scientific reports

Check for updates

OPEN

CoolMomentum: a method for stochastic optimization by Langevin dynamics with simulated annealing

Oleksandr Borysenko[1,3]✉ & Maksym Byshkin[2,3]

Deep learning applications require global optimization of non-convex objective functions, which have multiple local minima. The same problem is often found in physical simulations and may be resolved by the methods of Langevin dynamics with Simulated Annealing, which is a well-established approach for minimization of many-particle potentials. This analogy provides useful insights for non-convex stochastic optimization in machine learning. Here we find that integration of the discretized Langevin equation gives a coordinate updating rule equivalent to the famous Momentum optimization algorithm. As a main result, we show that a gradual decrease of the momentum coefficient from the initial value close to unity until zero is equivalent to application of Simulated Annealing or slow cooling, in physical terms. Making use of this novel approach, we propose CoolMomentum—a new stochastic optimization method. Applying Coolmomentum to optimization of Resnet-20 on Cifar-10 dataset and Efficientnet-B0 on Imagenet, we demonstrate that it is able to achieve high accuracies.

A rapid growth of machine learning applications has been observed in recent years. Training of machine learning models is performed by finding such values of their parameters that optimize an objective function. Usually the number of parameters is large and the training dataset is massive. The first order stochastic optimization methods are proved to be most appropriate in this case. To reduce computational costs, the gradient of the objective function with respect to the model parameters is computed on relatively small subsets of the training data, called mini-batches. The resulting value is an unbiased stochastic estimator of the true gradient and it is used with stochastic gradient descent (SGD) methods.

Most theoretical works are focused on convex optimization[1,2], but optimization of nonconvex objective functions is required usually. Empirically it is shown that several optimization algorithms, e.g SGD with momentum[3], Adagrad[4], RMSProp[5], Adadelta[6] and Adam[7] are efficient for training artificial neural networks and optimization of nonconvex objective functions[8,9] In nonconvex setting, the objective function has multiple local minima and the efficient algorithms rely on the "hill climbing" heuristics. Currently, there is a significant gap between mathematical theory and heuristic stochastic optimization methods popular in machine learning.

There is a useful connection between multivariate optimization and molecular simulations. In molecular simulations the hill climbing heuristics is related to passing through the energy barriers. Local energy minima are typical for molecular systems. Based on the detailed analogy between the multivariate optimization and annealing in molecular systems, the Simulated Annealing method was proposed[10]. This nature-inspired optimization method takes name and inspiration coming from annealing (slow cooling) in materials science and computational physics. Simulation of annealing can be used to find an approximation of the global minimum for a function $U(x)$ of many variables. In physics this function is known as a potential energy $U(x)$ of a molecular system. In order to apply Simulated Annealing, one needs a method for sampling from the Gibbs-Boltzmann distribution

$$w_n = \exp(-U_n/T)/Z, \tag{1}$$

where T is a parameter called temperature and Z is a normalizing constant, $Z = \sum_n \exp(-U_n/T)$. The Gibbs distribution w_n gives the probability to find a system x in a state n with energy $U_n = U(x)$. The mean of any quantity $f(x)$ may be calculated utilising the Gibbs distribution, using the formula $\langle f \rangle = \sum_n w_n f$. The Gibbs distribution is one of most important formulas in statistical physics[11].

[1]National Science Center "Kharkiv Institute of Physics and Technology", Kharkiv 61108, Ukraine. [2]Institute of Computational Science, Università della Svizzera italiana, 6900 Lugano, Switzerland. [3]These authors contributed equally: Oleksandr Borysenko and Maksym Byshkin. ✉email: alessandro.borisenko@gmail.com

Classical methods for simulation of molecular systems are Markov chain Monte Carlo (MCMC), molecular dynamics (MD) and Langevin dynamics (LD). Either MD, LD or MCMC lead to equilibrium averaged distributions in the limit of infinite time or number of steps. If simulation is performed at a constant temperature T, these methods may be used to generate samples of Eq. (1). Simulated Annealing can be used with any of these methods, but instead of performing simulation at a constant temperature T, the temperature should be decreased slowly. By performing simulation first at high temperature and then gradually decreasing the temperature value, the states close to the global minimum of $U(x)$ may be found. MCMC, MD and LD have different application areas. MD and LD are based on a numerical integration of the classical equation of motion. They simulate the dynamics of systems, based on the values of the gradient $dU(x)/dx$, that has to be computed on every step. MCMC does not require the gradient information, only $U(x)$ values are required to compute the Metropolis acceptance probability. MCMC methods may overcome energy barriers more efficiently, but they require special MCMC proposals, and there are no equivalently efficient proposals for different systems. If the values of $dU(x)/dx$ are available, then MD and LD are more straightforward methods.

The adaptation of MCMC and LD for optimization is a prospective research direction[12]. MCMC methods are widely used in machine learning, but applications of Langevin dynamics to machine learning only start to appear[13–17]. In this paper, we propose to adapt the methods of Langevin dynamics to the problems of nonconvex optimization, that appear in machine learning. In "Molecular and Langevin dynamics" section we give a brief review of the methods of Molecular and Langevin dynamics and show their relation to the stochastic optimization method. In "Optimization by simulated annealing for machine learning" section we discuss the basics of Simulated Annealing. In "Relation of the Langevin equation with momentum optimizer" section we explore the relation of the discretized Langevin equation with the Momentum optimizer. In "Algorithm" section we present the details of the CoolMomentum algorithm. In "Evaluation" section we evaluate the new algorithm and compare its performance to Adam and Momentum and we leave "Conclusions" section for conclusions.

Molecular and Langevin dynamics

Molecular and Langevin dynamics were proposed for simulation of molecular systems by integration of the classical equation of motion to generate a trajectory of the system of particles. Both methods operate with the classical equation of motion of N particles with coordinates $x = (x_1, x_2, \ldots, x_N)$, velocities $v = dx/dt$ and accelerations $a = d^2x/dt^2$. The Newton's equation of motion for a conservative system is given by

$$m\frac{d^2x}{dt^2} = f(x) \equiv -\frac{dU(x)}{dx}, \tag{2}$$

where m is the mass of particles, $f(x)$ is known as force, and $U(x)$ is the potential energy. The kinetic energy is given by

$$E_k = \sum_{i=1}^{N} \frac{m_i v_i^2}{2}. \tag{3}$$

There are several integration schemes based on discretization of the differential equation (2), the Verlet and Velocity-Verlet algorithms being the most popular among them[18].

In conservative systems, described by Eq. (2), the sum of potential and kinetic energies conserves: $E_k + U = const$. The mean double kinetic energy per dimension per particle

$$T_k = \frac{1}{3 \cdot N}\left\langle \sum_{i=1}^{N} m_i v_i^2 \right\rangle = \frac{2\langle E_k \rangle}{3N} \tag{4}$$

is a parameter called temperature. Here and below $\langle f \rangle = \frac{1}{t}\int_0^t f(t')dt'$ means averaging over time or iterations. Often it is desirable to perform simulations at a given temperature, so that

$$T_k \approx T, \tag{5}$$

where T is the desirable temperature, a parameter of the simulation. In physical simulations, an algorithm or a rule which controls the temperature is conventionally called a thermostat.

If molecules under consideration are allowed to exchange their kinetic energy with a medium (other molecules), then their total energy does not conserve any more. In Langevin Dynamics, two forces are added to the conservative force to account for the energy exchange with the medium - a friction force proportional to the velocity with a friction coefficient $\gamma \geq 0$ and a thermal white noise. These two forces play a role of the thermostat in LD. Explicitly, the Langevin dynamics may be described by the following equation[18–21]:

$$m\frac{d^2x}{dt^2} = f(x) - m\gamma v(t) + R(t), \tag{6}$$

where $R(t)$ is a random uncorrelated force with zero mean and a temperature-dependent magnitude:

$$\begin{aligned} &\langle R(t) \rangle = 0; \\ &\left\langle R(t)R(t') \right\rangle = 2mT\gamma\delta(t - t'), \end{aligned} \tag{7}$$

$\delta(t - t')$ being the Dirac Delta function.

The magnitude of the friction γ determines the relative strength of the dissipation forces with respect to the conservative force $f(x)$. If $\gamma = 0$, one only has conservative forces without energy dissipation and Eq. (6) reduces to Eq. (2).

Several discretization schemes for the Langevin equation were proposed, e.g. a generalization of the Velocity-Verlet integrator to Langevin Dynamics by Vanden-Eijnden and Cicotti[20].

In the high friction limit, the acceleration term in the LHS of Eq. (6) may be neglected and one has

$$m\gamma v(t)dt = f(x(t))dt + R(t)dt. \tag{8}$$

It is known as overdamped Langevin equation. Its first order integrator was proposed by Ermak and McCammon[18]:

$$x(t+\Delta t) = x(t) + \Delta t \frac{1}{m\gamma} f(x(t)) + \sqrt{\Delta t}\sqrt{\frac{2T}{m\gamma}}\xi, \tag{9}$$

where ξ is a random Gaussian noise with zero mean and unit variance. The last term in the RHS of Eq. (9) results from the integral of the random force (7) $\int_0^{\Delta t} R(t')dt'$, known as the Wiener process.

From Eq. (9) one can see that γ enters its denominator, and would result in infinitely large values of updating steps if friction γ is close to zero. Therefore, this integrator is appropriate for essentially high friction values only.

Optimization by simulated annealing for machine learning

Simulated Annealing (SA) is a well established optimization technique to locate the global $U(x)$ minimum without getting trapped into local minima. Though originally SA was proposed as an extension of MCMC[10], SA can be considered as an extension of either MCMC or molecular/Langevin dynamics (see Ch. 12.5 of Schlick[18]). In this paper we propose to adapt these methods to the problem of optimization in machine learning, that require minimization of a function based on the values of its gradients. For instance, this function may be attributed as a loss and the values of the gradient dU/dx may be computed by backpropagation[3].

To get an idea about the basics of Simulated Annealing, one can think as follows. Consider a heavy ball moving in a one-dimensional potential well with multiple minima, separated by barriers. The deepest of the minima is the global one, the others are local. Let the initial mean kinetic energy of the ball be high enough to overcome any energy barrier, therefore the ball passes through all the minima on its quasiperiodic trajectory. According to Eq. (4), high kinetic energy corresponds to high temperature. Suppose now, that the temperature (mean kinetic energy) is gradually decreased. This process has to be slow enough, to ensure that the characteristic cooling time is much longer than the characteristic time of the quasiperiodic motion. In the course of this cooling, another higher-lying local minimum eventually becomes inaccessible as soon as the mean kinetic energy becomes less than the height of its energy barrier. And finally, when the mean kinetic energy becomes less than the barrier between the global and the first local minimum, the ball becomes localized in the global minimum. This consideration may be freely generalized to multiple dimensions.

Therefore, if the values of $dU(x)/dx$ are available, then Simulated Annealing in a combination with molecular dynamics is a well-established method for locating the global minimum of a multivariate function $U(x)$. It is proved to be particularly efficient for nonconvex functions. The value of constant m may be selected arbitrary. For simplicity we can set $m = 1$ throughout. SA may be implemented using e.g. the Velocity-Verlet integrator and one of the thermostats[18]. The beauty of the described above SA is that it has theoretical guarantees to converge to the global minimum of a nonconvex function[22]. However, the convergence is guaranteed in the limit of very slow cooling only. In practice, the efficiency of SA depends on the annealing schedule, that has to be specified by the user.

If the training data is large, then it is computationally expensive to compute the loss and its gradient on the full training set. In this case stochastic optimization is proved to be the only appropriate approach. In stochastic optimization, the values of the loss and its gradient are estimated approximately, on small subsets of training data, called minibatches. If these minibatches are selected randomly from the training data, then the estimated values of the loss $\hat{U}(x)$ and its gradient $d\hat{U}/dx$ are the Monte Carlo approximations of their exact values. Stochastic Gradient Descent is the simplest optimization method and is the method of choice for many applications. Formally it may be written as

$$x_{n+1} = x_n - lr\frac{d\hat{U}}{dx}. \tag{10}$$

In Eq. (10) the constant lr is known as a learning rate, and $d\hat{U}/dx$ is a stochastic gradient. This equation can be compared with Eq. (9). Besides the thermal noise, there are only two differences between these equations: I) $f(x) = -\frac{dU}{dx}$ in (9) is the exact gradient, while $\frac{d\hat{U}}{dx}$ in (10) is the stochastic gradient and II) the discrete time variable t in Eq. (9) is substituted with the iteration number n, so that $lr = \Delta t/(m\gamma)$.

Though the Monte Carlo approximation $d\hat{U}/dx$ is a good unbiased approximation, it is still an approximation and contains noise. One can write[23]

$$\hat{f} = -d\hat{U}/dx = -dU/dx + R, \tag{11}$$

where R is an uncorrelated random noise with zero mean. If the size of the minibatch is large, or the gradient $d\hat{U}/dx$ is computed on the full training data set, then $d\hat{U}/dx = dU/dx$ and $R = 0$. In this case molecular dynamics in a combination with simulated annealing is a well established method for global optimization[18]. On the

other hand, if the batch size is small, then the random noise R may be large. In this case the Langevin dynamics in a combination with simulated annealing may be adapted for global optimization[18].

Relation of the Langevin equation with momentum optimizer

Setting $m = 1$ in the Langevin equation (6) and defining the stochastic force $\hat{f} = f + R$, one obtains

$$\frac{\Delta^2 x}{\Delta t^2} = \hat{f} - \gamma v(t). \tag{12}$$

Expressing the time derivatives in finite differences, one can obtain the next equation:

$$\frac{\Delta^2 x}{\Delta t^2} = \frac{\Delta x_{n+1} - \Delta x_n}{\Delta t^2} = \hat{f}_n - \gamma \frac{\Delta x_{n+1} + \Delta x_n}{2\Delta t}. \tag{13}$$

Now, it is straightforward to obtain the next coordinate updating formula:

$$\Delta x_{n+1} = \rho \Delta x_n + \hat{f}_n \cdot lr \tag{14}$$

with

$$\rho = \frac{1 - \gamma \Delta t/2}{1 + \gamma \Delta t/2} \tag{15}$$

and

$$lr = \frac{\Delta t^2}{1 + \gamma \Delta t/2} = \frac{1 + \rho}{2} \Delta t^2. \tag{16}$$

Equation (14) is nothing else but a famous Momentum optimization algorithm[3] with ρ being a momentum coefficient and lr a learning rate constant.

Due to the change to discrete variables and $m = 1$, Eq. (7) becomes:

$$\begin{aligned} \langle R_n \rangle &= 0; \\ \left\langle R_n^2 \right\rangle \Delta t &= 2\gamma T. \end{aligned} \tag{17}$$

Using Eq. (15) to obtain

$$\gamma = \frac{2}{\Delta t} \cdot \frac{1 - \rho}{1 + \rho}, \tag{18}$$

one can change the last Eq. (17) to:

$$\left\langle R_n^2 \right\rangle \Delta t^2 = 4T \cdot \frac{1 - \rho}{1 + \rho}. \tag{19}$$

For many machine learning applications the optimal ρ value is in the range from 0.5 to 0.99. If $\rho = 0$ then Eq. (14) becomes equivalent to Eq. (10), the Langevin dynamics becomes overdamped, and the Momentum optimizer becomes SGD.

Algorithm

In order to apply Simulated Annealing for optimization, one needs a thermostat to control the temperature. In addition, a temperature schedule (or cooling strategy) has to be specified by the user. The temperature itself does not enter explicitly into our algorithm described by Eqs. (14)–(16) (see also pseudocode in Table 1). From Eq. (19) one can see that, for $\left\langle R_n^2 \right\rangle \Delta t^2 = const$, the product of the temperature and a function of the momentum coefficient stays constant: $4T(1 - \rho)/(1 + \rho) = const$. Therefore, instead of decreasing the temperature directly, one can increase the ratio $(1 - \rho)/(1 + \rho)$ by decreasing the momentum coefficient ρ, which enters our algorithm explicitly.

From Eqs. (15) and (18) one can see that ρ decreases from unity to zero as γ increases from zero to its maximal value $2/\Delta t$, which corresponds to the overdamped regime. The decreasing ρ schedule has to be specified by the user. Different ρ schedules may be used. A possible ρ schedule is given by

$$\rho_n = 1 - (1 - \rho_0)/\alpha^n. \tag{20}$$

If $\alpha = 1$ then $\rho_n = \rho_0$, and if $\alpha < 1$ then ρ_n is a decreasing function of n. In the Momentum optimizer the ρ_n value should be in the range from 0 to 1. Let S be the number of steps (usually S= number of epochs · steps per epoch). Then the algorithm we propose may be presented as a pseudocode given in Table 1.

Comparing with the classical Momentum optimizer, described by Eq. (14), this algorithm requires one additional hyperparameter α, that we call a "cooling rate". Every additional hyperparameter may be painful for machine learning application. However, a good α value may be easily computed. In Simulated Annealing the temperature should be slowly decreased until some minimal value, and therefore the ρ value should be slowly decreased until $\rho = 0$. Given $\rho_S = 0$, from Eq. (20) one can obtain:

Algorithm "CoolMomentum"
Require: $lr = \Delta t^2$ (base learning rate)
Require: ρ_0 (initial momentum coefficient)
Require: S (number of iterations)
Compute: $\alpha = (1-\rho_0)^{1/S}$ (cooling rate)
Initialization: x_0 (Initial parameter vector)
Initialization: $\Delta x_0 = 0$ (Initialize update vector)
for $n = 0..(S-1)$ **do**: (loop over S iterations)
$\hat{f}(x_n) = -d\hat{U}/dx$ (compute stochastic gradient)
$\rho_n = \max(0, 1-(1-\rho_0)/\alpha^n)$ (slowly decrease ρ value until zero)
$lr_n = lr \cdot (1+\rho_n)/2$ (recalculate the learning rate)
$\Delta x_{n+1} = \rho_n \Delta x_n + \hat{f}(x_n) \cdot lr_n$ (update momentum)
$x_{n+1} = x_n + \Delta x_{n+1}$ (update parameters)
end do
return x_S (Resulting parameters)

Table 1. Pseudocode

$\rho_0 \backslash lr$	**0.001**	**0.01**	**0.02**
0.9	0.8697	0.9062	0.9139
0.99	0.8972	**0.9160**	0.9057
0.999	0.9064	div	div

Table 2. Test accuracy of Resnet-20, trained on Cifar-10 for 200 epochs versus Coolmomentum hyperparameters lr and ρ_0. "div" means "divergent". The best value is in bold.

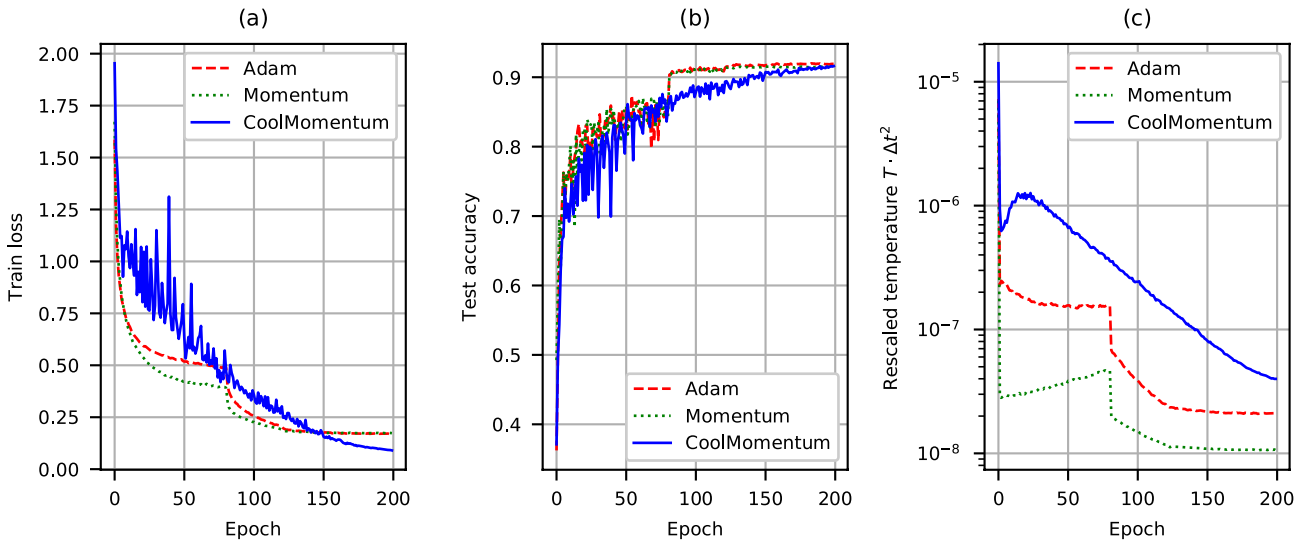

Figure 1. Cifar-10 classification with ResNet-20: training loss (**a**), test accuracy (**b**) and rescaled temperature ($T \cdot \Delta t^2$) (**c**).

$$\alpha = (1-\rho_0)^{1/S}. \tag{21}$$

Evaluation

To evaluate our optimization method, we study the problem of image classification. We trained a deep residual neural network[24] ResNet-20 on the CIFAR-10 dataset with 50000 training images and 10000 testing ones using Adam[7], Momentum[3] and Coolmomentum optimizers. This model has a complicated architecture, more than 270k

of trainable parameters and therefore it is a good model to check the performance of optimization methods. We used the code shared by the Keras team[25]. Training of this model for 200 epochs on gtx1080ti GPU takes about 2 hours. For the Adam optimizer we took the initial value of the learning rate $lr = 0.001$ with an original learning rate decay schedule, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. For the Momentum optimizer we took the initial value of the learning rate $lr = 0.01$ with an original learning rate decay schedule and $\rho = 0.9$ for the momentum coefficient. For Adam and Momentum the learning rate decay factor of 0.1 was applied after the 80th, 120th and 160th epochs and a factor of 0.5 was applied after the 180th epoch. For Coolmomentum we took the base value of the learning rate $lr = 0.01$ and the value of the cooling rate α was taken from Eq. (21) with $\rho_0 = 0.99$. The values of hyperparameters were selected by the trial and error method (see Table 2). For the sake of reproducability, all calculations were performed with the same fixed random generator's seed value.

lr	**0.1**	**0.2**	**0.6**	**0.7**
Top-1, %	76.03	76.55	**76.99**	76.87
Top-5, %	92.63	93.08	93.24	**93.32**

Table 3. Top accuracies for Imagenet classification with Efficientnet-B0 optimized with Coolmomentum at different learning rates. The best values are in bold.

In order to check the performance of the optimization methods on ResNet-20, for each epoch we compute the training loss on the training data set (50000 images) and the testing accuracy on the testing data set (10000 images), and compare the optimization results in Fig. 1a,b, respectively.

To be sure that Simulated Annealing is applied properly, i.e. that the temperature is decreased slowly, one needs a method to calculate the temperature directly during the optimization process. This may be done by using Eq. (4), setting $m = 1$ and changing to discrete variables to obtain:

$$T = \frac{1}{\text{Size}}\left\langle \sum_{i=1}^{\text{Size}} v_i^2 \right\rangle = \frac{1}{\text{Size}\cdot \text{S}} \sum_{i=1}^{\text{Size}} \sum_{n=1}^{S} \left(\frac{\Delta x_{i,n}}{\Delta t} \right)^2, \tag{22}$$

where Size is a number of training parameters of the model and S is a number of time iterations per epoch.

In Fig. 1c we present the values of rescaled temperature $T \cdot \Delta t^2$ calculated with Eq. (22) for all the three optimizers being compared. We choose to calculate rescaled temperature instead of the ordinary one because the actual value of the time step Δt is inavailable for Adam. From Fig. 1c one can see that on the first epoch the temperature significantly drops down for all three optimizers, but only in the case of Coolmomentum it evolves continuously on further epochs, while it changes stepwise according to the prescribed learning rate decay schedule for Adam and Momentum. Therefore, Coolmomentum performs optimization in the Simulated Annealing regime, and by slowly decreasing the temperature it samples the states of the Gibbs distribution (1), which continuously approach the global minimum of the loss function. On the contrary, Adam and Momentum drop the temperature in a stepwise manner. In materials science and physical simulations this cooling regime is called quenching. It produces a variety of non-equilibrium disordered structures, including different glasses. Similarly to physical systems, in this regime the trained model becomes caught in a local minimum of the loss function, and continues to walk there, because the temperature is too low to overcome the local barrier. Indeed, from Fig. 1a one can see that both Adam and Momentum saturate to the constant (and equal) value of the training loss, while Coolmomentum continuously goes below this level.

On the first epochs the training and testing results, produced by CoolMomentum, are worse than those of Momentum and Adam. Indeed, on the first epochs Coolmomentum gives the temperature values significantly higher than Adam and Momentum do (see Fig. 1c). But at high temperatures the Gibbs distribution (1) is less efficient to distinguish between the states with high and low values of the loss function. Nevertheless, as the temperature decreases, Coolmomentum achieves the top values produced by others in terms of the test accuracy (see Fig. 1b) and outperforms them in terms of training loss values (see Fig. 1a), which encourages further studies of different models and datasets.

We also trained Efficientnet[26] B0 on the Imagenet (1000 classes) dataset[27] with 1281167 training images and 50000 testing ones for 218948 steps (about 350 epochs) with batch size 2048 for about 30 hours on v2-8 cloud TPU. At first we ran the publicly available code for training Efficientnet on cloud TPU[28] with default settings: RMSprop with batch-scaled learning rate $0.128 = 0.016 \cdot (2048/256)$, momentum coefficient 0.9, exponential running average decay 0.9, $\epsilon = 0.001$, learning rate decay factor 0.97 for each 2.4 epochs with a linear warm-up for the first 5 epochs. Then we modified it to realize Coolmomentum with base $lr = 0.6$, $\rho_0 = 0.99$ and the cooling rate α calculated from Eq. (21). We set $\rho = 0$ for the first 5 epoch for warm-up. The value of the base learning rate was selected by the trial and error method based on the data of Table 3.

The results are presented in Fig. 2. One can see that in this case Coolmomentum also achieves the top results.

Conclusions

We explore relations between the Langevin dynamics and the stochastic optimization methods, popular in machine learning. The relation of underdamped Langevin dynamics with the Momentum optimizer was studied recently[16]. In this paper we combine Langevin dynamics with Simulated Annealing. To apply Simulated Annealing, the temperature should be decreased slowly until some minimal value. This is usually done by decreasing

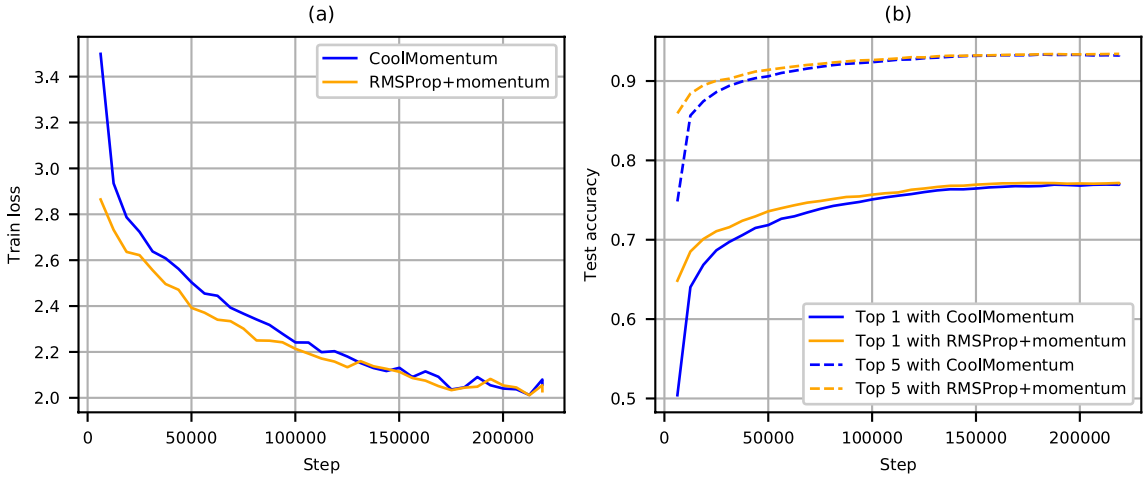

Figure 2. Imagenet classification with Efficientnet-B0: training loss (**a**) and test accuracy (**b**).

the learning rate with a certain schedule. Indeed, from Eq. (19) one can see that, from decreasing the value of $lr \sim \Delta t^2$, the temperature T decreases proportionally. Alternatively, we propose to adapt Simulated Annealing by slowly decreasing the momentum coefficient of the Momentum optimizer, and propose a decreasing schedule for the values of this coefficient. In our case, at the minimal temperature the momentum coefficient becomes zero and the Langevin dynamics becomes overdamped.

The proposed Coolmomentum optimizer requires only 3 tunable hyperparameters (base learning rate, initial momentum coefficient and the total number of optimization steps), while SGD with momentum requires 1 more parameter (learning rate decay factor) and RMSprop and Adam require 2 extra parameters (exponential running average coefficient and a small constant to avoid divergence). In this way, our approach is advantageous, because it reduces the number of tunable hyperparameters and, therefore, demands less computational budget to choose the best values[29]. We demonstrate that training of Resnet-20 on Cifar-10 dataset and Efficientnet-B0 on Imagenet with Coolmomentum optimizer allows to achieve high accuracies. The obtained results indicate that the combination of the Langevin dynamics with Simulated Annealing is an efficient approach for gradient-based optimization of stochastic objective functions.

The convergence analysis of Simulated Annealing was performed by several authors[30–34]. We hope to attract attention of researchers to this optimization method.

Code availability

Our open source code is available at https://github.com/borbysh/coolmomentum.

Received: 31 August 2020; Accepted: 4 May 2021
Published online: 21 May 2021

References

1. Schmidt, M., Le Roux, N. & Bach, F. Minimizing finite sums with the stochastic average gradient. *Math. Program.* **162**, 83–112 (2017).
2. Reddi, S. J., Kale, S. & Kumar, S. On the convergence of adam and beyond. arXiv preprint arXiv:1904.09237 (2019).
3. Rumelhart, D., Hinton, G. & Williams, R. Learning representations by back-propagating errors. *Nature* **323**, 533–536 (1986).
4. Duchi, J., Hazan, E. & Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.* **12**, 2121–2159 (2011).
5. Tieleman, T. & Hinton, G. Lecture 6.5-rmsprop: divide the gradient by a running average of its recent magnitude. *COURSERA Neural Netw. Mach. Learn.* **4**, 26–31 (2012).
6. Zeiler, M. D. Adadelta: an adaptive learning rate method. arXiv preprint arXiv:1212.5701 (2012).
7. Kingma, D. P. & Ba, J. Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014).
8. Goodfellow, I., Bengio, Y. & Courville, A. *Deep Learning* (MIT Press, 2016).
9. Bottou, L., Curtis, F. E. & Nocedal, J. Optimization methods for large-scale machine learning. *Siam Rev.* **60**, 223–311 (2018).
10. Kirkpatrick, S., Gelatt, C. D. & Vecchi, M. P. Optimization by simulated annealing. *Science* **220**, 671–680 (1983).
11. Landau, L. D. & Lifshitz, E. M. *Course of Theoretical Physics, Vol. 5. Statistical Physics* (Pegamon, 1980).
12. Ma, Y.-A., Chen, Y., Jin, C., Flammarion, N. & Jordan, M. I. Sampling can be faster than optimization. *Proc. Natl. Acad. Sci.* **116**, 20881–20885 (2019).
13. Welling, M. & Teh, Y. W. Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 681–688 (2011).
14. Ding, N. *et al.* Bayesian sampling using stochastic gradient thermostats. in *Advances in Neural Information Processing Systems* 27 (2014).
15. Ye, N., Zhu, Z. & Mantiuk, R. Langevin dynamics with continuous tempering for training deep neural networks. in *Advances in Neural Information Processing Systems* 30 (2017).
16. Ma, Y. -A. *et al.* Is there an analog of Nesterov acceleration for MCMC? arXiv preprint arXiv:1902.00996 (2019).
17. Wenzel, F. *et al.* How good is the Bayes posterior in deep neural networks really? arXiv preprint arXiv:2002.02405 (2020).
18. Schlick, T. *Molecular Modeling and Simulation: An Interdisciplinary Guide*, vol. 21 (Springer Science & Business Media, 2010).
19. Bussi, G. & Parrinello, M. Accurate sampling using Langevin dynamics. *Phys. Rev. E* **75**, 056707 (2007).

20. Vanden-Eijnden, E. & Ciccotti, G. Second-order integrators for Langevin equations with holonomic constraints. *Chem. Phys. Lett.* **429**, 310–316 (2006).
21. Van Gunsteren, W. & Berendsen, H. Algorithms for Brownian dynamics. *Mol. Phys.* **45**, 637–647 (1982).
22. Granville, V., Krivánek, M. & Rasson, J.-P. Simulated annealing: a proof of convergence. *IEEE Trans. Pattern Anal. Mach. Intell.* **16**, 652–656 (1994).
23. Friedlander, M. P. & Schmidt, M. Hybrid deterministic-stochastic methods for data fitting. *SIAM J. Sci. Comput.* **34**, A1380–A1405 (2012).
24. He, K., Zhang, X., Ren, S. & Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* 770–778, (2016).
25. https://keras.io/zh/examples/cifar10_resnet/.
26. Tan, M. & Le, Q. V. Efficientnet: rethinking model scaling for convolutional neural networks. arXiv preprint arXiv:1905.11946 (2020).
27. http://www.image-net.org/challenges/LSVRC/2012/downloads.
28. https://cloud.google.com/tpu/docs/tutorials/efficientnet.
29. Sivaprasad, P. T., Mai, F., Vogels, T., Jaggi, M. & Fleuret, F. Optimizer benchmarking needs to account for hyperparameter tuning. arXiv preprint arXiv:1910.11758 (2020).
30. Geman, S. & Hwang, C.-R. Diffusions for global optimization. *SIAM J. Control Optim.* **24**, 1031–1043 (1986).
31. Gidas, B. Global optimization via the Langevin equation. In *1985 24th IEEE Conference on Decision and Control*, 774–778 (IEEE, 1985).
32. Gidas, B. Nonstationary Markov chains and convergence of the annealing algorithm. *J. Stat. Phys.* **39**, 73–131 (1985).
33. Holley, R. & Stroock, D. Simulated annealing via Sobolev inequalities. *Commun. Math. Phys.* **115**, 553–569 (1988).
34. Márquez, D. Convergence rates for annealing diffusion processes. *Ann. Appl. Probab.* **7**, 1118–1139 (1997).

Acknowledgements

MB thanks Swiss National Science Foundation, Grant Number 167326, National Research Program 75 (Big Data) for financial support. OB thanks National Academy of Science of Ukraine, grant number 0121U108687 for financial support. We thank Jeff Dean, TensorFlow Research Cloud and Google Cloud Research for high performance computational resources.

Author contributions

O.B. and M.B. equally developed the theory, conducted the numerical calculations, analysed the results and prepared the manuscript.

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to O.B.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.