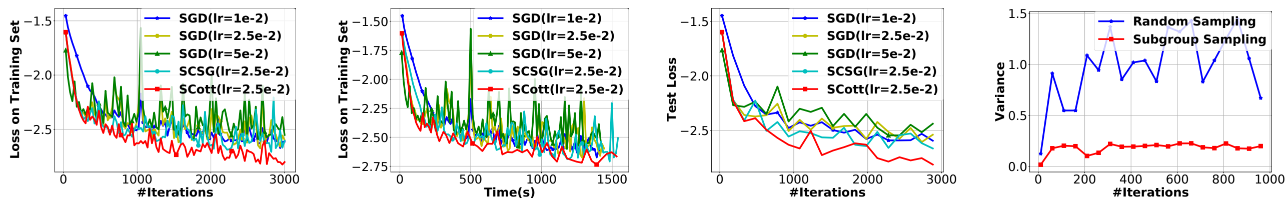


(a) Loss w.r.t. # Iterations
VAR - Synthetic Dataset

(b) Loss w.r.t. Time(s)
VAR - Synthetic Dataset

(c) Loss w.r.t. # Iterations
LSTM - Synthetic Dataset

(d) Loss w.r.t. Time(s)
LSTM - Synthetic Dataset

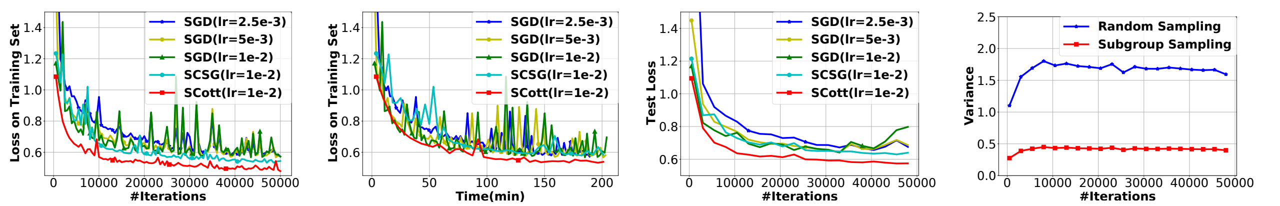


(e) Training Loss / # Iterations
MLP - Traffic Dataset

(f) Training Loss / Time(s)
MLP - Traffic Dataset

(g) Test Loss / # Iterations
MLP - Traffic Dataset

(h) Variance of Samplers
MLP - Traffic Dataset



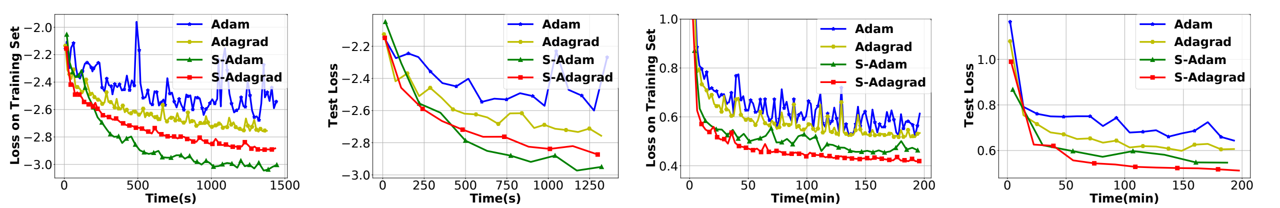
(i) Training Loss / # Iterations
N-BEATS - Electricity Dataset

(j) Training Loss / Time(min)
N-BEATS - Electricity Dataset

(k) Test Loss / # Iterations
N-BEATS - Electricity Dataset

(l) Variance of Samplers
N-BEATS - Electricity Dataset

Figure 2. Performance of different optimizers. This experiment focus on comparing SCott with original SGD-type algorithms.



(a) Training Loss / Time(s)
MLP - Traffic Dataset

(b) Test Loss / Time(s)
MLP - Traffic Dataset

(c) Training Loss / Time(min)
N-BEATS - Electricity Dataset

(d) Test Loss / Time(min)
N-BEATS - Electricity Dataset

Figure 3. SCott Variants on Adam and Adagrad. The "S-" denotes SCott version of the optimizer. The results are fine tuned.

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 testset. 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 enhance 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

Abernethy, J., Awasthi, P., Kleindessner, M., Morgenstern, J., and Zhang, J. Adaptive sampling to reduce disparate performance. *arXiv preprint arXiv:2006.06879*, 2020.

Aghabozorgi, S., Shirkhorshidi, A. S., and Wah, T. Y. Time-series clustering–a decade review. *Information Systems*, 53:16–38, 2015.

Alain, G., Lamb, A., Sankar, C., Courville, A., and Bengio, Y. Variance reduction in sgd by distributed importance sampling. *arXiv preprint arXiv:1511.06481*, 2015.

Alexandrov, A., Benidis, K., Bohlke-Schneider, M., Flunkert, V., Gasthaus, J., Januschowski, T., Maddix, D. C., Rangapuram, S., Salinas, D., Schulz, J., et al. Gluonts: Probabilistic time series models in python. *arXiv preprint arXiv:1906.05264*, 2019.

Babanezhad, R., Ahmed, M. O., Virani, A., Schmidt, M., Konečnỳ, J., and Sallinen, S. Stop wasting my gradients: Practical svrg. *arXiv preprint arXiv:1511.01942*, 2015.

Babanezhad Harikandeh, R., Ahmed, M. O., Virani, A., Schmidt, M., Konečnỳ, J., and Sallinen, S. Stopwasting my gradients: Practical svrg. *Advances in Neural Information Processing Systems*, 28:2251–2259, 2015.

Bandara, K., Bergmeir, C., and Hewamalage, H. Lstm-msnet: Leveraging forecasts on sets of related time series with multiple seasonal patterns. *IEEE transactions on neural networks and learning systems*, 2020a.

Bandara, K., Bergmeir, C., and Smyl, S. Forecasting across time series databases using recurrent neural networks on groups of similar series: A clustering approach. *Expert systems with applications*, 140:112896, 2020b.

Bottou, L. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pp. 177–186. Springer, 2010.

Box, G. E., Jenkins, G. M., and Reinsel, G. C. *Time series analysis: forecasting and control*, volume 734. John Wiley & Sons, 2011.

Brahim-Belhouari, S. and Bermak, A. Gaussian process for nonstationary time series prediction. *Computational Statistics & Data Analysis*, 47(4):705–712, 2004.

Carmon, Y., Duchi, J. C., Hinder, O., and Sidford, A. Lower bounds for finding stationary points i. *Mathematical Programming*, pp. 1–50, 2019.

Chen, Y., Kang, Y., Chen, Y., and Wang, Z. Probabilistic forecasting with temporal convolutional neural network. *Neurocomputing*, 399:491–501, 2020.

Csiba, D. and Richtárik, P. Importance sampling for minibatches. *The Journal of Machine Learning Research*, 19 (1):962–982, 2018.

De Livera, A. M., Hyndman, R. J., and Snyder, R. D. Forecasting time series with complex seasonal patterns using exponential smoothing. *Journal of the American statistical association*, 106(496):1513–1527, 2011.

Defazio, A. and Bottou, L. On the ineffectiveness of variance reduced optimization for deep learning. *arXiv preprint arXiv:1812.04529*, 2018.

Durbin, J. and Koopman, S. J. *Time series analysis by state space methods*. Oxford university press, 2012.

Faloutsos, C., Flunkert, V., Gasthaus, J., Januschowski, T., and Wang, Y. Forecasting big time series: Theory and practice. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 3209–3210, 2019.

Fu, T. and Zhang, Z. Cpsg-mcmc: Clustering-based preprocessing method for stochastic gradient mcmc. In *Artificial Intelligence and Statistics*, pp. 841–850. PMLR, 2017.

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.

Gao, J., Jagadish, H., and Ooi, B. C. Active sampler: Lightweight accelerator for complex data analytics at scale. *arXiv preprint arXiv:1512.03880*, 2015.

Gasthaus, J., Benidis, K., Wang, Y., Rangapuram, S. S., Salinas, D., Flunkert, V., and Januschowski, T. Probabilistic forecasting with spline quantile function rnns. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 1901–1910, 2019.

Gneiting, T. and Katzfuss, M. Probabilistic forecasting. *Annual Review of Statistics and Its Application*, 1:125–151, 2014.

Gower, R. M., Loizou, N., Qian, X., Sailanbayev, A., Shulgin, E., and Richtárik, P. Sgd: General analysis and improved rates. *arXiv preprint arXiv:1901.09401*, 2019.

Hewamalage, H., Bergmeir, C., and Bandara, K. Recurrent neural networks for time series forecasting: Current status and future directions. *International Journal of Forecasting*, 37(1):388–427, 2021.

Holstein, K., Wortman Vaughan, J., Daumé III, H., Dudik, M., and Wallach, H. Improving fairness in machine learning systems: What do industry practitioners need? In *Proceedings of the 2019 CHI conference on human factors in computing systems*, pp. 1–16, 2019.

Horváth, S., Lei, L., Richtárik, P., and Jordan, M. I. Adaptivity of stochastic gradient methods for nonconvex optimization. *arXiv preprint arXiv:2002.05359*, 2020.

Iosifidis, V., Fetahu, B., and Ntoutsi, E. Fae: A fairness-aware ensemble framework. In *2019 IEEE International Conference on Big Data (Big Data)*, pp. 1375–1380. IEEE, 2019.

Johnson, R. and Zhang, T. Accelerating stochastic gradient descent using predictive variance reduction. *Advances in neural information processing systems*, 26:315–323, 2013.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Lai, G., Chang, W.-C., Yang, Y., and Liu, H. Modeling long- and short-term temporal patterns with deep neural networks. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pp. 95–104, 2018.

Lara-Benitez, P., Carranza-García, M., and Riquelme, J. C. An experimental review on deep learning architectures for time series forecasting. *arXiv preprint arXiv:2103.12057*, 2021.

Lee, K., Maji, S., Ravichandran, A., and Soatto, S. Meta-learning with differentiable convex optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10657–10665, 2019.

Lee, R., Kochenderfer, M. J., Mengshoel, O. J., and Silbermann, J. Interpretable categorization of heterogeneous time series data. In *Proceedings of the 2018 SIAM International Conference on Data Mining*, pp. 216–224. SIAM, 2018.

Lei, L., Ju, C., Chen, J., and Jordan, M. I. Non-convex finite-sum optimization via scsg methods. In *Advances in Neural Information Processing Systems*, pp. 2348–2358, 2017.

Li, Z. and Li, J. A simple proximal stochastic gradient method for nonsmooth nonconvex optimization. In *Advances in neural information processing systems*, pp. 5564–5574, 2018.

Liao, T. W. Clustering of time series data—a survey. *Pattern recognition*, 38(11):1857–1874, 2005.

Liberty, E., Lang, K., and Shmakov, K. Stratified sampling meets machine learning. In *International conference on machine learning*, pp. 2320–2329. PMLR, 2016.

Liu, W., Qian, H., Zhang, C., Shen, Z., Xie, J., and Zheng, N. Accelerating stratified sampling sgd by reconstructing strata. In Bessiere, C. (ed.), *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pp. 2725–2731. International Joint Conferences on Artificial Intelligence Organization, 7 2020. doi: 10.24963/ijcai.2020/378. URL https://doi.org/10.24963/ijcai.2020/378. Main track.

London, B. A pac-bayesian analysis of randomized learning with application to stochastic gradient descent. *arXiv preprint arXiv:1709.06617*, 2017.

Loshchilov, I. and Hutter, F. Online batch selection for faster training of neural networks. *arXiv preprint arXiv:1511.06343*, 2015.

Maharaj, E. A., D'Urso, P., and Caiado, J. *Time series clustering and classification*. CRC Press, 2019.

Montero-Manso, P. and Hyndman, R. J. Principles and algorithms for forecasting groups of time series: Locality and globality. *arXiv preprint arXiv:2008.00444*, 2020.

Nagaraj, D., Jain, P., and Netrapalli, P. Sgd without replacement: Sharper rates for general smooth convex functions. In *International Conference on Machine Learning*, pp. 4703–4711. PMLR, 2019.

Nelson, B. L. Control variate remedies. *Operations Research*, 38(6):974–992, 1990.

Oreshkin, B. N., Carpov, D., Chapados, N., and Bengio, Y. N-beats: Neural basis expansion analysis for interpretable time series forecasting. *arXiv preprint arXiv:1905.10437*, 2019.

Park, Y. and Ryu, E. K. Linear convergence of cyclic saga. *Optimization Letters*, 14(6):1583–1598, 2020.

Park, Y., Dhar, S., Boyd, S., and Shah, M. Variable metric proximal gradient method with diagonal barzilai-borwein stepsize. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3597–3601. IEEE, 2020.

Rangapuram, S. S., Seeger, M. W., Gasthaus, J., Stella, L., Wang, Y., and Januschowski, T. Deep state space models for time series forecasting. In *Advances in neural information processing systems*, pp. 7785–7794, 2018.

Rasul, K., Sheikh, A.-S., Schuster, I., Bergmann, U., and Vollgraf, R. Multi-variate probabilistic time series forecasting via conditioned normalizing flows. *arXiv preprint arXiv:2002.06103*, 2020.

Salinas, D., Flunkert, V., Gasthaus, J., and Januschowski, T. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3):1181–1191, 2020.

Sen, R., Yu, H.-F., and Dhillon, I. S. Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/3a0844cee4fcf57de0c71e9ad3035478-Paper.pdf.

Smyl, S. A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International Journal of Forecasting*, 36(1):75–85, 2020.

Vlahogianni, E. I., Karlaftis, M. G., and Golias, J. C. Short-term traffic forecasting: Where we are and where we're going. *Transportation Research Part C: Emerging Technologies*, 43:3–19, 2014.

Wang, Y., Smola, A., Maddix, D. C., Gasthaus, J., Foster, D., and Januschowski, T. Deep factors for forecasting. *arXiv preprint arXiv:1905.12417*, 2019.

Wang, Z., Qinami, K., Karakozis, I. C., Genova, K., Nair, P., Hata, K., and Russakovsky, O. Towards fairness in visual recognition: Effective strategies for bias mitigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8919–8928, 2020.

Ward, R., Wu, X., and Bottou, L. Adagrad stepsizes: Sharp convergence over nonconvex landscapes. In *International Conference on Machine Learning*, pp. 6677–6686, 2019.

Xu, L., Wang, S., and Tang, R. Probabilistic load forecasting for buildings considering weather forecasting uncertainty and uncertain peak load. *Applied Energy*, 237:180–195, 2019.

Yu, T., Zhai, X., and Sra, S. Near optimal stratified sampling. *arXiv e-prints*, pp. arXiv–1906, 2019.

Yu, T., Liu, X.-W., Dai, Y.-H., and Sun, J. A variable metric mini-batch proximal stochastic recursive gradient algorithm with diagonal barzilai-borwein stepsize. *arXiv preprint arXiv:2010.00817*, 2020.

Zhang, C., Kjellstrom, H., and Mandt, S. Determinantal point processes for mini-batch diversification. *arXiv preprint arXiv:1705.00607*, 2017.

Zhang, G. P. Time series forecasting using a hybrid arima and neural network model. *Neurocomputing*, 50:159–175, 2003.

Zhang, T. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 116, 2004.

Zhao, P. and Zhang, T. Accelerating minibatch stochastic gradient descent using stratified sampling. *arXiv preprint arXiv:1405.3080*, 2014.