Comparison of Gradient-based Optimization Methods for the Time Series Sample Mean Problem in DTW Space

Max Reinhard
max.reinhard@campus.tu-berlin.de
Technical University Berlin

Faraz Maschhur
f.maschhur@campus.tu-berlin.de
Technical University Berlin

ABSTRACT

Finding the mean of a given sample of time series is a NP-hard problem. Therefore optimization methods have to be applied to efficiently find (possibly sub-optimal) solutions. The adaptation of stochastic gradient descent for this problem in the form of SSG showed empirical success. Founded on that, two more advanced gradient-based optimization methods Adam and SGLD were adapted. Empirical results for these adapted methods under different configurations show that they generally perform worse than SSG. Also the results suggest that the number of update steps is most influential for the solution quality of the considered methods in respect to the examined configuration parameters.

KEYWORDS

Time series, DTW Mean, Fréchet function, Stochastic optimization

1 INTRODUCTION

Time series consist of a sequence of time dependent data points. Apart from the corresponding point in time these data points can be of one dimension (i.e. a scalar) or multiple dimensions (i.e. a vector). Examples of phenomena that are observed as time series include audio signals, electrocardio- and electroencephalograms, weather recordings and various sensor data like seismograph recordings. Different applications such as speech recognition or medical diagnosis call for analysis of this data, which includes pattern recognition and other data mining methods. Due to the varying length and speed along the time axis, many standard methods cannot be readily applied. The most common case of methods that can not be applied are those, that use distance between data points, e.g. K-Means Algorithms or Nearest Neighbour Methods. A common practice to address these issues is to apply dynamic time warping (DTW). This technique optimally aligns the time axes of two time series and provides a distance measure between them.

Extending standard data analysis methods to DTW spaces is one direction of research in data analysis for time series. A very fundamental technique for data analysis in general is averaging, i.e. the computation of a mean. "Inspired by the sample mean in Euclidean spaces, the goal of time series averaging is to construct a time series that is located in the center of a given sample of time series." [7] This is commonly done by synthesizing an average time series based on the sample time series, which were aligned with respect to the DTW distance. There exist several variations of this approach. One research direction among these is to pose time series averaging as an optimization problem: Given $\mathcal{X} = (x^{(1)}, \ldots, x^{(N)})$ as a sample of N time series $x^{(i)}$ with $i \in \{1, \ldots, N\}$, a (sample) mean in DTW space is any time series x that minimizes the Frechèt function

$$F(x) = \frac{1}{N} \sum_{k=1}^{N} \mathrm{dtw}^2 \left(x, x^{(k)} \right)$$

where dtw is the DTW distance. Because finding a global minimum of the Frechèt function is NP-hard [2], it is an ongoing research problem to develop or transfer suitable approximation algorithms for this problem. These should efficiently find (possibly sub-optimal) solutions that approximate the global optimum as good as possible. This article investigates different optimization algorithms in application to the time series sample mean problem in DTW space. The main contributions are as follows:

(1) Adaptation of the Adam[9] and SGLD[4] gradient descent optimization methods to the time series sample mean problem in DTW space.
(2) Empirical assessment of the suitability of the aforementioned algorithms with different configurations for this very problem.

A widely used form of optimization methods are those based on gradient descent[6]. Since the Fréchet function is non-differentiable, it is not possible to directly apply gradient-based optimization methods. Instead we have adapted these methods to work with subgradients, as it has been done with the stochastic gradient descent method in the form of the stochastic subgradient method (SSG) in [7]. Correspondingly we realise our adaptations of Adam and SGLD also as stochastic methods subgradient Adam (SAdam) and subgradient SGLD (SSGLD). One parameter to choose for the configuration of stochastic (sub-)gradient based methods is the number of samples to incorporate into the computation of the (sub-)gradients, commonly referred to as (mini-)batch size. We compare the empirical results we achieve with SSG, SAdam and SSGLD on multiple datasets using different configurations.

The rest of the article is structured as follows: Section 2 provides an overview of the theoretical background. In Section 3 our adapted optimization methods are introduced. Section 4 explains the experimental setup, presents the empirical results and discusses them. Finally Section 5 concludes.

2 BACKGROUND AND RELATED WORK

In this section we give mathematical definitions for the concepts needed to explain our work, such as DTW distance and the Fréchet function. These definitions are very closely adapted from [7]. We then proceed to introduce the SSG algorithm.

2.1 Timeseries and Warping Paths

A time series $x = (x_1, ..., x_m)$ is defined as an ordered sequence of length m containing elements $x_i \in \mathbb{R}^d$ where $i \in \{1, ..., m\}$. For a fixed dimension $d \in \mathbb{N}$, we write $\mathcal{T}_*$ for the set containing all time series of finite length. By $\mathcal{T}_m$ we denote the set of all time series of length $m \in \mathbb{N}$.

A warping path $p = (p_1, ..., p_L)$ of order $m \times n$ is a sequence of L points $p_l = (i_l, j_l) \in \{1, ..., m\} \times \{1, ..., n\}$ with $m, n \in \mathbb{N}$ where the following holds

(1) $p_1 = (1, 1)$ & $p_L = (m, n)$

(2) $p_{l+1} - p_l \in \{(1, 0), (0, 1), (1, 1)\}$ for all $l \in [L - 1]$

$\mathcal{P}_{m,n}$ denotes the set containing all paths of order $m \times n$. For fixed $m, n \in \mathbb{N}$, $\mathcal{P}^N$ denotes the set of all possible lists with length N of warping paths $p \in \mathcal{P}_{m,n}$. A warping path $p = (p_1, ..., p_L) \in P_{m,n}$ defines an alignment of two time series $x = (x_1, ..., x_m)$ and $y = (y_1, ..., y_n)$. Every point $p \ni p_l = (i_l, j_l)$ aligns element $x_{i_l} \in x$ to element $y_{j_l} \in y$. The cost of such alignments is defined as

$$C_p(x, y) = \sum_{l=1}^{L} \|x_{i_l} - y_{j_l}\|^2.$$

2.2 Dynamic Time Warping Distance

The DTW distance is defined over the minimal cost of aligning two time series x and y respectively with length m and n over all warping paths in $\mathcal{P}_{m,n}$

$$\operatorname{dtw}(x, y) = \min \left\{ \sqrt{C_p(x, y)} : p \in \mathcal{P}_{m,n} \right\}$$

If $\operatorname{dtw}(x, y) = \sqrt{C_p(x, y)}$ holds true for a warping path $p \in P_{m,n}$, we call that warping path optimal.

2.3 Fréchet Function

Let $\mathcal{X} = \left(x^{(1)}, ..., x^{(N)}\right)$ be a sample which contains N time series $x^{(k)} \in \mathcal{T}_*$. The Fréchet function of sample $\mathcal{X}$ is defined as

$$F : \mathcal{T}_n \to \mathbb{R},\ x \mapsto \frac{1}{N} \sum_{k=1}^{N} \operatorname{dtw}^2 \left(x, x^{(k)}\right)$$

The value $F(x)$ is the Fréchet variation of $\mathcal{X}$ at $x \in \mathcal{T}_n$.

The sample mean set of $\mathcal{X}$ is the set

$$\mathcal{F} = \{z \in \mathcal{T}_n : F(z) \leq F(x) \text{ for all } x \in \mathcal{T}_n\}$$

Each element of $\mathcal{F}$ is called a (sample) mean of $\mathcal{X}$. A sample mean is a time series that minimizes the Fréchet variation of the given sample. Finding such a sample mean is called the time series sample mean problem.

2.4 Configuration and Component Function

We define a configuration as an ordered list $C = \left(p^{(1)}, ..., p^{(N)}\right) \in \mathcal{P}^N$ of warping paths. We define a component function of $F(x)$ as

$$F_C : \mathbb{R}^n \to \mathbb{R}, x \mapsto \frac{1}{N} \sum_{k=1}^{N} C_{p^{(k)}} \left(x, x^{(k)}\right)$$

where $C = \left(p^{(1)}, ..., p^{(N)}\right)$ is a configuration.

Given a sample $\mathcal{X} = \left(x^{(1)}, ..., x^{(N)}\right) \in \mathcal{T}^N$ we can write the Fréchet function as shown in [7]:

$$F(x) = \min_{C \in \mathcal{P}^N} F_C(x)$$

If $F_C(x) = F(x)$ holds true, F_C is an active component at x and the configuration C is optimal at x. $\mathcal{A}_{\mathcal{F}}$ denotes the set of all active component functions of F at x.

2.5 Valence and Warping Matrix

Given a warping path $p \in \mathcal{P}_{n,n}$

(1) the warping matrix of p is a matrix $W \in \{0, 1\}^{n \times n}$ with elements

$$W_{i,j} = \begin{cases} 1 : & (i, j) \in p \\ 0 : & else \end{cases}$$

(2) the valence matrix of p is the diagonal matrix $V \in \mathbb{N}^{n \times n}$ with elements

$$V_{i,i} = \sum_{j=1}^{n} W_{i,j}$$

2.6 Subgradients of the Fréchet Function

In [7] it was shown that the gradient of an active component function $\mathcal{F}_C$ at x is a subgradient of the Fréchet function F at x. Also it was concluded that it is sufficient to define the gradients of the component functions to calculate a subgradient of F, because for every $x \in \mathcal{T}$ there is an active component function $\mathcal{F}_C \in \mathcal{A}_{\mathcal{F}}$ at x.

The gradient of a component function $\mathcal{F}_C$ at $x \in \mathcal{T}$ given a sample of time series $\mathcal{X} = \left(x^{(1)}, ..., x^{(N)}\right) \in \mathcal{T}^N$ and a configuration of warping paths $C \in \mathcal{P}^N$ is given by

$$\nabla F_C(x) = \frac{2}{N} \sum_{k=1}^{N} \left(V^{(k)} x - W^{(k)} x^{(k)}\right)$$

with $V^{(k)}$ being the valence matrix and $W^{(k)}$ being the warping matrix of warping path $p^{(k)} \in C$.

2.7 Stochastic Subgradient Mean Algorithm

To find a sample mean $z \in \mathcal{F}$ for a given sample of time series $\mathcal{X} = \left(x^{(1)}, ..., x^{(N)}\right) \in \mathcal{T}^N$ is the problem of minimizing the Fréchet function of sample $\mathcal{X}$. This problem is NP-hard and therefore there is no algorithm known which is able to solve it in polynomial time. Since there is a way to find a subgradient $\nabla F(x)$ of the Fréchet function $F(x)$ for any given time series x, we can use stochastic gradient descent methods to approximately find a local minimum of $F(x)$ in reasonable time.

The stochastic subgradient mean (SSG) algorithm has been shown to outperform other state-of-the-art subgradient methods in finding the sample mean of a sample of time series. [7]

The SSG algorithm initializes a solution z and best solution z^* using the same random time series of the sample $\mathcal{X} = \left(x^{(1)}, ..., x^{(N)}\right)$ for both. Then, in each iteration t, it randomly chooses a $k \in$

$\{1, ..., N\}$ calculates the warping path $p^{(k)}$ between z and $x^{(k)}$ and finds a new z using the subgradient $\nabla F^{(k)}_{p^{(k)}}(x) = 2\left(V^{(k)}x - W^{(k)}x^{(k)}\right)$ by calculating

$$z = z - 2\eta\left(V^{(k)}x - W^{(k)}x^{(k)}\right)$$

Then the best solution is updated by comparison of the old best solution and the new z

$$z^* = \underset{x \in \{z, z^*\}}{\operatorname{argmin}} F(x)$$

This update rule utilizes the same procedure as seen in the gradient descent approach [1].

3 ADAPTED OPTIMIZATION METHODS

Following up on the success of SSG in application to the sample mean problem for time series, we adapted two more advanced gradient-based optimization methods for this domain. Also we add some changes to the data processing procedure of SSG for the use in this study.

3.1 SSG

We modified the SSG algorithm from [7] so that it does not rely on a fixed number of epochs anymore. Instead we adapted it to visit a specific number of samples during one iteration, which is given via the parameter n_{coverage}. If the given dataset is smaller than that, some samples will be visited multiple times. The idea behind this modification is that a fixed number of epochs means that the total number of updates is dependent on the dataset size. Given a small dataset this could result in too few updates to find a suitable solution and given a larger dataset this results in possibly unnecessarily longer computation time. To further mitigate this problem we also included a convergence threshold $d_{\text{convergence}}$: if the relative difference between the quality of the solution after an epoch and the solution after the previous epoch is smaller than this threshold, the algorithm stops.

Additionally we introduce the concept of batch updates to the SSG algorithm. The batch size s_{batch} defines the number of samples that are incorporated into each subgradient computation. Consequently the number of updates steps performed is given by $n_{\text{updates}} = \lceil n_{\text{coverage}}/s_{\text{batch}} \rceil$. When n_{coverage} is not fully divisible by s_{batch}, an extra batch is performed to ensure that minimally n_{coverage} samples are visited.

3.2 SAdam

Adam is a very popular gradient descent optimization algorithm[1] and has been widely and successfully applied to train deep neural networks in various fields.[2] The key ideas behind Adam (which is short for "Adaptive Moment Estimation") are momentum and adaptive learning rates for each parameter dimension. Momentum means to increase the learning rate for dimensions where the gradients successively point in the same direction and reduce the learning rate for those dimensions where the gradient changes direction. To realize this Adam stores an exponentially decaying average of past gradients m and squared gradients v. We adapted Adam to work with subgradients and to process the data in the same way as it is described for our version of the SSG algorithm. For pseudocode of SAdam see Algorithm 1.

[1]The respective Google Scholar page lists nearly 50k citations (as of 13.06.2020): https://scholar.google.de/scholar?cites=16194105527543080940&as_sdt=2005
[2]This blogpost shows that Adam was mentioned in about 20% of the papers published to arXiv in relevant categories from 2015 to 2017: https://medium.com/@karpathy/a-peek-at-trends-in-machine-learning-ab8a1085a106

3.3 SSGLD

SGLD (short for "Stochastic Gradient Langevin dynamics") is an optimization algorithm, which is mainly used in machine learning problems. SGLD has been shown to perform well on a variety of tasks [5] [8]. When optimizing some function $f(z)$, the main idea of SGLD is using scaled gaussian noise in the update procedure to be able to escape the local minima of the function. The algorithm chooses a starting point using random values in the function domain space. This point is saved in the set of possible solutions $\mathcal{Z}$. Then, in each iteration, the current point is updated according to gradient descent, with the addition of adding scaled gaussian noise to it. If the gaussian noise has moved the new point out of the function domain space or if the euclidian distance between the old and the new point is bigger than some hyperparameter D, the update will not have any effect and the old point is kept. If the point changes, it is saved to $\mathcal{Z}$. In the end, SGLD calculates the value of $f(z)$ for every element $z \in \mathcal{Z}$ and returns the z that yielded the smallest function value.

In SSGLD (our adaptation of the SGLD algorithm) we enabled the algorithm to use subgradients and use the DTW distance measure instead of euclidian distance. We determined the hyperparameters by empirical testing and oriented towards the results in [9]. Regarding the data processing SSGLD follows the same procedure as our version of SSG.

Although SSGLD poses as a promising approach, the computational effort behind the algorithm has to be noted. For every update step, that results in a new entry in $\mathcal{Z}$, SSGLD has to calculate the Frechét variation for that point. The functionality of SSGLD is shown in Algorithm 2.

4 EXPERIMENTS

To compare the quality of the solutions acquired by the presented subgradient-based optimization methods, we conducted a series of experiments with different configurations of the algorithms on multiple time series datasets.

4.1 Data

The datasets we used in the experiments are part of the UCR Time Series Classification Archive [3]. We selected 24 of these datasets following the selection in [7] for comparability. Each time series within one dataset has the same length. As this data was designed for the evaluation of classification systems, each time series has been assigned a class, which were omitted. Additionally the datasets are split into a train set and a test set, which have been merged for our experiments. The selected datasets and basic statistics about them can be found in Table 1.

Algorithm 1: SAdam

```
Data: Sample $\mathcal{X} = \{x^{(1)}, ..., x^{(N)}\}$ where $x \in \mathbb{R}^d$
Required parameters: $n_{coverage}$, $s_{batch}$, $d_{convergence}$
Fixed parameters:
$\alpha \longleftarrow 0.001$ (Stepsize)
$\beta_1 \longleftarrow 0.9$, $\beta_2 \longleftarrow 0.999$ (Decay rates)
$\epsilon \longleftarrow 10^{-8}$
Initialization:
$n_{epochs} \longleftarrow \lceil n_{coverage}/N - (N \mod s_{batch}) \rceil$
$n_{batches} \longleftarrow \lfloor N/s_{batch} \rfloor$
$i_{samples} \longleftarrow 0$
$m \longleftarrow \mathbf{0} \in \mathbb{R}^d$, $v \longleftarrow \mathbf{0} \in \mathbb{R}^d$, $t \longleftarrow 1$
$z \longleftarrow x \in \mathcal{X}$ (Picked randomly), $z^* \longleftarrow z$ (Best solution)
$f_0 \longleftarrow F(z)$
for $k \longleftarrow 1$ to $n_{epochs}$ do
    Shuffle sample $\mathcal{X}$ (changes indices of all $x^{(i)} \in \mathcal{X}$)
    $i \longleftarrow 0$
    for $h \longleftarrow 1$ to $n_{batches}$ do
        if $i_{samples} \geq n_{coverage}$ then
            break
        end
        Bag of subgradients $SG \longleftarrow \{\}_b$
        for $l \longleftarrow 1$ to $s_{batch}$ do
            $p^{(i+1)} \longleftarrow$ optimal warping path of $z$ and $x^{(i+1)}$
            $V^{(i+l)} \longleftarrow$ valence matrix of $p^{(i+1)}$
            $W^{(i+l)} \longleftarrow$ warping matrix of $p^{(i+1)}$
            $sg^{(l)} \longleftarrow V^{(i+l)} z - W^{(i+l)} x^{(i+l)}$
            $SG \longleftarrow SG \cup \left\{ sg^{(l)} \right\}_b$
        end
        $sg \longleftarrow \frac{1}{s_{batch}} \sum_{l=1}^{s_{batch}} sg^{(l)}$
        $m \longleftarrow \beta_1 m + (1 - \beta_1) \cdot sg$
        $v \longleftarrow \beta_2 v + (1 - \beta_2) \cdot sg^2$
        $\hat{m} \longleftarrow m/(1 - \beta_1^{\,t})$
        $\hat{v} \longleftarrow v/(1 - \beta_2^{\,t})$
        $z \longleftarrow z - \alpha \cdot \hat{m}/(\sqrt{\hat{v}} + \epsilon)$ (Update solution)
        $t \longleftarrow t + 1$
        $i \longleftarrow i + s_{batch}$, $i_{samples} \longleftarrow i_{samples} + s_{batch}$
    end
    $f_k \longleftarrow F(z)$
    if $f_k < f_{k-1}$ then
        $z^* \longleftarrow z$
    end
    if $\left| \frac{f_k - f_{k-1}}{f_{k-1}} \right| < d_{convergence}$ then
        break
    end
end
Output: $z^*$
```

Algorithm 2: SSGLD

```
Data: Sample $\mathcal{X} = \{x^{(1)}, ..., x^{(N)}\}$ where $x^{(i)} \in \mathbb{R}^d$
Required parameters: $n_{coverage}$, $s_{batch}$
Fixed parameters: $\eta \longleftarrow 0.05$
Initialization:
$n_{epochs} \longleftarrow \lceil n_{coverage}/N - (N \mod s_{batch}) \rceil$
$n_{batches} \longleftarrow \lfloor N/s_{batch} \rfloor$
$i_{samples} \longleftarrow 0$
$\mathbb{R}^d \ni x_{max} \longleftarrow$ vector of max per dimension of all $x^{(i)} \in \mathcal{X}$
$\mathbb{R}^d \ni x_{min} \longleftarrow$ vector of min per dimension of all $x^{(i)} \in \mathcal{X}$
$\xi \longleftarrow \frac{8500}{\| \min(x_{max}) \|}$, D $\longleftarrow 8\sqrt{\frac{2\eta d}{\frac{\xi}{100}}}$
$z_0 \longleftarrow$ starting point with random values between $x_{max}$ and $x_{min}$ in every dimension
$\mathcal{Z} \longleftarrow \{z_0\}$
for $k \longleftarrow 1$ to $n_{epochs}$ do
    Shuffle sample $\mathcal{X}$ (changes indices of all $x^{(i)} \in \mathcal{X}$)
    $i \longleftarrow 0$
    for $h \longleftarrow 1$ to $n_{batches}$ do
        if $i_{samples} \geq n_{coverage}$ then
            break
        end
        Bag of subgradients $SG \longleftarrow \{\}_b$
        for $l \longleftarrow 1$ to $s_{batch}$ do
            $p^{(i+1)} \longleftarrow$ optimal warping path of $z$ and $x^{(i+1)}$
            $V^{(i+l)} \longleftarrow$ valence matrix of $p^{(i+1)}$
            $W^{(i+l)} \longleftarrow$ warping matrix of $p^{(i+1)}$
            $sg^{(l)} \longleftarrow V^{(i+l)} z - W^{(i+l)} x^{(i+l)}$
            $SG \longleftarrow SG \cup \left\{ sg^{(l)} \right\}_b$
        end
        $sg \longleftarrow \frac{1}{s_{batch}} \sum_{l=1}^{s_{batch}} sg^{(l)}$
        $w_h \longleftarrow \mathcal{N}(\mathbf{0}, \mathbf{1}) \in \mathbb{R}^d$
        $y_h \longleftarrow z_{h-1} - \eta \cdot sg + \sqrt{\frac{2 \cdot \eta}{\xi}} \cdot w$
        $z_h \longleftarrow \begin{cases} y_h & x_{min} \leq y_h \leq x_{max} \;\; \& \\ & dtw(x_{h-1}, y_h) < D \\ z_{h-1} & else \end{cases}$
        $\mathcal{Z} \longleftarrow \mathcal{Z} \cup \{z_h\}$
        $i \longleftarrow i + s_{batch}$
        $i_{samples} \longleftarrow i_{samples} + s_{batch}$
    end
end
Output: $z^* \longleftarrow \mathrm{argmin}_{z \in \mathcal{Z}}(F(z))$
```

Algorithm 3: Experimental Procedure

```
foreach dataset in selected datasets do
    foreach algorithm in algorithm configurations do
        for iteration ← 1 to 30 do
            Timed application of algorithm to dataset
            Yields Fréchet variaton $F(z^*)$ and runtime
        end
    end
end
```

Table 1: Basic statistics of the used datasets

Name	Length	Samples	Classes
Adiac	176	781	37
Beef	470	60	5
CBF	128	930	3
ChlorineConcentration	166	4307	3
Coffee	286	56	2
ECG200	96	200	2
ECG5000	140	5000	5
ElectricDevices	96	16637	7
FaceAll	131	2250	14
FaceFour	350	112	4
FiftyWords	270	905	50
Fish	463	350	7
GunPoint	150	200	2
Lightning2	637	121	2
Lightning7	319	143	7
OliveOil	570	60	4
OSULeaf	427	442	6
PhalangesOutlinesCorrect	80	2658	2
SwedishLeaf	128	1125	15
SyntheticControl	60	600	6
Trace	275	200	4
TwoPatterns	128	5000	4
Wafer	152	7164	2
Yoga	426	3300	2
mean	255.17	2191.71	8.12
median	171.00	690.50	4.00
min	60.00	56.00	2.00
max	637.00	16637.00	50.00

Table 2: Experimentally evaluated configurations

Name	Method	n_{coverage}	s_{batch}
ssg-1000-1	SSG	1000	1
ssg-1000-10	SSG	1000	10
ssg-2000-5	SSG	2000	5
sadam-1000-1	SAdam	1000	1
sadam-1000-10	SAdam	1000	10
sadam-2000-5	SAdam	2000	5
ssgld-2000-5	SSGLD	2000	5

4.2 Algorithms

The previously described optimization methods SSG, SAdam and SSGLD were evaluated on the listed datasets using different configurations. To evaluate the influence of n_{coverage} and s_{batch}, different configurations of the algorithms regarding these parameters were conceived. In contrast, the convergence threshold is always set to $d_{\text{convergence}} = 0.0001$. The evaluated configurations are listed in Table 2.

4.3 Experimental Setup

Every configuration was executed on every dataset for 30 iterations, amounting to $24 \cdot 30 = 720$ iterations per configuration. The experimental procedure is described by Algorithm 3. An average of the Fréchet variation and standard deviation over the 30 iterations of every configuration of the algorithms is reported.

While deciding on an experimental setup, we tested different $n_{coverage}$ and s_{batch} to use for the algorithms and found that executing SSGLD with a great number of n_{updates} (i.e. high $n_{coverage}$ and low s_{batch}) is too computationally expensive. Because of that, we decided to only run one version of SSGLD with an $n_{coverage} = 2000$ and $s_{batch} = 5$ in our main experiment line, but to keep the results of the early SSGLD testing with different parameters for comparison.

The source code for the experiments including the configurations discussed in subsection 4.2 has been made available to members of this institution (TU Berlin): https://gitlab.tubit.tu-berlin.de/wuxmax/ssp-dtw-mean-optimization

4.4 Results and Discussion

The full results for all datasets and all algorithms can be found in Table 5. For better comparability, we plotted the mean variation per dataset and algorithm relative to the mean variation of all algorithms for the dataset in Figure 1. At this point it is important to outline, that the experiment line regarding the ssgld-2000-5 configuration did not terminate in time, which results in no values for that algorithm for the datasets TwoPatterns, Wafer and Yoga.

As explained in subsection 4.3 for SSGLD we only ran ssgld-2000-5 in the main experiment line. However, we have results for two further configurations of SSGLD (ssgld-1000-1: $n_{coverage} = 1000$, $s_{batch} = 1$ and ssgld-1000-10: $n_{coverage} = 1000$, $s_{batch} = 10$) on four datasets: Coffee, Beef, FaceFour and OliveOil.

To compare the effectiveness of different configurations, we measured the percentage of datasets, where one specific configuration outperforms other configurations of the same method. ssgld-2000-5 outperforms ssgld-1000-1 and ssgld-1000-10 each in 75% of the datasets. Also, ssgld-1000-1 outperforms ssgld-1000-10 in 75% of the datasets. By outperform we mean, that the mean variation over all iterations of a specific configuration is lower than the respective value for another configuration on the same dataset. These results are listed in Table 3.

For the SSGLD algorithms we have found that, on the four given datasets, ssgld-2000-5 is outperforming its counterparts. However, the amount of different datasets in this comparison is not large enough to generalize this claim and further experiments need to be conducted to draw definite conclusions.

	ssgld-1000-1	sgld-1000-10	sgld-2000-5
ssgld-1000-1	0.00	25.00	75.00
ssgld-1000-10	75.00	0.00	75.00
ssgld-2000-5	25.00	25.00	0.00

Table 3: Comparison of different configurations for SSGLD (win percentage of column vs. row)

The experiments for the two other types of algorithms, namely SSG and SAdam, were conducted with three different configurations for $n_{coverage}$ and s_{batch} for each method. The results of the pairwise comparison the different configurations of these algorithms can be seen in Table 4.

The results show that ssg-1000-1 [sadam-1000-1] outperforms ssg-1000-10 [sadam-1000-10] for 79.17% [83.33%] of the datasets. Also it outperforms ssg-2000-5 [sadam-2000-5] for 70.83% [70.83%] of the datasets. Notable is that ssg-1000-10 outperforms ssg-1000-10 in 20.83% of the datasets while it outperforms ssg-2000-5 in 16.67% of the datasets.

As the data suggests, the $n_{coverage} = 1000$, $s_{batch} = 1$ configurations mostly outperform the other configurations in SSG and SAdam. This might be due to the algorithm doing 1000 update steps in this configuration, whereas it does 400 update steps in the $n_{coverage} = 2000$, $s_{batch} = 5$ configuration and only 100 update steps for $n_{coverage} = 1000$, $s_{batch} = 10$. Despite what seems like a natural assumption, visiting more datapoints (choosing a high $n_{coverage}$) in the optimization process is not enough, for the algorithm to perform well. The more important parameter seems to be the number of updates made. On average, a higher number of updates done during an algorithms runtime results in generally better performance.

During the execution of the experiments we noticed a large difference in computational demand between SSGLD and the other algorithms. While the experiment line for all configurations of the SSG and SAdam algorithms took about one day to terminate, the ssgld-2000-5 runs on all datasets took a week to finish. This is due to the computational effort behind calculating the value of the Fréchet function, which SSGLD has to do significantly more frequently.

When comparing all algorithms and their configurations, there is a clear winner in performance:

On 79.17% of the datasets some configuration of SSG outperforms all other algorithms. In the remaining 20.83% of the datasets some configuration of SAdam outperforms all other algorithms. On average, SSG outperforms the other algorithms. Since we have conducted the experiment line using only a single configuration of SSGLD, we cannot fully expand our findings onto SSGLD as a whole. However, in our small sample size, ssgld-2000-5 is the best performing configuration we found.

Even when comparing the SSG algorithm to other, more sophisticated optimization methods like SAdam or SSGLD, the SSG algorithm still mostly outperforms the other algorithms. While SAdam is only slightly outperformed by SSG in most cases and even outperforms SSG in some, SSGLD loses heavily on nearly all the datasets. One reason for this could be the complex hyperparameter configuration of SSGLD, which might have a large effect on its general performance in different settings.

We found no correlation between the length or the amount of samples of the datasets and the performance of algorithms on them.

5 CONCLUSION

The basic stochastic gradient descent method has been adapted for the time series sample mean problem in DTW space in the form of SSG and showed empirical success. This work follows up on that and contributes the following:

(1) Adapted two more advanced gradient-descent based optimization methods for the time series sample mean problem in DTW space and modified the data processing procedure of SSG:
  (a) Introduced the concept of n_{coverage} and s_{batch} to SSG
  (b) Adapted Adam as SAdam
  (c) Adapted SGLD as SSGLD
(2) Empirical evaluation of of SSG, SAdam and SSGLD with different configurations on multiple datasets:
  (a) Comparison of the different optimizations methods clearly shows that SSG generally outperforms the other methods, albeit being less sophisticated.
  (b) Comparison of n_{coverage}, s_{batch} and the resulting n_{updates} suggest that the number of updates contributes more to an algorithms performance than the number of visited samples (given the scope of parameters values, which were evaluated).

The contributions in (1) can be used as ground work to further examine different state-of-the-art optimization methods in the context of the time series sample mean problem. Further future work may consist of more research in the determination of hyperparameters for the SSGLD algorithms and further experiments with different configurations for all algorithms to gain more insight in the sense of contribution (2)(b).

REFERENCES

[1] Léon Bottou. 2010. Large-Scale Machine Learning with Stochastic Gradient Descent. In *Proceedings of the 19th International Conference on Computational Statistics (COMPSTAT'2010)*, Yves Lechevallier and Gilbert Saporta (Eds.). Springer, Paris, France, 177–187. http://leon.bottou.org/papers/bottou-2010

[2] Laurent Bulteau, Vincent Froese, and Rolf Niedermeier. 2018. Hardness of Consensus Problems for Circular Strings and Time Series Averaging. (04 2018).

[3] Hoang Anh Dau, Anthony Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, and Eamonn Keogh. 2018. The UCR Time Series Archive. arXiv:1810.07758 [cs.LG]

[4] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2015).

[5] Chunyuan Li, Changyou Chen, David E Carlson, and Lawrence Carin. 2016. Preconditioned Stochastic Gradient Langevin Dynamics for Deep Neural Networks.. In *AAAI*, Vol. 2. 4.

[6] Sebastian Ruder. 2016. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747* (2016).

[7] David Schultz and Brijnesh J. Jain. 2017. Nonsmooth Analysis and Subgradient Methods for Averaging in Dynamic Time Warping Spaces. *CoRR* abs/1701.06393 (2017). arXiv:1701.06393 http://arxiv.org/abs/1701.06393

[8] Max Welling and Yee Whye Teh. 2011. Bayesian Learning via Stochastic Gradient Langevin Dynamics. In *Proceedings of the 28th International Conference on International Conference on Machine Learning* (Bellevue, Washington, USA) *(ICML'11)*. Omnipress, Madison, WI, USA, 681–688.

[9] Yuchen Zhang, Percy Liang, and Moses Charikar. 2017. A Hitting Time Analysis of Stochastic Gradient Langevin Dynamics. *CoRR* abs/1702.05575 (2017). arXiv:1702.05575 http://arxiv.org/abs/1702.05575

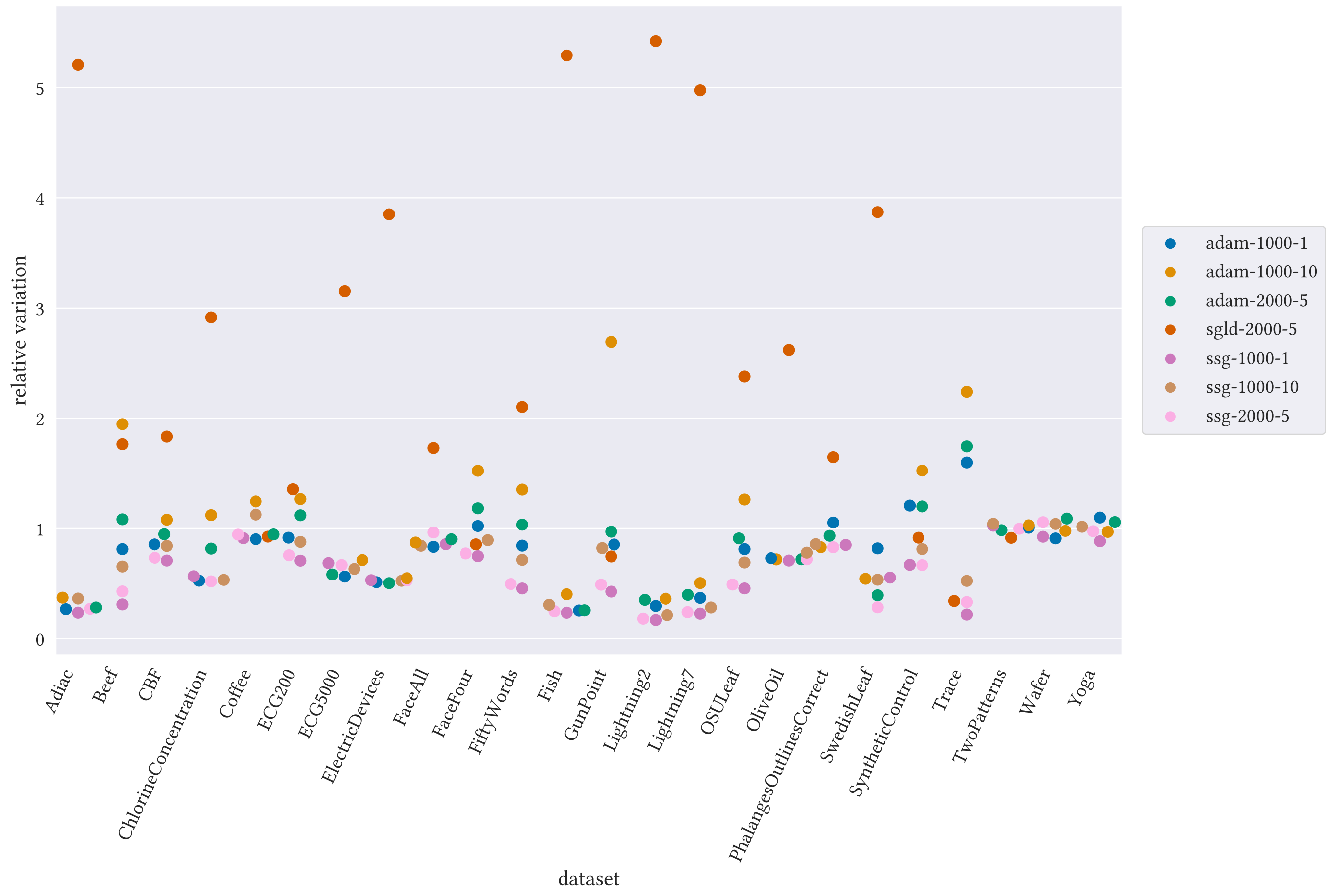

Figure 1: Plot of the experimental results

	ssg-1000-1	ssg-1000-10	ssg-2000-5
ssg-1000-1	0.00	20.83	29.17
ssg-1000-10	79.17	0.00	83.33
ssg-2000-5	70.83	16.67	0.00

	sadam-1000-1	sadam-1000-10	sadam-2000-5
sadam-1000-1	0.00	16.67	29.17
sadam-1000-10	83.33	0.00	79.17
sadam-2000-5	70.83	20.83	0.00

Table 4: Comparison of different configurations for SSG and SAdam (win percentage of column vs. row)

dataset	optimizer	sadam-1000-1	sadam-1000-10	sadam-2000-5	ssgld-2000-5	ssg-1000-1	ssg-1000-10	ssg-2000-5
Adiac	mean	0.6334	0.8810	0.6702	12.3286	0.5602	0.8615	0.6418
	std	0.0171	0.2260	0.0412	10.4651	0.0246	0.1986	0.0238
Beef	mean	42.7037	102.3383	56.9683	92.7796	16.3401	34.4340	22.5390
	std	13.2397	102.4333	42.4478	61.4644	0.6709	5.4278	3.3410
CBF	mean	22.1207	27.9342	24.5138	47.4244	18.3451	21.7667	19.0013
	std	1.7847	4.3366	3.3700	44.0707	0.2665	1.3357	0.4207
ChlorineConcentration	mean	26.0803	55.5761	40.4839	144.4973	28.0808	26.4369	25.7853
	std	3.1357	108.7362	72.7179	92.4818	11.0282	3.2218	2.8286
Coffee	mean	0.6935	0.9575	0.7270	0.7114	0.7007	0.8658	0.7256
	std	0.0235	0.2114	0.0715	0.0182	0.0380	0.0767	0.0241
ECG200	mean	8.9640	12.3988	10.9621	13.2587	6.9289	8.5890	7.4080
	std	2.3594	5.2450	4.3603	5.4638	0.4872	0.6904	0.4297
ECG5000	mean	44.9040	56.8447	46.4531	251.2325	54.6904	50.4415	53.2723
	std	12.0342	17.3523	11.6788	178.2748	24.6783	15.4601	25.2175
ElectricDevices	mean	90.1589	96.5995	88.6735	677.5335	93.4173	92.4552	93.0169
	std	15.4942	23.8630	14.7059	717.2856	22.2824	15.5279	14.4031
FaceAll	mean	48.9950	51.2871	53.0391	101.7478	50.3958	49.5862	56.6840
	std	17.8689	18.4973	16.2786	144.8582	14.5343	15.9082	23.2956
FaceFour	mean	48.8731	72.8313	56.5621	40.8919	35.7737	42.7082	36.9710
	std	8.3356	19.6718	13.8876	6.4577	1.3824	2.6172	1.2195
FiftyWords	mean	34.2455	54.8822	41.9959	85.3197	18.4811	29.0326	20.1219
	std	6.0018	12.8661	7.0771	149.2806	0.4928	5.2793	1.7285
Fish	mean	1.4182	2.2315	1.4285	29.3411	1.3095	1.7032	1.3834
	std	0.1053	1.3634	0.1140	27.0728	0.0207	0.2587	0.0944
GunPoint	mean	4.9057	15.4650	5.5753	4.2851	2.4510	4.7181	2.8126
	std	2.3145	8.1713	1.2471	3.2860	0.1822	0.7390	0.3964
Lightning2	mean	140.3098	171.3626	166.4420	2564.6851	80.4868	101.5192	86.2644
	std	45.1561	47.5282	86.0530	314.8979	3.6153	4.7540	3.9065
Lightning7	mean	80.1243	109.2765	86.0591	1078.0485	49.3173	61.3128	52.3020
	std	22.7166	44.0387	25.2270	158.5487	0.8373	2.8649	2.0022
OSULeaf	mean	50.7832	78.9375	56.8700	148.6253	28.4953	43.2393	30.6916
	std	10.7321	28.8593	9.4726	160.7086	0.4683	3.1897	0.8292
OliveOil	mean	0.0316	0.0312	0.0312	0.1135	0.0307	0.0338	0.0312
	std	0.0016	0.0020	0.0017	0.0136	0.0007	0.0042	0.0014
PhalangesOutlinesCorrect	mean	3.4023	2.6783	3.0136	5.3176	2.7450	2.7676	2.6783
	std	2.8618	1.2064	1.1619	9.8518	1.2131	1.1893	1.2064
SwedishLeaf	mean	12.1891	8.0923	5.8527	57.6055	8.2508	7.9644	4.2354
	std	11.2548	5.1090	1.8177	58.1228	4.4727	4.8908	0.1857
SyntheticControl	mean	40.2598	50.8333	40.0025	30.5153	22.3392	27.0704	22.2573
	std	7.7732	11.6128	7.8746	10.6911	0.4347	1.2053	0.2417
Trace	mean	155.1166	217.2904	169.2841	33.1536	21.3048	50.8862	32.1813
	std	19.5812	27.1548	28.9826	16.9852	6.6516	33.1358	21.6413
TwoPatterns	mean	42.5022	43.4263	41.5476	nan	43.2842	44.0149	42.1114
	std	3.9381	4.3847	3.5361	nan	4.3643	7.2100	3.9436
Wafer	mean	111.2541	119.6274	133.3946	nan	113.1229	127.3713	129.3313
	std	41.1354	41.0689	43.7954	nan	38.8414	43.0932	48.1105
Yoga	mean	79.1960	69.7777	76.2090	nan	63.6301	73.1535	70.2342
	std	34.8982	33.7083	31.0540	nan	19.8401	32.6218	29.4631

Table 5: Results (’mean’ denotes the mean over all iterations, ’std’ denotes the standard deviation)