The Right Learning Rate at the Right Time

Anonymous submission

Abstract

The selection of an appropriate optimizer plays a crucial role in deep learning, significantly influencing model performance. In the context of batch learning, where all data is available at a time, extensive research has been conducted to understand the implications of different optimizer choices and various optimization techniques for deep learning architectures have emerged. However, when it comes to online learning, where models need to adapt to evolving data streams, the exploration of optimizer choices is still relatively limited. This paper focuses on bridging the gap in knowledge by investigating how the choice of optimizer changes from batch learning to online learning scenarios. Our study conducts an in-depth analysis on the impact of optimizer, learning rate and batch size on the (i) predictive performance as well as the capability of adapting to changes on the underlying data pattern that might occur over time within the evolving data stream. Additionally, we explore practical choices for gradient-based online training of deep architectures, with a specific emphasis on adaptive methods. These methods dynamically adjust the learning rate based on gradient characteristics, offering potential advantages in online learning scenarios.

de on
ch term
se. On-
Learning
Stream
ning
Online
ds bet-
ut might
onfusing
e SGD
chnically
ys an
ne algo.

more

Introduction

Deep learning models have demonstrated exceptional performance in various domains, with the choice of optimizer playing a crucial role in achieving outstanding results. In the context of batch learning, where all data is available simultaneously, extensive research has been conducted to explore different optimizer choices and optimization techniques for deep learning architectures. Numerous methods have emerged to effectively update the weights of these architectures. However, the investigation of optimizer choices in online learning, where models must adapt to evolving data streams, remains relatively limited.

This paper aims to bridge this knowledge gap by investigating how the choice of optimizer changes when transitioning from batch learning to online learning scenarios. Specifically, we address the following research questions:

- How does the choice for the optimizer change from batch to online learning?
- What are practical choices for gradient-based online training of deep architectures in online learning?
- Are adaptive optimization methods better suited in Online Deep Learning?

Put LR tu
ing graph
here?

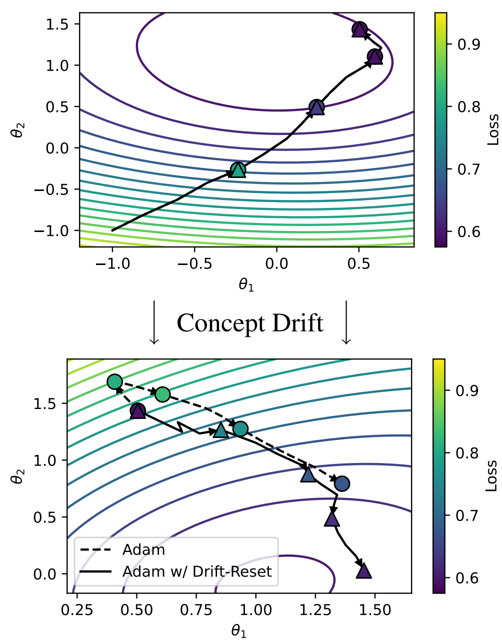

Figure 1: Parameter trajectory of Adam (Kingma and Ba 2017) with or without adaptation to concept drift on synthetic data stream with abrupt concept drift. Marker colors depict the expected prequential loss over the last 16 data instances.

For the first research question, we explore how the selection of an optimizer differs when moving from the traditional batch learning setting to the dynamic online learning scenario. We examine the suitability of various optimizer choices in online learning and their impact on model performance.

The second research questions investigates practical choices for gradient-based online training of deep architectures. We analyze different optimization techniques and explore their effectiveness in adapting to evolving data streams while maintaining model performance. The third research question focuses on the performance of adaptive methods in online deep learning scenarios. These methods dynamically adjust the learning rate based on gradient characteristics, allowing models to adapt more effectively to changing data patterns. We compare the performance of adaptive methods against other optimization approaches to determine their suitability for online deep learning tasks. Through our in-depth analysis and experimentation, we aim to enhance our understanding of optimizer choices in online deep learning. By shedding light on the impact of optimizers, learning rates, and batch sizes, and comparing the effectiveness of adaptive methods, we aim to enable researchers and practitioners to make informed decisions when selecting optimization techniques for real-time learning tasks.

Research Questions

- How does the choice for the optimizer change from batch to online learning? -¿ Batch learning: goal maximize batch size for best performance -¿ suitable for online learning? -¿ Learning Rate -¿ In-depth analysis on the influence of the learning rate and batch size on the effectiveness of online stochastic gradient descent -¿ Grafik SGD,
- What are practical choices for gradient-based online training of deep architectures in online learning? -¿ Description adaptive methods
- Are adaptive methods better suited in Online Deep Learning? -¿ Table with results

Motivation:

- Learning rate is one of the most important hyperparameters when it comes to training deep learning models.
- A small learning rate causes slow convergence, while an excessive learning rate causes instability (observable in the form of oscillations) or even increases in the training objective.
- Bengio (2012) goes as far as to say that out of all possible hyperparameters, if any, the learning rate is the one to be tuned.
- This applies arguably even more so to online learning. Unlike in batch learning, for instance, repeating the entire process if training diverges is infeasible, due to the restriction that previously observed data samples can, if at all, only be stored in small numbers.
- Furthermore, the trained model must be ready to predict at any given moment meaning that an inadequate learning rate has an immediate impact on the quality of generated predictions.
- In batch learning it is still common practice to determine the learning rate by running multiple training runs, beginning with a high learning rate and restarting the process with a lower learning rate every time the objective value diverges (Bengio 2012).

Contributions:

- In-depth analysis on the influence of the learning rate and batch size on the effectiveness of online stochastic gradient descent
- Experimental comparison of different SGD-based optimizers
- Investigation into possible approaches for determining an appropriate learning rate

Learning Rate Scheduling

In the following, we will briefly outline the most important differences between the influence of the learning rate of first order gradient-based optimization methods in streaming- and in conventional batch learning environments.

First order gradient-based optimization approaches like stochastic gradient descent and its derivatives aim to iteratively minimize the error of a DL model using only first order gradient information at each step t. We denote the gradient of the prediction error for a mini-batch of training samples $y_t, X_t \sim p_t$ with respect to model-parameters θ as

$$g_t(\theta) = \nabla_\theta \mathcal{L}(y_t, f(X_t; \theta)), \tag{1}$$

where $\mathcal{L}$ represents a loss function (e.g., cross-entropy for classification- or mean squared error for regression tasks). Using this notation, the update performed by SGD with a constant learning rate η at each iteration t is given by

$$\theta_{t+1} = \theta_t - \eta \cdot g_t(\theta_t). \tag{2}$$

Many previous works deal with the intricacies of SGD in the context of training deep architectures in a batch learning setting (see Bengio (2012); Bottou (2012); Goodfellow, Bengio, and Courville (2016)). The primary trade-off identified by these works when it comes to the selection of an appropriate learning rate is that between the speed of convergence and the amount of stochasticity. While increasing the learning rate speeds up convergence, it also increases stochasticity and therefore leads to the divergence of the training criterion beyond a certain threshold. Smith and Le (2018) for instance, found that when modelling SGD as a stochastic differential equation, the "noise scale" is directly tied to η (Smith and Le 2018). In biological terms, increasing the learning rate increases plasticity, whereas decreasing it increases stability.

To get the best out of large and small values for η, a learning rate schedule $(\eta_1, \ldots, \eta_T)$, which assigns a learning rate to each update step $t \in 1, \ldots, T$ can be used. It is for instance common to set a high learning rate initially and decrease it throughout the training process. This ensures fast convergence and a higher level of noise at the start of training, which has been claimed to help SGD skip over sharp minima with poor generalization (Hochreiter and Schmidhuber 1997; Chaudhari et al. 2017), while mitigating jumping around potential minima at later stages. Some have likened this procedure to simulated annealing, which shifts its focus from exploration at high temperatures to exploitation once temperatures have sufficiently decreased (Smith et al. 2018).

A simple way to achieve such a schedule is to exponentially decay η after each update by multiplying it with a factor $\gamma < 1$, although some practitioners have come to prefer schedules with sharper drop-offs like step schedules that decrease η by a larger margin after a certain number of updates (Smith et al. 2018). Other popular options include cyclic learning rate schedules which oscillate η between a base- and a maximum or minimum value over a predefined interval. Some studies (Smith 2017; Smith and Topin 2018) have found cyclic schedules to significantly speed up the convergence of neural networks even when compared to adaptive techniques like Adam (Kingma and Ba 2017). Wu et al. (2019) provide a detailed analysis on the effect of learning rate policies like the aforementioned ones.

In contrast to the field of conventional batch learning, the impact of the learning rate in stream-based deep learning is a lesser studied issue. According to Bifet et al. (2010) a machine learning model operating in such an environment must be able to

R1: process a single instance at a time,
R2: process each instance in a limited amount of time,
R3: use a limited amount of memory,
R4: predict at any time,
R5: adapt to changes in the data distribution.

These requirements lead to significant differences with respect to the problem of learning rate optimization compared to batch learning. Nevertheless, only few studies on the impact and tuning of the learning rate in stream-based deep learning exist.

In batch learning, the task of finding an optimal schedule for η can be defined as

$$\begin{aligned} &\min_{\eta_0,\dots,\eta_T} \sum_{i=1}^{V} \mathcal{L}(y_i, f(X_i;\theta_T)) \\ &\text{s.t.} \quad X_i, y_i \sim p^{(v)} \qquad \forall i \in 1,\dots,V, \end{aligned} \tag{3}$$

where $p^{(v)}$ is a distribution of validation data, usually made up of a dataset split off from the training dataset. Verbally, learning rate optimization in batch learning aims to find a schedule $(\eta_0,\dots,\eta_T)$ leading to parameter values θ_T at the end of a training run that in turn minimize the prediction error for validation data.

Under the requirements described above, however, we must define the task differently as

$$\begin{aligned} &\min_{\eta_0,\dots,\eta_T} \sum_{t=0}^{T} \mathcal{L}(y_t, f(X_t;\theta_{t-1})) \\ &\text{s.t.} \quad X_t, y_t \sim p_t \qquad \forall t \in 1,\dots,T. \end{aligned} \tag{4}$$

Compared to Problem (3), the most apparent difference is that there is no separate validation data. Instead, due to Requirement , the goal is to minimize $\mathcal{L}$ with respect to the next instances X_{t+1}, y_{t+1} at each timestep t.

This means that in contrast to Problem (3), not only the final parameters θ_T but every parameter configuration θ_t in the entire trajectory contributes equally to the objective. Therefore, speed of convergence is of much larger importance in the streaming setting, whereas the performance of the final parameters θ_T has relatively little impact. Since memory is limited (Requirement), it is also not possible to continue training on previously observed data as long as $\mathcal{L}$ decreases, which puts an even greater emphasis on quick adaptation. At the same time, a larger learning rate causing temporary loss increases, due to skipping over sharp minima can be suboptimal with respect to Problem 4 even if it eventually yields a lower loss. Another difference to conventional batch learning is that the distribution p_t of the data stream might, and in practice most likely will, be subjected to change in the form of concept drift over time. Under such circumstances, the optimal parameter values θ^* move throughout the progression of the stream increasing the distance to the model parameters.

Since the theoretically optimal learning rate η^* is proportional to the quadratic distance between initial and optimal parameters $||\theta_1 - \theta^*||^2$ (albeit under some constraints like the absence of noise) (Carmon and Hinder 2023), it should be larger when concept drift occurs.

Based on this notion, Kuncheva and Plumpton (2008) introduced an adaptive schedule that updates the learning rate using

$$\eta_{t+1} = \eta_t^{1+(\mathcal{L}_t - \bar{\mathcal{L}}_{t-1})}, \tag{5}$$

where $\mathcal{L}_t$ represents the loss for the current sample, and $\bar{\mathcal{L}}_{t-1}$ a rolling mean of past losses. By doing so, Kuncheva and Plumpton (2008) increases in loss lead to increases in the learning rate and vice versa. While this approach seems intuitively sound, it bears a high risk of η increasing indefinitely, since increases in loss caused by an excessive learning rate would lead to a feedback loop.

We therefore propose a simple adaptation to popular decaying learning rate schedules that operates in a fixed value range and increases the model's plasticity when needed by resetting η to its original value if a concept drift has occurred. To this end we apply an ADWIN drift detector (Bifet and Gavaldà 2007) to the underlying model's prequential loss to reduce the drift detection task to a computationally less expensive univariate problem. While our approach does not allow for a continuous adaptation like the technique by Kuncheva and Plumpton (2008) and might therefore be less suited for more subtle drift, it offers more reliability since the initial learning rate cannot be exceeded.

Talk abou drift-probabilit dependent setting wi P-KSWIN

Concept drift also complicates the tuning of η, since even if data is available beforehand drift would eventually cause the stream to diverge from the distribution of data used for tuning. This effect, combined with the previously described differences in the evaluation scheme can cause conventional learning rate tuning to produce unsuitable results for stream-based learning. We therefore propose a slightly different online learning specific tuning approach, that aims to approximately solve Problem 4.

To emulate the targeted data stream we continually draw samples with replacement from the tuning data in a bootstrapping procedure instead of training on all data for multiple epochs. By doing so we aim to increase data variability, and therefore the resemblance to an actual data stream with random distributional shifts. We then optimize η with respect to the mean prequential performance over the emulated stream instead of the performance on a validation set. For this purpose we use a basic grid-search as is customary in batch learning (Defazio and Mishchenko 2023). We provide a detailed experimental evaluation of our approach in Section .

Optimizer	Complexity	Space	Param. specific
DAdapt	$\mathcal{O}(6D)$	$\mathcal{O}(2D)$	✗
DoG	$\mathcal{O}(5D)$	$\mathcal{O}(1D)$	✗
Mechanic	$\mathcal{O}(4D)$	$\mathcal{O}(1D)$	✗
WNGrad	$\mathcal{O}(2D)$	$\mathcal{O}(0)$	✗
SGDHD	$\mathcal{O}4D$	$mathcalO$	✗
COCOB	$\mathcal{O}(14D)$	$\mathcal{O}(4D)$	✓
vSGD	$\mathcal{O}(21D)$[1]	$\mathcal{O}(4D)$	✓

Table 1: Overview of additional time- and space-complexity of parameter-free first-order optimizers compared to basic SGD. Values are given in big O notation with respect to the number of model parameters D.

Parameter-Free Optimizers

While determining the learning rate through a separate tuning phase with parameter searches like grid- or random-search is still the de facto standard in deep learning (Defazio and Mishchenko 2023), this approach causes significant computational overhead. To avoid this overhead, several previous works have developed parameter-free variants of SGD (see Table 1) that estimate the optimal step size online as training progresses, eliminating the learning rate entirely.

In one of the earlier works, Schaul, Zhang, and LeCun (2013) proposed *vSGD*, which uses first and second order moments of the gradients as well as local curvature information (Schaul, Zhang, and LeCun 2013) to estimate η. The authors obtain the latter by estimating positive diagonal entries of the Hessian with respect to the parameters through a backpropagation formula (Schaul, Zhang, and LeCun 2013). Even though Schaul, Zhang, and LeCun (2013) demonstrate *vSGD's* robustness to non-stationary data distributions, it has, to the best of our knowledge, not been widely adopted in the online learning space.

not eval-
vSGD
e its old
there is
mplemen-
n yet. Is
valid?

Instead of using curvature information for adapting η, the *COCOB* optimizer proposed by Orabona and Tommasi (2017)

Furthermore, several studies developed paramater-free optimizers for specific areas of application such as time series forecasting (Miyaguchi and Kajino 2019; Fekri et al. 2021; Zhang 2021), federated learning (Canonaco et al. 2021) and recommender systems (Ferreira Jose, Enembreck, and Paul Barddal 2020). Due to our focus for the present work being general data stream applications, we did not further investigate these techniques either.

Besides Schaul, Zhang, and LeCun (2013) none of these optimizers target batch learning scenarios and therefore do not explicitly consider concept drift, which raises the question of whether they are suitable for stream-based learning. Previous work on parameter-free optimization of online deep learning models is rather limited, and mostly focused on specific applications.

[1]Complexity for feed-forward neural networks. Since *vSGD* requires additional backpropagation steps, its complexity is architecture dependent.

Adaptive Optimizers

In the case of adaptive approaches like SGD with momentum (Rumelhart, Hinton, and Williams 1986), AdaGrad (Duchi, Hazan, and Singer 2011) or Adam (Kingma and Ba 2017) the learning rate also varies with respect to individual model parameters, in which case

$$\eta_t \in \mathbb{R}^D \qquad \forall t \in 0, \dots, T. \tag{6}$$

For SGD with momentum,

$$\eta_t = \alpha \cdot \sum_{i=1}^{t} \beta^{t-i} \cdot \frac{g_i}{g_t} \tag{7}$$

- Adaptive learning rate optimizers for batch learning (Kingma and Ba 2017; Zeiler 2012; Duchi, Hazan, and Singer 2011; Tieleman and Hinton 2012) - Learning-Rate free optimizers (Wu, Ward, and Bottou 2020; Schaul, Zhang, and LeCun 2013; Orabona and Tommasi 2017; Miyaguchi and Kajino 2019; van Erven and Koolen 2016; Baydin et al. 2018)

Experiments

Type	Data Stream	Samples	Features	Classes
Synth.	RBF abrupt	20000	20	5
	RBF incremental	20000	20	5
Real	Insects abrupt	52848	33	6
	Insects incremental	57018	33	6
	Insects incr.-grad.	24150	33	6
	Covertype[2]	100000	54	7
	Electricity	45312	8	2

Table 2: Datasets used for experimental evaluations.

We ran prequential evaluations using basic SGD with variable batch sizes and learning rates for synthetic data streams with and without incremental concept drift, the results of which are displayed in Figure 5. For static data, the average prequential accuracy over the entire stream gradually improves when moving up from an inadequately low learning rate until a certain point where training begins to diverge and performance consequently crashes. Based on our results, there seems to be an inverse relationship between batch size and both the optimal learning rate and the optimal accuracy, with larger batch sizes seemingly increasing the risk of divergence.

$$\tag{8}$$

This effect is much stronger in the presence of concept drift as the results for RBF Incremental show.

¡-could be explained by the fact that the presence of concept drift exacerbates the gradient stochasticity caused by the delay between observation and learning of samples.

[2]We used the first 100k from a total of 581k examples only.

Optimizer	Schedule	RBF abrupt	RBF incr.	Covertype	Insects abrupt	Electricity
SGD	Fixed	93.70±.90	69.33±1.36	83.08±.18	71.12±.08	73.12±.42
	Exp.	94.00±.52	69.24±1.06	82.65±.19	71.32±.19	73.06±.42
	Exp. Reset	94.28±.37	69.76±.92	82.70±.27	71.27±.14	73.05±.45
	Step	94.00±.68	69.16±.99	82.74±.10	71.39±.17	72.96±.48
	Step Reset	94.04±.70	69.75±.81	83.03±.13	71.19±.14	73.18±.50
	Cyclic	<u>94.45±.25</u>	<u>73.72±1.16</u>	**83.40±.21**	71.41±.20	67.80±1.03
	Cyclic Reset	**94.50±.21**	**73.78±1.15**	<u>83.33±.13</u>	71.39±.15	67.83±1.00
Adam	Fixed	92.77±.41	66.46±4.39	78.85±.22	**75.08±.13**	69.23±.41
	Exp.	92.17±.82	63.91±3.52	78.53±.27	74.88±.15	69.33±.40
	Exp. Reset	92.83±1.19	62.03±2.59	77.05±.08	72.04±.49	68.31±.40
AdaGrad	Fixed	91.34±.83	50.39±3.60	81.07±.22	74.31±.34	76.64±1.92
SGDHD	Fixed	91.03±.45	63.38±1.55	82.33±.12	67.35±.16	73.10±.10
COCOB	Fixed	93.40±.38	63.52±2.70	82.27±.46	74.75±.11	**84.30±.56**
WNGrad	Fixed	87.23±1.24	44.79±.76	76.95±.15	66.14±.15	70.74±.59
DAdaptSGD	Fixed	74.91±4.22	45.47±2.75	76.69±.79	50.05±11.26	66.03±1.75
DoG	Fixed	92.73±.59	<u>73.17±2.72</u>	83.07±.64	70.59±.26	71.53±.70

Table 3: Average prequential accuracy [%] for the three best learning rates.

Resetting Approach	RBF abrupt	RBF incr.	Covertype	Insects abrupt	Electricity
ADWIN (Two Tailed)	**94.28±.37**	<u>69.76±.92</u>	82.70±.27	<u>71.27±.14</u>	<u>73.05±.45</u>
ADWIN (One Tailed)	<u>94.25±.38</u>	<u>70.22±2.58</u>	82.64±.20	<u>71.25±.15</u>	<u>73.00±.50</u>
ADWIN Weight Reset	71.79±.73	65.07±.31	82.54±.16	50.97±.36	70.19±1.00
KSWIN	<u>94.23±.55</u>	<u>70.10±1.98</u>	**83.01±.06**	**71.38±.16**	<u>73.13±.31</u>
P-KSWIN	93.86±.48	**70.71±1.38**	**83.01±.18**	<u>71.25±.15</u>	**73.26±.43**

Table 4: Average prequential accuracy [%] for the three best learning rates.

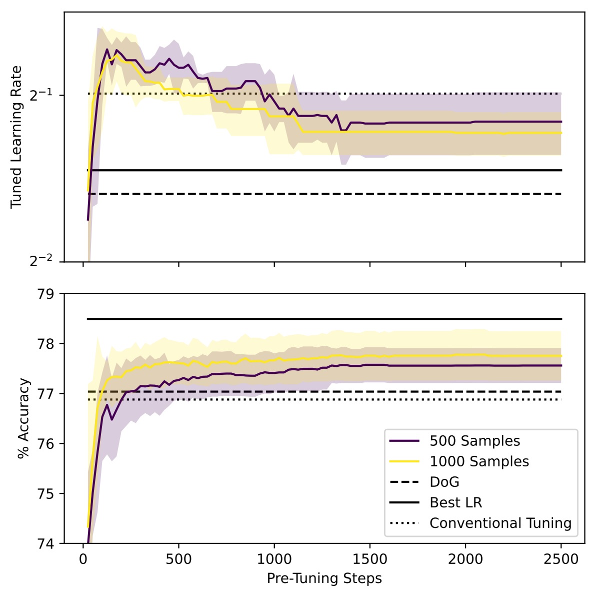

Figure 2: Pre-tuned LR (LR that maximizes accuracy on pre-tuning data) and resulting accuracy on data streams when using SGD and an exponential learning rate schedule with 500 or 1000 separate tuning samples. Results are averaged over all real-world datasets. The shaded area represents the 1σ-interval.

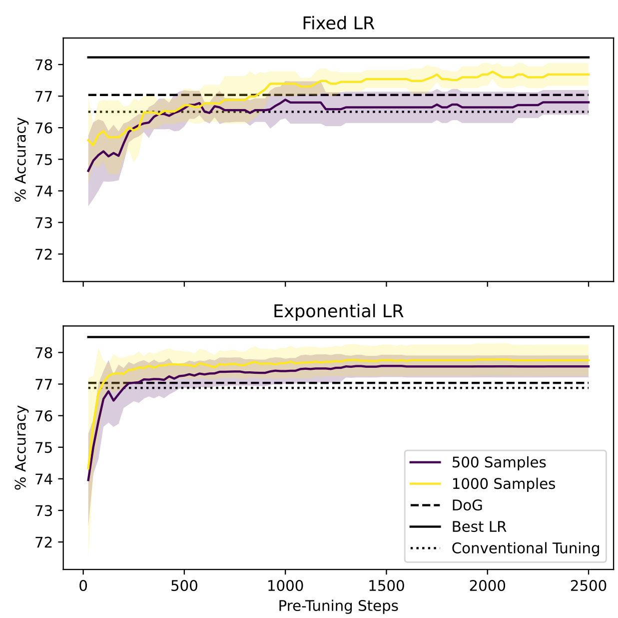

Figure 3: Accuracy achieved by pre-tuning on 500 or 1000 samples when using SGD with a fixed LR schedule (top) or an exponential schedule (bottom), averaged over all real-world datasets. The shaded area represents the 1σ-interval.

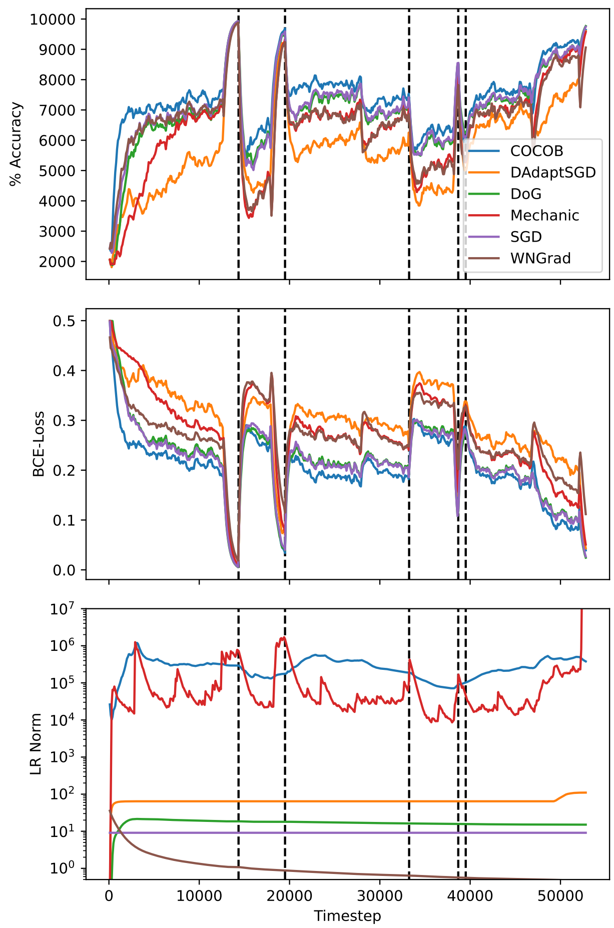

Figure 4: Prequential accuracy, binary cross-entropy loss and LR norms $||\eta_t||$ over time for various optimization algorithms on Insects abrupt. Each dashed vertical line represents a concept drift. Lines are exponentially smoothed with a factor of 0.8.

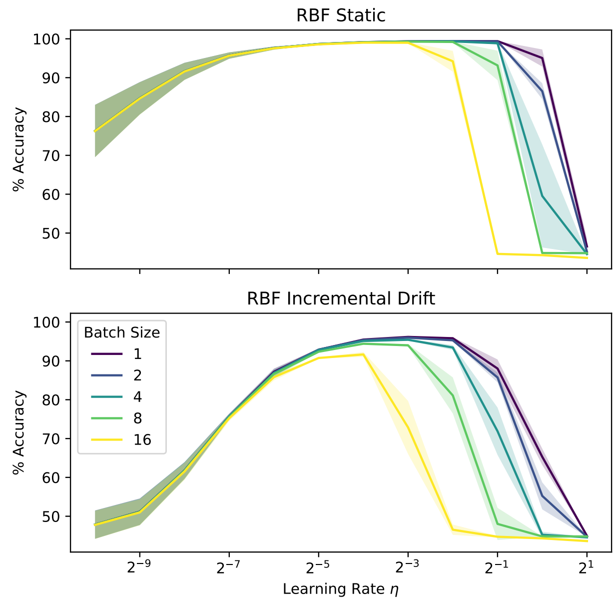

Figure 5: Average accuracy over static and incrementally drifting synthetic data streams in relation to SGD mini-batch size and learning rate η. Shaded areas mark the 1σ interval.

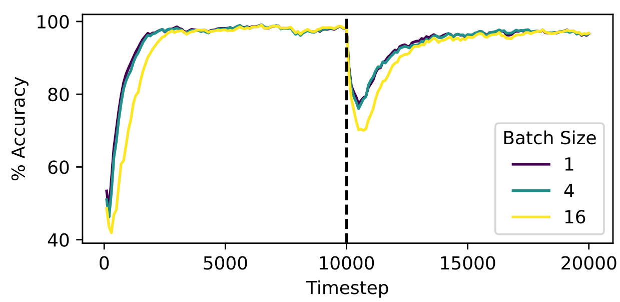

Figure 6: Prequential accuracy for different batch sizes on RBF abrupt data stream.

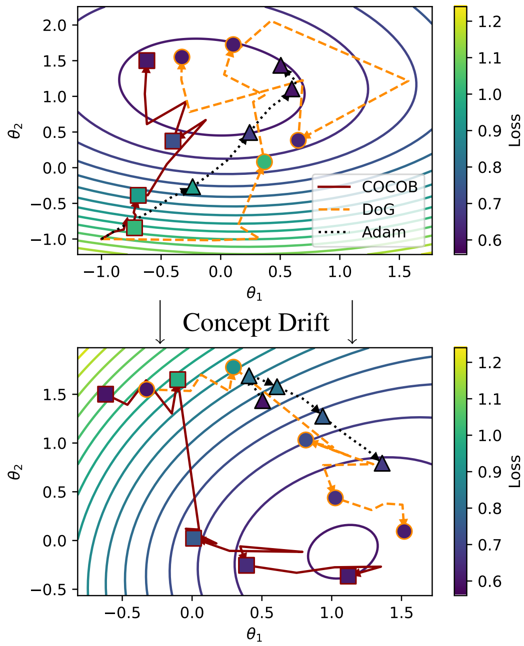

Figure 7: Parameter trajectory of COCOB (Orabona and Tommasi 2017), DoG (Ivgi, Hinder, and Carmon 2023) and Adam (Kingma and Ba 2017) on synthetic data stream with abrupt concept drift. Marker colors depict the expected prequential loss over the last 16 data instances.

Conclusion

References

Baydin, A. G.; Cornish, R.; Rubio, D. M.; Schmidt, M.; and Wood, F. 2018. Online Learning Rate Adaptation with Hypergradient Descent. In *ICLR Proceedings*.

Bengio, Y. 2012. Practical Recommendations for Gradient-Based Training of Deep Architectures. arxiv:1206.5533.

Bifet, A.; and Gavaldà, R. 2007. Learning from Time-Changing Data with Adaptive Windowing. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, 443–448. Society for Industrial and Applied Mathematics. ISBN 978-0-89871-630-6 978-1-61197-277-1.

Bifet, A.; Holmes, G.; Kirkby, R.; and Pfahringer, B. 2010. MOA: Massive Online Analysis. *Journal of Machine Learning Research*, 11.

Bottou, L. 2012. Stochastic Gradient Descent Tricks. In Montavon, G.; Orr, G. B.; and Müller, K.-R., eds., *Neural Networks: Tricks of the Trade: Second Edition*, Lecture Notes in Computer Science, 421–436. Berlin, Heidelberg: Springer. ISBN 978-3-642-35289-8.

Canonaco, G.; Bergamasco, A.; Mongelluzzo, A.; and Roveri, M. 2021. Adaptive Federated Learning in Presence of Concept Drift. In *2021 International Joint Conference on Neural Networks (IJCNN)*, 1–7.

Carmon, Y.; and Hinder, O. 2023. Making SGD Parameter-Free. arxiv:2205.02160.

Chaudhari, P.; Choromanska, A.; Soatto, S.; LeCun, Y.; Baldassi, C.; Borgs, C.; Chayes, J.; Sagun, L.; and Zecchina, R. 2017. Entropy-SGD: Biasing Gradient Descent Into Wide Valleys. arxiv:1611.01838.

Defazio, A.; and Mishchenko, K. 2023. Learning-Rate-Free Learning by D-Adaptation. arxiv:2301.07733.

Duchi, J. C.; Hazan, E.; and Singer, Y. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 12(61): 2121–2159.

Fekri, M. N.; Patel, H.; Grolinger, K.; and Sharma, V. 2021. Deep Learning for Load Forecasting with Smart Meter Data: Online Adaptive Recurrent Neural Network. *Applied Energy*, 282: 116177.

Ferreira Jose, E.; Enembreck, F.; and Paul Barddal, J. 2020. ADADRIFT: An Adaptive Learning Technique for Long-history Stream-based Recommender Systems. In *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2593–2600. Toronto, ON, Canada: IEEE. ISBN 978-1-72818-526-2.

Goodfellow, I.; Bengio, Y.; and Courville, A. 2016. *Deep Learning*. MIT Press.

Hochreiter, S.; and Schmidhuber, J. 1997. Flat Minima. *Neural Computation*, 9(1): 1–42.

Ivgi, M.; Hinder, O.; and Carmon, Y. 2023. DoG Is SGD's Best Friend: A Parameter-Free Dynamic Step Size Schedule. arxiv:2302.12022.

Kingma, D. P.; and Ba, J. 2017. Adam: A Method for Stochastic Optimization. arxiv:1412.6980.

Kuncheva, L. I.; and Plumpton, C. O. 2008. Adaptive Learning Rate for Online Linear Discriminant Classifiers. In Da Vitoria Lobo, N.; Kasparis, T.; Roli, F.; Kwok, J. T.; Georgiopoulos, M.; Anagnostopoulos, G. C.; and Loog, M., eds., *Structural, Syntactic, and Statistical Pattern Recognition*, volume 5342, 510–519. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-540-89688-3 978-3-540-89689-0.

Miyaguchi, K.; and Kajino, H. 2019. Cogra: Concept-Drift-Aware Stochastic Gradient Descent for Time-Series Forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01): 4594–4601.

Orabona, F.; and Tommasi, T. 2017. Training Deep Networks without Learning Rates Through Coin Betting. In *NIPS*.

Rumelhart, D. E.; Hinton, G. E.; and Williams, R. J. 1986. Learning Representations by Back-Propagating Errors. *Nature*, 323(6088): 533–536.

Schaul, T.; Zhang, S.; and LeCun, Y. 2013. No More Pesky Learning Rates. arxiv:1206.1106.

Smith, L. N. 2017. Cyclical Learning Rates for Training Neural Networks. arxiv:1506.01186.

Smith, L. N.; and Topin, N. 2018. Super-Convergence: Very Fast Training of Neural Networks Using Large Learning Rates. arxiv:1708.07120.

Smith, S. L.; Kindermans, P.-J.; Ying, C.; and Le, Q. V. 2018. Don't Decay the Learning Rate, Increase the Batch Size. arxiv:1711.00489.

Smith, S. L.; and Le, Q. V. 2018. A Bayesian Perspective on Generalization and Stochastic Gradient Descent. arxiv:1710.06451.

Tieleman, T.; and Hinton, G. 2012. Lecture 6.5-Rmsprop: Divide the Gradient by a Running Average of Its Recent Magnitude. In *COURSERA: Neural Networks for Machine Learning*. Coursera.

van Erven, T.; and Koolen, W. M. 2016. MetaGrad: Multiple Learning Rates in Online Learning. arxiv:1604.08740.

Wu, X.; Ward, R.; and Bottou, L. 2020. WNGrad: Learn the Learning Rate in Gradient Descent. arxiv:1803.02865.

Wu, Y.; Liu, L.; Bae, J.; Chow, K.-H.; Iyengar, A.; Pu, C.; Wei, W.; Yu, L.; and Zhang, Q. 2019. Demystifying Learning Rate Policies for High Accuracy Training of Deep Neural Networks. arxiv:1908.06477.

Zeiler, M. D. 2012. ADADELTA: An Adaptive Learning Rate Method. arxiv:1212.5701.

Zhang, W. 2021. POLA: Online Time Series Prediction by Adaptive Learning Rates. arxiv:2102.08907.