OPT2023: 15th Annual Workshop on Optimization for Machine Learning

Non-Uniform Sampling and Adaptive Optimizers in Deep Learning

Thibault Lahire THIBAULT.LAHIRE@DASSAULT-AVIATION.COM
Dassault Aviation, France

Abstract

Stochastic gradient descent samples uniformly the training set to build an unbiased gradient estimate with a limited number of samples. However, at a given step of the training process, some data are more helpful than others to continue learning. Importance sampling for training deep neural networks has been widely studied to propose sampling schemes yielding better performance than the uniform sampling scheme. After recalling the theory of importance sampling for deep learning, this paper focuses on the interplay between the sampling scheme and the optimizer used. We show that the sampling proportional to the per-sample gradient norms is not optimal for adaptive optimizers, although it is the case for stochastic gradient descent in its standard form. This implies that new sampling schemes have to be designed with respect to the optimizer used. Thus, using approximations of the per-sample gradient norms scheme with adaptive optimizers is likely to yield unsatisfying results.

Keywords: Importance Sampling, Stochastic Gradient Descent, Deep Learning, Optimizers

1. Introduction

Deep neural networks tend to become very large, with a high number of weights to optimize. Training deep neural networks is usually done with Stochastic Gradient Descent based algorithms [17, SGD], such as RMSProp or ADAM [10]. With such large architectures, training deep neural networks is computationally costly, since the cost of computing gradients is proportional to the number of weights. To alleviate this computational cost, one possibility is to take better SGD steps, so that the optimum is reached in fewer SGD iterations.

SGD builds at each iteration an unbiased estimate of the empirical gradient by sampling uniformly the training set. However, at a given step of the learning process, some parts of the training set might be well handled by the neural network, whereas the error it makes on other parts is large. Some training data are more useful than others for training the neural network. Hence, sampling uniformly the training set at each step of the learning process to perform SGD might be sub-optimal.

Importance sampling [18] applied to SGD has already been explored and applied to a wide range of datasets in supervised learning [14, 23], as well as in reinforcement learning [19]. It gives sampling schemes for SGD which speed up convergence to the optimum in theory, by selecting the most helpful training data for the learning process. However, the optimal sampling scheme, yielding the highest convergence speed, is intractable in practice.

To the best of our knowledge, all the works taking the best of importance sampling for SGD steps use approximations of the optimal sampling scheme [9, 19]. Moreover, these approximated sampling schemes are interchangeably used with all common deep learning optimizers, such as RMSProp or ADAM [10]. We show that the optimal sampling scheme for RMSProp and ADAM is different from the one for standard SGD. This implies that new sampling schemes aiming at accelerating the learning process have to be designed with respect to the optimizer used. To the

best of our knowledge, this point has not yet been studied in the literature, and it has implications described in this work.

2. Background

2.1. Importance sampling in deep learning

Let $\Psi(\theta, \cdot)$ be any deep neural network parameterized by θ and ℓ be the loss to minimize during training. Over a training set of N items $(x_i, y_i)_{1 \leq i \leq N}$, the goal of training is to find: $\theta^* \in \arg\min_\theta \frac{1}{N} \sum_{i=1}^N \ell(\Psi(\theta, x_i), y_i)$.

Writing u the uniform probability distribution over the N items of the training set, i.e. $\forall i \in [1; N], u_i = 1/N$, the empirical gradient of the loss function defined above can be written as an expectation: $\frac{1}{N} \sum_{i=1}^N \nabla_\theta \ell(\Psi(\theta, x_i), y_i) = \sum_{i=1}^N u_i \nabla_\theta \ell(\Psi(\theta, x_i), y_i) = \mathbb{E}_{i \sim u}[\nabla_\theta \ell(\Psi(\theta, x_i), y_i)]$. Writing p any probability distribution over the training set, importance sampling can be used: $\mathbb{E}_{i \sim u}[\nabla_\theta \ell(\Psi(\theta, x_i), y_i)] = \mathbb{E}_{i \sim p}\left[\nabla_\theta \ell(\Psi(\theta, x_i), y_i) \frac{u_i}{p_i}\right] = \frac{1}{N} \mathbb{E}_{i \sim p}\left[\frac{1}{p_i} \nabla_\theta \ell(\Psi(\theta, x_i), y_i)\right]$.

All these expectations can be approximated with mean estimators. With B the mini-batch size, $B \ll N$, it yields: $\mathbb{E}_{i \sim p}\left[\frac{1}{p_i} \nabla_\theta \ell(\Psi(\theta, x_i), y_i)\right] \approx \frac{1}{B} \sum_{i=1}^B \frac{1}{p_i} \nabla_\theta \ell(\Psi(\theta, x_i), y_i)$, $i \sim p$. We introduce $G_i^{(t)} = w_i \nabla_\theta \ell(\Psi(\theta_t, x_i), y_i)$ with $w_i = 1/(N p_i)$ for any sampling scheme p such that $\forall i \in [1; N], \quad p_i > 0$. This quantity will be useful to study SGD based algorithms using sampling p, possibly non uniform. Note that, when $p = u$, $w_i = 1$.

Setting η as a constant learning rate, a standard (full) gradient descent update has the form: $\theta_{t+1} = \theta_t - \eta \mathbb{E}_{i \sim p}[G_i^{(t)}]$. This can be surprising at first sight since it seems to depend on the sampling scheme p used, whereas there is no sampling for the standard (full) gradient descent. It is indeed the case: $\mathbb{E}_{i \sim p}[G_i^{(t)}] = \sum_{i=1}^N p_i G_i^{(t)} = \frac{1}{N} \sum_{i=1}^N \nabla_\theta \ell(\Psi(\theta_t, x_i), y_i)$, and this notation gives consistency when writing the stochastic gradient descent update under sampling scheme p:

$$\theta_{t+1} = \theta_t - \eta \frac{1}{B} \sum_{i=1}^B G_i^{(t)} = \theta_t - \eta \frac{1}{B} \sum_{i=1}^B \frac{1}{N p_i} \nabla_\theta \ell(\Psi(\theta_t, x_i), y_i). \tag{1}$$

Following the notations of Katharopoulos and Fleuret [9], let us define the convergence speed S of SGD under a sampling scheme p as $S(p) = -\mathbb{E}_{i \sim p}\left[\|\theta_{t+1} - \theta^*\|_2^2 - \|\theta_t - \theta^*\|_2^2\right]$. We recall that a stochastic gradient descent update with $B = 1$ has the form $\theta_{t+1} = \theta_t - \eta G_i^{(t)}$, where $G_i^{(t)}$ is the gradient estimate built from sampling element i with probability p. [9, 23] rewrite the convergence speed: $S(p) = 2\eta(\theta_t - \theta^*)^T \mathbb{E}_{i \sim p}[G_i^{(t)}] - \eta^2 \mathbb{E}_{i \sim p}[{G_i^{(t)}}^T G_i^{(t)}]$. This derivation can be found in Appendix A. Our goal is to find p that maximizes the convergence speed, which is equivalent to minimizing $\mathbb{E}_{i \sim p}[{G_i^{(t)}}^T G_i^{(t)}]$. This minimization is a constrained optimization problem:

$$\min_p \mathbb{E}_{i \sim p}[{G_i^{(t)}}^T G_i^{(t)}] = \min_p \sum_{i=1}^N p_i \|G_i^{(t)}\|_2^2 \quad \text{such that } \sum_{i=1}^N p_i = 1 \text{ and } \forall i \in [1, N],\ p_i > 0.$$

The optimal sampling is $p_i^{GN} = \|\nabla_\theta \ell(\Psi(\theta, x_i), y_i)\|_2 / \sum_{j=1}^N \|\nabla_\theta \ell(\Psi(\theta, x_j), y_j)\|_2$, it is the sampling scheme proportional to the per-sample gradient norms. The proof is in Appendix A. The optimal sampling scheme requires computing $\nabla_\theta \ell(\Psi(\theta, x_i), y_i)$ for all items in the training set, which is too costly to be used in practice. Indeed, computing per-sample gradients requires a

forward and a backward pass on the whole training data before performing each gradient step. For this reason, all the works cited in what follows use approximations of the optimal sampling scheme to keep computations tractable.

2.2. Related work

In this section, we review existing sampling schemes, both in the supervised learning literature and the reinforcement learning one. We start by reviewing the supervised learning literature, where works on importance sampling can be divided into two categories: methods applied to convex problems and methods designed for deep neural networks.

Importance sampling [18] has been a widely studied topic in the context of convex optimization problems in recent years. Bordes et al. [3] introduced LASVM, an online algorithm that leverages importance sampling to train kernelized support vector machines. Richtárik and Takáč [16] subsequently proposed a generalized coordinate descent algorithm that uses importance sampling to optimize the convergence rate of the algorithm. In the context of simple linear classification, the optimal sampling distribution is proportional to the Lipschitz constant of the per-sample loss function [14, 25]. A class of algorithms known as SVRG (Stochastic Variance Reduced Gradient) algorithms [8] has been developed to accelerate the convergence of SGD through variance reduction. While these algorithms offer asymptotic improvements, they have been observed to perform worse than SGD with momentum in the multi-modal setting commonly encountered in Deep Learning.

Importance sampling has previously been utilized in the context of deep learning, often in the form of manually tuned sampling schemes. For example, Bengio et al. [2] manually design a sampling scheme inspired by the way human children learn, while Simo-Serra et al. [21] and Schroff et al. [20] prioritize the sampling of hard examples due to the abundance of easy, non-informative ones. Loshchilov and Hutter [13] use the loss to create the sampling distribution, both keeping a history of losses for previously seen samples and sampling proportionally to a loss ranking. It is worth noting the work of Wu et al. [24], who design a distribution specifically for distance-based losses that maximizes the diversity of losses within a single batch. Fan et al. [5] use reinforcement learning to train a neural network that selects samples for another neural network in order to optimize convergence speed. Importance sampling is also used to accelerate convergence of deep reinforcement learning algorithms [22] using a replay buffer (which acts as a training set). A seminal work introducing a non uniform sampling of the replay buffer is Prioritized Experience Replay [19]; it has been improved in different ways, notably in [7], [11], and [12].

The beginning of this section introducing the convergence speed of SGD, and showing that the optimal sampling scheme is proportional to the per-sample gradient norms, is a result of the works done by Needell et al. [14] and Wang et al. [23]. All cited works of the previous paragraph uses approximations of the optimal sampling scheme, since it can not be used in practice due to a prohibitive computational cost. Instead of studying approximations, Alain et al. [1] use clusters of GPU workers to compute the optimal sampling distribution exactly, rendering the computational time acceptable provided high computing resources.

3. Optimizers and sampling schemes

The derivations of Katharopoulos and Fleuret [9] to obtain the optimal sampling scheme in Subsection 2.1 used the most general form of an SGD step with a mini-batch of one sample, namely $\theta_{t+1} = \theta_t - \eta G_i^{(t)}$, i being the selected index in the training set. This yields the optimal sampling

scheme being proportional to the per-sample gradient norms. This subsection takes into account that the standard SGD is no longer used as optimizer. Improved versions are now used, such as SGD with momentum, RMSProp or ADAM. Our contribution is to highlight the differences between these optimizers in terms of analytical expression of the optimal sampling scheme. p^{GN} remains the optimal sampling scheme for SGD with momentum, but it is not the case for RMSProp and ADAM. The derivations for these three optimizers can be found in Appendices B, C and D. In what follows, we give the general idea of these derivations in the case of RMSProp.

The update equation for RMSProp is: $\theta_{t+1} = \theta_t - \frac{\eta}{\epsilon + \sqrt{v_t}} G_i^{(t)}$ with $v_t = \alpha v_{t-1} + (1-\alpha)\|G_i^{(t)}\|_2^2$,

where ϵ is a small positive constant to avoid the division by 0, η is the learning rate and α is the moving average parameter, generally set to 0.99. For initialization, $v_{-1} = 0$ is classically chosen. The derivations for the convergence speed S given the update equations of RMSProp yield:

$$S(p) = -\mathbb{E}_{i\sim p}\left[\frac{\eta^2 {G_i^{(t)}}^T G_i^{(t)}}{\left(\epsilon + \sqrt{\alpha v_{t-1} + (1-\alpha)\|G_i^{(t)}\|_2^2}\right)^2} - \frac{2\eta {G_i^{(t)}}^T (\theta_t - \theta^*)}{\epsilon + \sqrt{\alpha v_{t-1} + (1-\alpha)\|G_i^{(t)}\|_2^2}}\right].$$

Details of these derivations are given in Appendix C. Contrarily to the derivations done for SGD without momentum, the two terms have to be optimized since they both depend on p (through $G_i^{(t)}$). Given the form of the function to optimize, the solution of the optimization will depend on θ^*. A sampling scheme depending on θ^* is impractical since we (obviously) do not have access to this quantity.

The special case where $\alpha = 1$ deserves our attention, since α is in practice often close to 1. With the initialization $v_{-1} = 0$ and $\alpha = 1$, note that $\forall t,\ v_t = v_{t-1} = 0$. We end up with the equation:

$$S(p) = -\mathbb{E}_{i\sim p}\left[\frac{\eta^2}{\epsilon^2} {G_i^{(t)}}^T G_i^{(t)} - \frac{2\eta}{\epsilon} {G_i^{(t)}}^T (\theta_t - \theta^*)\right],$$

which is the same equation than the one obtained with SGD, the only difference being the learning rate. Once again, the optimization of $\mathbb{E}_{i\sim p}\left[{G_i^{(t)}}^T G_i^{(t)}\right]$ has to be done. In this special case, the optimal sampling scheme is the one proportional to the per-sample gradient norms. However, note that using RMSProp with $\alpha = 1$ is useless since it stripes RMSProp of what does its specificity, namely the moving average of past gradients.

The conclusion of this section is the following. The sampling scheme proportional to the per-sample gradient norms is not the optimal sampling scheme for RMSProp and ADAM. Applying this sampling scheme with RMSProp and ADAM as optimizer is not theoretically grounded. There is no theoretical evidence that it will bring improvements over the uniform sampling scheme in terms of convergence speed.

4. Experimental results

In this section, we verify experimentally that using the sampling scheme proportional to the per-sample gradient norms with RMSProp and ADAM does not necessarily yield improvements over the uniform sampling scheme.

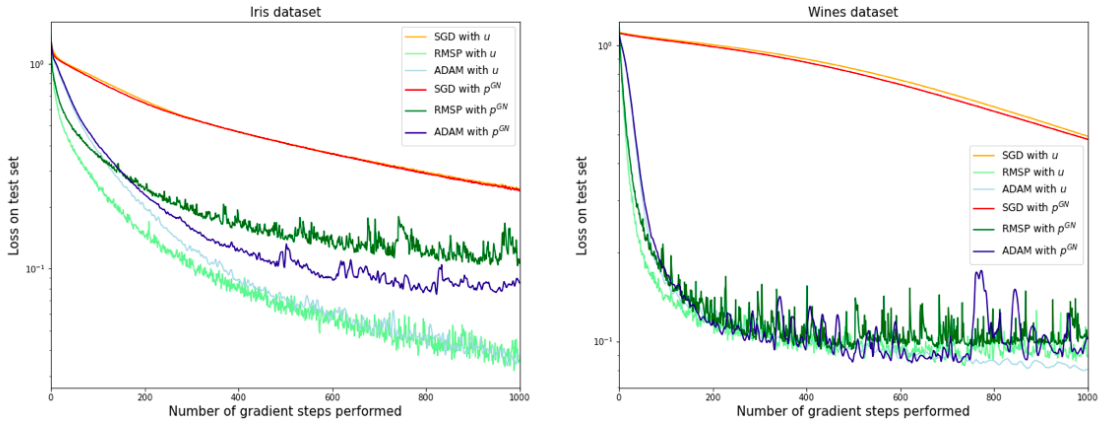

Figure 1: Loss decrease for different optimizers used with different sampling schemes. The x-axis is the number of gradient steps, the y-axis is the value of the loss in log scale.

As explained before, the optimal sampling scheme for SGD, proportional to the per-sample gradient norms, cannot be used in practice since it requires computing the individual gradients of all items in the training set. Since it is not possible to evaluate this sampling scheme on well-known data sets that contain too many items, we restrict ourselves to toy data sets, namely the iris data set [6] and the wines data set [4].

We choose a simple multi-layer perceptron neural network to limit computational burden. Its characteristics are described in Appendix E. The results were obtained thanks to an Apple Macbook Air 2017: 1,8 GHz Intel Core i5 double. With a learning rate of 0.01 and a mini-batch size of 5, we test the optimizers SGD, RMSProp and ADAM of pytorch [15] with default hyper-parameters. The results reported on Fig. 1 are an average on 50 runs. The time taken for each run is reported in Table 1.

Fig. 1 left and right illustrate experimentally that the sampling scheme proportional to the per-sample gradient norms is indeed faster (in terms of iteration) than the uniform sampling scheme for the SGD optimizer. They also show it is not the case for RMSProp and ADAM, which corroborates our conclusion of Section 3: the sampling scheme proportional to the per-sample gradient norms is not optimal for RMSProp and ADAM. On the contrary, it does not bring any improvement compared to the uniform sampling scheme.

Fig. 1 cannot be fully appreciated without Table 1, where the computational time of each sampling scheme used in this work is reported. Whatever the optimizer, the computational time is approximately the same for a given sampling scheme. This table highlights the computational burden of p^{GN} compared to u, knowing that the training sets are made of few items compared to real world datasets. Noting that the cost of p^{GN} grows linearly with the size of the training set, this illustrates the fact that p^{GN} is impractical for common deep learning data sets. If the x-axis of Fig. 1 was the time instead of the number of iterations, the advantage of p^{GN} over u for SGD would disappear.

These preliminary experimental results on toy datasets suggest that the sampling scheme cannot be designed independently from the optimizer it is used with. In particular, it can reasonably be

Table 1: Wall-clock time (in seconds) for each optimizer, for one run (1000 SGD steps)

Dataset	Sampling	SGD	RMSProp	ADAM
Iris	u	0.85 ± 0.14	0.82 ± 0.16	0.83 ± 0.11
Iris	p^{GN}	2.52 ± 0.43	2.58 ± 0.53	2.51 ± 0.49
Wine	u	0.84 ± 0.15	0.82 ± 0.13	0.84 ± 0.11
Wine	p^{GN}	2.49 ± 0.45	2.48 ± 0.51	2.55 ± 0.45

inferred that a sampling scheme that approximates p^{GN} used with an adaptive optimizer is likely to give unsatisfying results. We think this finding is worth sharing, since it has the potential to promote better future research in the field of importance sampling for deep neural network training.

5. Conclusion and future works

This paper is a work in progress, and aims at studying the interplay between the sampling scheme and the optimizer used. To the best of our knowledge, this connection has not already been tackled in the literature.

From a theoretical viewpoint, we highlight that the sampling scheme proportional to the per-sample gradient norms is not optimal for adaptive optimizers (such as RMSProp and ADAM). Preliminary experimental results on toy datasets show that this appears as an issue, since this sampling scheme does not outperform uniform sampling when used with RMSProp and ADAM, besides being more computationally costly. This is all the more important that the cited related works make extensive use of adaptive optimizers, and try to derive new sampling schemes by approximating p^{GN}. The goal of our paper is to highlight this crucial point, so that better research on new sampling schemes will be conducted in the future.

In future works, we will fill in this contribution with more experiments, especially experiments on realistic datasets. We will also try to find computationally efficient sampling schemes specifically designed for adaptive optimizers.

6. Acknowledgments

This work highlights some mathematical issues at stake in deep learning and is to be pursued in the AI illustrations carried out in the EICACS (European Initiative Collaborative Air Combat Standardisation) project.

This publication was co-funded by the European Union. Its contents are the sole responsibility of the author and do not necessarily reflect the views of the European Union or the European Commission. Neither the European Union nor the granting authority can be held responsible for them.

Co-funded by the European Union

References

[1] Guillaume Alain, Alex Lamb, Chinnadhurai Sankar, Aaron Courville, and Yoshua Bengio. Variance reduction in sgd by distributed importance sampling. In *International Conference on Learning Representations*, 2016.

[2] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.

[3] Antoine Bordes, Seyda Ertekin, Jason Weston, Léon Botton, and Nello Cristianini. Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research*, 6(9), 2005.

[4] Dheeru Dua, Casey Graff, et al. Uci machine learning repository, 2017. *URL http://archive. ics. uci. edu/ml*, 7(1), 2019.

[5] Yang Fan, Fei Tian, Tao Qin, Jiang Bian, and Tie-Yan Liu. Learning what data to learn. *arXiv preprint arXiv:1702.08635*, 2017.

[6] Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188, 1936.

[7] Scott Fujimoto, David Meger, and Doina Precup. An equivalence between loss functions and non-uniform sampling in experience replay. *Advances in Neural Information Processing Systems*, 33, 2020.

[8] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. *Advances in neural information processing systems*, 26:315–323, 2013.

[9] Angelos Katharopoulos and François Fleuret. Not all samples are created equal: Deep learning with importance sampling. In *International conference on machine learning*, pages 2525–2534. PMLR, 2018.

[10] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR (Poster)*, 2015.

[11] Aviral Kumar, Abhishek Gupta, and Sergey Levine. Discor: Corrective feedback in reinforcement learning via distribution correction. *Advances in Neural Information Processing Systems*, 33, 2020.

[12] Thibault Lahire, Matthieu Geist, and Emmanuel Rachelson. Large batch experience replay. In *International Conference on Machine Learning*, pages 11790–11813. PMLR, 2022.

[13] Ilya Loshchilov and Frank Hutter. Online batch selection for faster training of neural networks. *arXiv preprint arXiv:1511.06343*, 2015.

[14] Deanna Needell, Rachel Ward, and Nati Srebro. Stochastic gradient descent, weighted sampling, and the randomized kaczmarz algorithm. In *Advances in neural information processing systems*, pages 1017–1025, 2014.

[15] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[16] Peter Richtárik and Martin Takáč. On optimal probabilities in stochastic coordinate descent methods. *arXiv preprint arXiv:1310.3438*, 2013.

[17] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[18] Reuven Y Rubinstein and Dirk P Kroese. *Simulation and the Monte Carlo method*. John Wiley & Sons, 2016.

[19] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In *ICLR (Poster)*, 2016.

[20] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[21] Edgar Simo-Serra, Eduard Trulls, Luis Ferraz, Iasonas Kokkinos, Pascal Fua, and Francesc Moreno-Noguer. Discriminative learning of deep convolutional feature point descriptors. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 118–126. IEEE, 2015.

[22] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[23] Linnan Wang, Yi Yang, Renqiang Min, and Srimat Chakradhar. Accelerating deep neural network training with inconsistent stochastic gradient descent. *Neural Networks*, 93:219–229, 2017.

[24] Chao-Yuan Wu, R Manmatha, Alexander J Smola, and Philipp Krähenbühl. Sampling matters in deep embedding learning. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2859–2867. IEEE, 2017.

[25] Peilin Zhao and Tong Zhang. Stochastic optimization with importance sampling for regularized loss minimization. In *international conference on machine learning*, pages 1–9, 2015.

Appendix A. Derivation of the optimal sampling scheme for SGD

In this appendix, we proove that the sampling scheme proportional to the per-sample gradient norms is the optimal sampling scheme for SGD in its most standard form. Following the notations of Wang et al. [23], let us recall the convergence speed S of SGD under a sampling scheme p as

$$S(p) = -\mathbb{E}_{i\sim p}\left[\|\theta_{t+1} - \theta^*\|_2^2 - \|\theta_t - \theta^*\|_2^2\right].$$

We recall that a stochastic gradient descent update has the form $\theta_{t+1} = \theta_t - \eta G_i^{(t)}$, where $G_i^{(t)}$ is the gradient estimate built from sampling element i with probability p. The following derivations from [23] yield:

$$\begin{aligned}
S(p) &= -\mathbb{E}_{i\sim p}\left[\|\theta_{t+1} - \theta^*\|_2^2 - \|\theta_t - \theta^*\|_2^2\right] \\
&= -\mathbb{E}_{i\sim p}\left[\theta_{t+1}^T\theta_{t+1} - 2\theta_{t+1}^T\theta^* - \theta_t^T\theta_t + 2\theta_t^T\theta^*\right] \\
&= -\mathbb{E}_{i\sim p}\left[(\theta_t - \eta G_i^{(t)})^T(\theta_t - \eta G_i^{(t)}) + 2\eta {G_i^{(t)}}^T\theta^* - \theta_t^T\theta_t\right] \\
&= -\mathbb{E}_{i\sim p}\left[-2\eta(\theta_t - \theta^*)^T G_i^{(t)} + \eta^2 {G_i^{(t)}}^T G_i^{(t)}\right] \\
&= 2\eta(\theta_t - \theta^*)^T\mathbb{E}_{i\sim p}[G_i^{(t)}] - \eta^2\mathbb{E}_{i\sim p}[{G_i^{(t)}}^T G_i^{(t)}].
\end{aligned}$$

Recall also that $\mathbb{E}_{i\sim p}[G_i^{(t)}]$ is a constant with respect to p. Hence, it is possible to gain a speed-up by sampling from the distribution that minimizes $\mathbb{E}_{i\sim p}[{G_i^{(t)}}^T G_i^{(t)}]$.

The minimization of $\mathbb{E}_{i\sim p}[{G_i^{(t)}}^T G_i^{(t)}]$ is a constrained optimization problem:

$$\min_p \mathbb{E}_{i\sim p}[{G_i^{(t)}}^T G_i^{(t)}] = \min_p \sum_{i=1}^N p_i \|G_i^{(t)}\|_2^2 \quad \text{such that } \sum_{i=1}^N p_i = 1 \text{ and } \forall i \in [1, N], p_i > 0.$$

Recall that $G_i^{(t)} = w_i \nabla_\theta \ell(\Psi(\theta_t, x_i), y_i)$ and $w_i = 1/(Np_i)$. Let $g_i = \|\nabla_\theta \ell(\Psi(\theta_t, x_i), y_i)\|_2$. The problem boils down to:

$$\min_p \frac{1}{N^2}\sum_{i=1}^N \frac{1}{p_i} g_i^2, \quad \text{such that } \sum_{i=1}^N p_i = 1 \text{ and } \forall i \in [1, N], p_i > 0.$$

The optimal distribution is $p_i^{GN} = \|\nabla_\theta \ell(\Psi(\theta, x_i), y_i)\|_2 / \sum_{j=1}^N \|\nabla_\theta \ell(\Psi(\theta, x_j), y_j)\|_2$,

it is the sampling scheme proportional to the per-sample gradient norms.

Proof We note $\mu \in \mathbb{R}$ the Lagrange multiplier associated to the equality constraint, $\nu \in \mathbb{R}_+^N$ the Lagrange multipliers associated to the inequality constraints. Hence:

$$\text{Lag}(p, \mu, \nu) = \sum_{i=1}^N \frac{1}{p_i} g_i^2 + \mu\left(\sum_{i=1}^N p_i - 1\right) - \sum_{i=1}^N \nu_i p_i$$

Setting the derivatives of the Lagrangian with respect to the primal variables yields:

$$\forall\, i \in [1, N],\ -\frac{g_i^2}{p_i^2} + \mu - \nu_i = 0$$

Multiplying the above equation by p_i and using $\forall\, i,\ p_i \nu_i = 0$ (complementary slackness), we have: $p_i = g_i / \sqrt{\mu}$, which yields the result. ■

Appendix B. Convergence speed for SGD with momentum

The update equation for SGD with momentum with $B = 1$ is $\theta_{t+1} = \theta_t - \eta v_t$ with $v_t = \mu v_{t-1} + G_i^{(t)}$, where η is the learning rate and μ is the momentum coefficient. For initialization, $v_{-1} = 0$ is classically chosen.

We write the derivations for the convergence speed S given the update equations of SGD with momentum. It yields:

$$\begin{aligned}
S(p) &= -\mathbb{E}_{i\sim p}\left[\|\theta_{t+1} - \theta^*\|_2^2 - \|\theta_t - \theta^*\|_2^2\right] \\
&= -\mathbb{E}_{i\sim p}\left[\theta_{t+1}^T\theta_{t+1} - 2\theta_{t+1}^T\theta^* - \theta_t^T\theta_t + 2\theta_t^T\theta^*\right] \\
&= -\mathbb{E}_{i\sim p}\left[(\theta_t - \eta v_t)^T(\theta_t - \eta v_t) - 2(\theta_t - \eta v_t)^T\theta^* - \theta_t^T\theta_t + 2\theta_t^T\theta^*\right] \\
&= -\mathbb{E}_{i\sim p}\left[\eta^2 v_t^T v_t - 2\eta(\theta_t - \theta^*)^T v_t\right] \\
&= -\mathbb{E}_{i\sim p}\left[\eta^2\left(\mu v_{t-1} + G_i^{(t)}\right)^T\left(\mu v_{t-1} + G_i^{(t)}\right) - 2\eta(\theta_t - \theta^*)^T\left(\mu v_{t-1} + G_i^{(t)}\right)\right] \\
&= -\mathbb{E}_{i\sim p}\left[\eta^2\mu^2 v_{t-1}^T v_{t-1} + 2\eta^2\mu v_{t-1}^T G_i^{(t)} + \eta^2 {G_i^{(t)}}^T G_i^{(t)} - 2\eta(\theta_t - \theta^*)^T\left(\mu v_{t-1} + G_i^{(t)}\right)\right] \\
&= -\eta^2\mu^2 v_{t-1}^T v_{t-1} - 2\eta^2\mu v_{t-1}^T\mathbb{E}_{i\sim p}\left[G_i^{(t)}\right] - \eta^2\mathbb{E}_{i\sim p}\left[{G_i^{(t)}}^T G_i^{(t)}\right] \\
&\quad + 2\eta\mu(\theta_t - \theta^*)^T v_{t-1} + 2\eta(\theta_t - \theta^*)^T\mathbb{E}_{i\sim p}\left[G_i^{(t)}\right].
\end{aligned}$$

Recall that $v_{t-1} = \mu v_{t-2} + G_i^{(t-1)}$ and $G_i^{(t-1)}$ depends on the sampling scheme at time step $t-1$, which is in the past. This means that v_t does not depend on the current sampling scheme p at time step t that has to be optimized. Hence, v_t is a constant with respect to the expectation. Recall also that $\mathbb{E}_{i\sim p}[G_i^{(t)}]$ does not depend on p. The optimization problem boils down to optimizing $\mathbb{E}_{i\sim p}[{G_i^{(t)}}^T G_i^{(t)}]$, which yields the same result as for SGD without momentum. The optimal sampling scheme is the sampling scheme proportional to the per-sample gradient norms when the optimizer is SGD with momentum.

Appendix C. Convergence speed for RMSProp

The update equation for RMSProp is: $\theta_{t+1} = \theta_t - \dfrac{\eta}{\epsilon + \sqrt{v_t}} G_i^{(t)}$ with $v_t = \alpha v_{t-1} + (1-\alpha)\|G_i^{(t)}\|_2^2$

where ϵ is a small positive constant to avoid the division by 0, η is the learning rate and α is the moving average parameter, generally set to 0.99. For initialization, $v_{-1} = 0$ is classically chosen.

We write the derivations for the convergence speed S given the update equations of RMSProp. It yields:

$$
\begin{aligned}
S(p) &= -\mathbb{E}_{i\sim p}\left[\|\theta_{t+1}-\theta^*\|_2^2 - \|\theta_t-\theta^*\|_2^2\right] \\
&= -\mathbb{E}_{i\sim p}\left[\theta_{t+1}^T\theta_{t+1} - 2\theta_{t+1}^T\theta^* - \theta_t^T\theta_t + 2\theta_t^T\theta^*\right] \\
&= -\mathbb{E}_{i\sim p}\left[\left(\theta_t - \frac{\eta}{\epsilon+\sqrt{\alpha v_{t-1}+(1-\alpha)\|G_i^{(t)}\|_2^2}}G_i^{(t)}\right)^T\right. \\
&\qquad\times\left(\theta_t - \frac{\eta}{\epsilon+\sqrt{\alpha v_{t-1}+(1-\alpha)\|G_i^{(t)}\|_2^2}}G_i^{(t)}\right) \\
&\qquad\left.-2\left(\theta_t - \frac{\eta}{\epsilon+\sqrt{\alpha v_{t-1}+(1-\alpha)\|G_i^{(t)}\|_2^2}}G_i^{(t)}\right)^T\theta^* - \theta_t^T\theta_t + 2\theta_t^T\theta^*\right] \\
&= -\mathbb{E}_{i\sim p}\left[\frac{\eta^2}{\left(\epsilon+\sqrt{\alpha v_{t-1}+(1-\alpha)\|G_i^{(t)}\|_2^2}\right)^2}{G_i^{(t)}}^T G_i^{(t)}\right. \\
&\qquad\left.-\frac{2\eta}{\epsilon+\sqrt{\alpha v_{t-1}+(1-\alpha)\|G_i^{(t)}\|_2^2}}{G_i^{(t)}}^T\theta_t + \frac{2\eta}{\epsilon+\sqrt{\alpha v_{t-1}+(1-\alpha)\|G_i^{(t)}\|_2^2}}{G_i^{(t)}}^T\theta^*\right] \\
&= -\mathbb{E}_{i\sim p}\left[\frac{\eta^2{G_i^{(t)}}^T G_i^{(t)}}{\left(\epsilon+\sqrt{\alpha v_{t-1}+(1-\alpha)\|G_i^{(t)}\|_2^2}\right)^2} - \frac{2\eta{G_i^{(t)}}^T(\theta_t-\theta^*)}{\epsilon+\sqrt{\alpha v_{t-1}+(1-\alpha)\|G_i^{(t)}\|_2^2}}\right].
\end{aligned}
$$

Appendix D. Convergence speed for ADAM

The update equation for ADAM is $\theta_{t+1} = \theta_t - \eta\frac{\hat{m_t}}{\epsilon+\sqrt{\hat{v_t}}}$ with $\hat{m}_t = m_t/(1-\beta_1^t)$ and $\hat{v}_t = v_t/(1-\beta_2^t)$ with $m_t = \beta_1 m_{t-1} + (1-\beta_1)G_i^{(t)}$ and $v_t = \beta_2 v_{t-1} + (1-\beta_2)\|G_i^{(t)}\|_2^2$. β_1 is generally set to 0.9 and β_2 to 0.999. For initialization, $v_{-1}=0$ and $m_{-1}=0$ are classically chosen.

We write the derivations for the convergence speed S given the update equations of ADAM. It yields:

$$
\begin{aligned}
S(p) &= -\mathbb{E}_{i\sim p}\left[\|\theta_{t+1}-\theta^*\|_2^2 - \|\theta_t-\theta^*\|_2^2\right] \\
&= -\mathbb{E}_{i\sim p}\left[\theta_{t+1}^T\theta_{t+1} - 2\theta_{t+1}^T\theta^* - \theta_t^T\theta_t + 2\theta_t^T\theta^*\right] \\
&= -\mathbb{E}_{i\sim p}\left[\left(\theta_t - \eta\frac{\hat{m}_t}{\epsilon+\sqrt{\hat{v}_t}}\right)^T\left(\theta_t - \eta\frac{\hat{m}_t}{\epsilon+\sqrt{\hat{v}_t}}\right) - 2\left(\theta_t - \eta\frac{\hat{m}_t}{\epsilon+\sqrt{\hat{v}_t}}\right)^T\theta^*\right. \\
&\qquad\qquad \left. -\theta_t^T\theta_t + 2\theta_t^T\theta^*\right] \\
&= -\mathbb{E}_{i\sim p}\left[\eta^2\frac{\hat{m}_t{}^T\hat{m}_t}{(\epsilon+\sqrt{\hat{v}_t})^2} - 2\eta\frac{\hat{m}_t{}^T\theta_t}{\epsilon+\sqrt{v_t}} + 2\eta\frac{\hat{m}_t{}^T\theta^*}{\epsilon+\sqrt{v_t}}\right] \\
&= -\mathbb{E}_{i\sim p}\left[\eta^2\frac{\hat{m}_t{}^T\hat{m}_t}{(\epsilon+\sqrt{\hat{v}_t})^2} - 2\eta(\theta_t-\theta^*)^T\frac{\hat{m}_t}{\epsilon+\sqrt{v_t}}\right] \\
&= -\mathbb{E}_{i\sim p}\left[\frac{\eta^2}{\left(\epsilon+\sqrt{\frac{v_t}{1-\beta_2^t}}\right)^2}\frac{m_t^Tm_t}{(1-\beta_1^t)^2} - 2\eta\frac{(\theta_t-\theta^*)^T}{\epsilon+\sqrt{\frac{v_t}{1-\beta_2^t}}}\frac{m_t}{1-\beta_1^t}\right] \\
&= -\mathbb{E}_{i\sim p}\left[\frac{\eta^2}{\left(\epsilon+\sqrt{\frac{\beta_2 v_{t-1}+(1-\beta_2)\|G_i^{(t)}\|_2^2}{1-\beta_2^t}}\right)^2}\frac{1}{(1-\beta_1^t)^2}\right. \\
&\qquad \times\left(\beta_1 m_{t-1}+(1-\beta_1)G_i^{(t)}\right)^T\left(\beta_1 m_{t-1}+(1-\beta_1)G_i^{(t)}\right) \\
&\qquad \left. -2\eta\frac{(\theta_t-\theta^*)^T}{\epsilon+\sqrt{\frac{\beta_2 v_{t-1}+(1-\beta_2)\|G_i^{(t)}\|_2^2}{1-\beta_2^t}}}\frac{\left(\beta_1 m_{t-1}+(1-\beta_1)G_i^{(t)}\right)}{(1-\beta_1^t)^2}\right].
\end{aligned}
$$

The derivations lead us to conclusions similar to the ones done for RMSProp. The two terms have to be optimized and the solution of the optimization will depend on θ^*, which makes the optimal sampling scheme impractical.

Appendix E. Neural network architecture

Ours tasks are a classification on simple datasets, which do not require complex architectures to obtain an acceptable classifier. For this reason, and to limit the computational cost, we choose a simple muti-layer perceptron neural network. The first layer is layer with 10 neurons, followed by a ReLU Rectified Linear Unit) activation. The second layer also has 10 neurons. The neural network computes a cross-entropy loss at the output of the last layer.