Mirror Descent of Hopfield Model

Hyungjoon Soh
hjsoh88@gmail.com
Department of Physics Education, Seoul National University, Seoul 08826, Korea

Dongyeob Kim
ktfa159@snu.ac.kr
Department of Physics and Astronomy, Seoul National University, Seoul 08826, Korea

Juno Hwang
wnsdh10@snu.ac.kr
Department of Physics Education, Seoul National University, Seoul 08826, Korea

Junghyo Jo*
jojunghyo@snu.ac.kr
Department of Physics Education, Department of Physics and Astronomy, and Center for Theoretical Physics and Artificial Intelligence Institute; Seoul National University, Seoul 08826, Korea; and School of Computational Sciences, Korea Institute for Advanced Study, Seoul 02455

Mirror descent is an elegant optimization technique that leverages a dual space of parametric models to perform gradient descent. While originally developed for convex optimization, it has increasingly been applied in the field of machine learning. In this study, we propose a novel approach for using mirror descent to initialize the parameters of neural networks. Specifically, we demonstrate that by using the Hopfield model as a prototype for neural networks, mirror descent can effectively train the model with significantly improved performance compared to traditional gradient descent methods that rely on random parameter initialization. Our findings highlight the potential of mirror descent as a promising initialization technique for enhancing the optimization of machine learning models.

1 Introduction

Recent advancement of machine learning heavily relies on optimization techniques, since machine-learning tasks can be formulated as optimization

*Junghyo Jo is the corresponding author.
The optimization of convex functions is a well-established subject (Nesterov et al., 2018; Nocedal & Wright, 2006). While naive gradient-descent (GD) methods are powerful for finding minima of loss functions, natural gradient-descent (NGD) methods are more advanced as they take into account the curvature of loss functions during the optimization process. Additionally, mirror descent (MD) leverages the dual structure of parametric optimization. With these advanced optimization techniques, the insights from convex optimization are anticipated to significantly contribute to solving various machine-learning problems. Notably, the dual structure of MD has proven to be effective in implementing NGD in machine learning without requiring explicit computation of the curvature of loss functions (Azizan et al., 2021; Zhan et al., 2021; Lan, 2023; Garcia et al., 2023).

In modern machine learning, deep neural networks often involve a massive number of parameters. As such, effective parameter initialization is critical for successfully training neural networks. Techniques such as independent component analysis (Yam et al., 2002), $k$-means clustering (Coates & Ng, 2012), principal component analysis (Chan et al., 2015), and linear discriminant analysis (Masden & Sinha, 2020) are commonly used to extract reference patterns from the data and determine initial weight parameters. While this issue was given serious consideration in the early days of machine learning, it has become less of a priority in light of the substantial computing resources now available. Data-independent initialization schemes, which adjust the scale of randomized parameter values to prevent shrinkage or explosion during forward signal propagation through neural network layers, have become popular. Representative examples include the Xavier and He initializations (Glorot & Bengio, 2010; He et al., 2015). However, given the substantial energy consumption of computation (Strubell et al., 2019), good parameter initialization remains an important issue for reducing computational costs. Therefore, it is still an area of active research to find better techniques for initializing parameters that can reduce the computational burden (Narkhede et al., 2022).

In this study, we find compelling evidence that MD is naturally linked to data-driven parameter initialization in parametric models that belong to the exponential family. Specifically, we observe that the dual variable of MD corresponds to the expectation value of the empirical data distribution, implying that the initial parameter value for the dual space is naturally related to data.

This letter is organized as follows. In section 2, we review the mirror descent to convey its essence for the subsequent discussion. In section 3, we adopt the Hopfield model as an ideal prototype model of neural networks to implement our idea. In section 4, by utilizing MD to optimize the Hopfield model, we demonstrate that MD outperforms usual optimization methods such as GD and NGD with random parameter initialization. We conclude our findings in section 5. To ensure the letter is self-contained, we briefly introduce GD and NGD in the appendix.
2 Mirror Descent

GD can be interpreted as a solution of the following proximal optimization,

$$\theta^{t+1} = \arg\min_{\theta} \left( L(\theta) + \frac{1}{\alpha} D(\theta, \theta^t) \right), \quad (2.1)$$

where $D(\theta, \theta^t) = ||\theta - \theta^t||^2$ is the squared Euclidean distance. GD iteratively updates the parameter $\theta^{t+1}$ by minimizing the augmented loss function that includes a proximal constraint, which ensures that $\theta^{t+1}$ remains “close” to $\theta^t$. Here, the distance between $\theta$ and $\theta^t$ can be generally defined as the Bregman divergence,

$$D(\theta, \theta^t) = F(\theta) - F(\theta^t) - \nabla_\theta F(\theta^t) \cdot (\theta - \theta^t), \quad (2.2)$$

for a convex function of $F(\theta)$. The Bregman divergence is a measure of the difference between the function $F(\theta)$ and its first-order Taylor approximation, $F(\theta^t) + \nabla_\theta F(\theta^t) \cdot (\theta - \theta^t)$, where the divergence vanishes only at $\theta = \theta^t$. Now, the optimal solution of equation 2.1 satisfies

$$\nabla_\theta L(\theta^t) + \frac{1}{\alpha} \left( \nabla_\theta F(\theta^{t+1}) - \nabla_\theta F(\theta^t) \right) = 0, \quad (2.3)$$

which is derived using the first-order Taylor approximation of the loss function $L(\theta)$ around $\theta^t$.

To make this condition more readable, we can introduce a new variable, $\mu \equiv \nabla_\theta F$, which is conjugate to $\theta$. Since the convex function $F(\theta)$ has distinct slopes $\mu$ for every position $\theta$, knowing $\mu$ is equivalent to knowing $\theta$. Then the optimal condition of equation 2.3 can be reformulated as

$$\mu^{t+1} = \mu^t - \alpha \nabla_\theta L(\theta^t). \quad (2.4)$$

At first sight, this equation looks like the GD formula. However, note that the gradient $\nabla_\theta L$ is computed for the change of $\theta$, not for the change of $\mu$. It is remarkable that equation 2.4 corresponds to NGD for the update in $\mu$ space, although it looks like GD (Raskutti & Mukherjee, 2015). To explicitly explain this point, another convex function $G(\mu)$ is introduced as a Legendre transformation of the original convex function $F(\theta)$:

$$F(\theta) + G(\mu) = \theta \cdot \mu. \quad (2.5)$$

The duality between $\theta$ and $\mu$ imposes $\mu = \nabla_\theta F$ and $\theta = \nabla_\mu G$. Then we are ready to interpret $\nabla_\theta L$ for the change of $\mu$ using chain rules,
Figure 1: Mirror descent. The algorithm consists of three steps: (1) transformation from the natural parameter \( \theta^t \) in primal space to \( \mu^t \) in dual space; (2) update of \( \mu^t \to \mu^{t+1} \); and (3) inverse transformation from \( \mu^{t+1} \) in dual space to \( \theta^{t+1} \) in primal space.

\[
\frac{\partial L}{\partial \theta_i} = \sum_j \frac{\partial \mu_j}{\partial \theta_i} \frac{\partial L}{\partial \mu_j} = \sum_j \left[ \frac{\partial^2 G}{\partial \mu_i \partial \mu_j} \right]^{-1} \frac{\partial L}{\partial \mu_j},
\]

(2.6)

where \( \theta_i = \partial G / \partial \mu_i \) is used. Thus, GD-like equation 2.4 implies NGD:

\[
\mu^{t+1}_i = \mu^t_i - \alpha \sum_j \left[ \frac{\partial^2 G}{\partial \mu_i \partial \mu_j} \right]^{-1} \frac{\partial L(\mu^t)}{\partial \mu_j}.
\]

(2.7)

Summarizing the updating procedures of MD as shown in Figure 1, (1) current \( \theta^t \) in primal space is transformed to \( \mu^t \) in dual space. (2) Then it is updated to \( \mu^{t+1} \) through the first-order GD. (3) Finally, the updated \( \mu^{t+1} \) is transformed back to \( \theta^{t+1} \) in primary space.

MD has a great computational merit: the first-order GD in dual space implicitly realizes the second-order NGD. However, one big obstacle for using MD is the inverse transformation to the primal space, which is usually not available. One exceptional case is using the \( p \)-norm as \( F(\theta) = (\sum_i |\theta_i|^p)^{1/p} \). This setup allows easy transformations between primal and dual spaces because the dual function \( G(\mu) = (\sum_i |\mu_i|^q)^{1/q} \) corresponds to the \( q \)-norm given \( 1/p + 1/q = 1 \) (Gentile, 2003). The MD with \( p \)-norm has been nicely applied in deep learning (Azizan et al., 2021).

Another useful application of MD is in policy updates for reinforcement learning (Lan, 2023; Zhan et al., 2021). Specifically, when the parameter vector \( \theta \) has a normalization constraint, such as \( \sum_i \theta_i = 1 \) as a probability distribution, and the Bregman divergence \( D(\theta, \theta') = \sum_i \theta_i \log(\theta_i / \theta'_i) \) is the Kullback-Leibler divergence, the solution to equation 2.1 provides the subgradient algorithm with nonlinear projections, as Beck & Teboulle (2003) showed.
It is important to note that both the $p$-norm and subgradient algorithms rely on a data-independent distance $D(\theta, \theta')$ between models. However, for machine learning problems, it is often desirable to use a data-dependent distance $D(\theta, \theta')$ between models, where the model function $F(\theta)$ is dependent on data. Therefore, our goal is to implement the optimization procedure of MD into a more realistic setting for machine learning problems by incorporating a data-dependent distance.

3 Hopfield Model

The Hopfield model can serve as an excellent prototype for applying MD in a data-dependent manner. It was developed to explain associative memory in neuroscience (Hopfield, 1982), and later it worked as a basic model for generative models of neural networks. Suppose we observe a firing pattern of $n$ neurons, $x = (x_1, x_2, \ldots, x_n)$, and obtain a data of $M$ patterns, $\{x(t)\}_{t=1}^M$, where $t$ is not necessary to be time, but it may be a sample index. Given the $M$ states, one can estimate the relative frequency of them as $\hat{P}(x) = 1/M \sum_{t=1}^M \delta(x - x(t))$. To model the empirical distribution, Hopfield proposed an exponential-family model,

$$ P(x; W, b) = \frac{\exp(b \cdot x + xWx^T)}{Z}, \quad (3.1) $$

where $Z = \sum_x \exp(b \cdot x + xWx^T)$ is a normalization factor, called partition function in statistical mechanics. Note that $x^T$ represents the transpose of $x$. Here the model parameter $b_i$ represents the bias of the $i$th neuron determining its average activity, and $W_{jk}$ represents the coupling strength between the $j$th and the $k$th neurons. Then Hopfield noticed a good choice of parameter values:

$$ b_i = \mathbb{E}_{\hat{P}}[x_i], \quad W_{jk} = \mathbb{E}_{\hat{P}}[x_j x_k]. \quad (3.2) $$

where $\mathbb{E}_{\hat{P}}[O] \equiv \sum_x O(x) \hat{P}(x)$ denotes expectation calculation. We note that the coupling strength $W_{jk}$ depends on the correlation $\mathbb{E}_{\hat{P}}[x_j x_k]$ between two neurons. This is consistent with Hebb’s rule: “Fire together and wire together” (Hebb, 2005). Introducing a parameter vector $\theta \equiv \{b_1, b_2, \ldots, W_{12}, \ldots\}$ and an operator vector $O \equiv \{x_1, x_2, \ldots, x_1 x_2, \ldots\}$, we can simplify the model of equation 3.1 and the Hopfield solution of equation 3.2 as

$$ P(x; \theta) = \frac{\exp(\theta \cdot O(x))}{Z}, \quad \theta_i = \mathbb{E}_{\hat{P}}[O_i]. \quad (3.3) $$
This Hopfield solution makes sense because the probability of a pattern $x$ becomes large once $O(x)$ aligns with the average pattern of $\theta = \mathbb{E}_p[O]$.

Indeed, one can have a better solution that makes the model distribution $P(x; \theta)$ closer to the data distribution $\hat{P}(x)$ (Ackley et al., 1985). Boltzmann machine defines the distance between the two distributions using the Kullback-Leibler divergence:

$$L(\theta) = D_{KL}[\hat{P}(x) || P(x; \theta)] = \sum_x \hat{P}(x) \ln \frac{\hat{P}(x)}{P(x; \theta)}.$$  \hspace{1cm} (3.4)

Given $L(\theta)$, it is straightforward to obtain its gradient:

$$\nabla_\theta L = \mathbb{E}_p[O] - \mathbb{E}_\hat{p}[O].$$ \hspace{1cm} (3.5)

Then, we can minimize the distance by using GD with $\nabla_\theta L$. Once the model and data expectations of $\mathbb{E}_p[O]$ and $\mathbb{E}_\hat{p}[O]$ are consistent, the gradient vanishes. Here we will use MD instead of this GD.

We start from the partition function, $Z = \sum_x \exp(\theta \cdot O(x))$. Its logarithm becomes a cumulant generating function, $F \equiv \ln Z$. One can easily obtain cumulants by differentiating $F$. For example, the first cumulant is obtained by

$$\nabla_\theta F = \sum_x O(x) \frac{\exp(\theta \cdot O)}{Z} = \mathbb{E}_p[O] = \mu.$$ \hspace{1cm} (3.6)

It is the unique feature of the exponential family that the conjugate variable $\mu$ corresponds to the expectation value. Now we apply MD for the Hopfield model in the exponential family:

1. Transformation to dual space by computing expectation, $\mu^t = \mathbb{E}_{p(x; \theta^t)}[O]$
2. Parameter update through GD, $\mu^{t+1} = \mu^t - \alpha \nabla_\theta L(\theta^t)$
3. Reverse transformation to primary space, $\theta^{t+1} = \nabla_\mu G(\mu^{t+1})$

Step 1 is computed by virtue of the exponential-family model. Then, step 2 is also well defined with the gradient of equation 3.5. As always, the obstacle is in the reverse transformation of step 3 because we do not have the explicit function $G(\mu)$, the Legendre transformation of $F(\theta)$.

We solve this problem by using the Taylor expansion of $G(\mu)$ at around $\mu = \mu^t$:

$$G(\mu) \approx G(\mu^t) + \sum_i \frac{\partial G(\mu^t)}{\partial \mu_i} (\mu_i - \mu^t_i) + \frac{1}{2} \sum_{j,k} \frac{\partial^2 G(\mu^t)}{\partial \mu_j \partial \mu_k} (\mu_j - \mu^t_j)(\mu_k - \mu^t_k).$$ \hspace{1cm} (3.7)
For an infinitesimal change of $\mu^t \rightarrow \mu^{t+1}$, it is sufficient to know the local landscape of $G(\mu)$. By differentiating $G(\mu)$ with respect to $\mu_i$ and evaluating at $\mu = \mu^{t+1}$, we have

$$\frac{\partial G(\mu^{t+1})}{\partial \mu_i} = \frac{\partial G(\mu^t)}{\partial \mu_i} + \sum_j \frac{\partial^2 G(\mu^t)}{\partial \mu_i \partial \mu_j} (\mu_j^{t+1} - \mu_j^t), \quad (3.8)$$

where the curvature is

$$\frac{\partial^2 G(\mu^t)}{\partial \mu_i \partial \mu_j} = \frac{\partial^2 \theta_j}{\partial \mu_i} \left[ \frac{\partial \mu_j}{\partial \theta_i} \right]^{-1} = \left[ \frac{\partial^2 F(\theta^t)}{\partial \theta_i \partial \theta_j} \right]^{-1}. \quad (3.9)$$

Here, $\partial \mu_j / \partial \theta_i$ is nothing but the covariance matrix of operators $O$,

$$\frac{\partial \mu_j}{\partial \theta_i} = \frac{\partial}{\partial \theta_i} \sum_x O_j(x) \frac{\exp(\theta^t \cdot O(x))}{Z} = \mathbb{E}_P[O_i O_j] - \mathbb{E}_P[O_i] \mathbb{E}_P[O_j] = C_{ij}. \quad (3.10)$$

Then, equation 3.8 becomes

$$\theta_i^{t+1} = \theta_i^t + \sum_j \left[C^{-1}\right]_{ij} (\mu_j^{t+1} - \mu_j^t). \quad (3.11)$$

This allows us to return to primary space. One caveat of this process is that we need to compute the curvature or the covariance matrix. This process loses the merit of MD that implies the second-order method without explicit calculation of the curvature. When the update step 2 is incorporated into equation 3.11 NGD emerges as

$$\theta_i^{t+1} = \theta_i^t + \alpha \sum_j \left[ \frac{\partial^2 F(\theta^t)}{\partial \theta_i \partial \theta_j} \right]^{-1} \frac{\partial L(\theta^t)}{\partial \theta_j}. \quad (3.12)$$

For the Hopfield model, the covariance matrix is identical to the Hessian matrix:

$$C_{ij} = \frac{\partial^2 F(\theta^t)}{\partial \theta_i \partial \theta_j} = \frac{\partial^2 L(\theta^t)}{\partial \theta_i \partial \theta_j}. \quad (3.13)$$

In summary, by using the Taylor expansion, we can run the MD algorithm for the Hopfield model. Furthermore, we realized that the MD update becomes identical to NGD. Then, one may ask how MD and NGD are different. An important difference exists between the abstract $\theta$ in primary space
and the observable $\mu$ in dual space. For GD or NGD, one should start from a random $\theta^0$ due to no prior knowledge on $\theta$. In contrast, $\mu^0$ in dual space is different. The dual parameter of the exponential family corresponds to an expectation value $\mu = \mathbb{E}_\rho[O]$. Then, we are tempted to adopt the Hopfield solution as an initial value of $\mu^0 = \mathbb{E}_\rho[O]$, since the data distribution is the most confident distribution at the moment. Note that we still do not know $\theta^0$ that is conjugate to $\mu^0 = \mathbb{E}_\rho[p(x; \theta^0)] = \mathbb{E}_\rho[O]$.

Starting from $\mu^0$ in dual space looks great, but it has a fundamental problem. Since $\mu^0$ represents the data distribution, it implies that $L(\theta^0) = D_{KL}[\hat{P}(x)||P(x; \theta^0)] = 0$ and $\nabla_\theta L(\theta^0) = 0$. The vanishing gradient keeps $\mu^1 = \mu^0$ from updating. This result makes perfect sense, since the data distribution is the best available with no need to change. As an alternative choice of $\mu^0 = \mathbb{E}_\rho[O]$, we consider the Hopfield solution as an initial value of $\theta^0$ instead of $\mu^0$. It should be noted that the $\mu^0$, conjugate to this choice of $\theta^0 = \mathbb{E}_\rho[O]$, is no longer $\mu^0 = \mathbb{E}_\rho[O]$, although $\mu^0$ may be close to $\mathbb{E}_\rho[O]$ because the Hopfield solution makes $P(x; \theta^0)$ pretty close to $\hat{P}(x)$. Given the choice of data-driven $\theta^0$, MD can be interpreted as the NGD starting from the Hopfield solution.

4 Results

Now we examine the performance of MD applied in the Hopfield model by comparing the performance of the usual GD algorithm. To conduct this experiment, we first need to synthesize data from the Hopfield model. We obtain a true parameter set of the Hopfield model by sampling from the standard normal distribution as $\theta_i \sim \mathcal{N}(0, 1)$. Given the model parameter $\theta$, we sampled $n$-dimensional binary states $x = (x_1, x_2, \ldots, x_n)$ from the model distribution $P(x; \theta)$ and obtained their data distribution $\hat{P}(x)$. The goal is to infer the true $\theta$ by using just the data distribution. To circumvent the uncertainty that originated from a finite size of samples, we assume that we have infinite samples. Given an infinite number of samples, the estimated distribution $\hat{P}(x)$ should be identical to the ideal distribution $P(x; \theta)$. Therefore, we use $P(x; \theta)$ as a substitute for $\hat{P}(x)$ with infinite samples. Additionally, to make it possible to sum probability distributions over the full configuration space in a tractable way, we limit the state dimension of the binary variable $x$ to a value $n < 20$. For the purpose of the following experiment, we chose $n = 10$, without loss of generality.

We compare the convergence speed and the inference accuracy of different gradient descent methods: GD, GD+Hopfield starting from the Hopfield solution, NGD, and MD. It is expected that the initial loss of MD and GD+Hopfield is lower than the loss of GD and NGD because starting from the Hopfield solution uses prior information of data unlike random initial starting. Furthermore, the second-order methods of MD and NGD lower the loss more dramatically during early iterations (see Figure 2a). Note that
Figure 2: (a) Learning performances of different gradient methods. The Hopfield model $P(x; \theta)$ learns the data distribution $\hat{P}(x)$ of $x = (x_1, x_2, \ldots, x_{10})$. The discrepancy between model and data distributions is defined as a loss $L(\theta)$. Plotted are the loss changes of gradient descent (GD) with different random parameter initializations (blue dotted lines); natural gradient descent (NGD) with different random parameter initializations (green dash-single dotted lines); GD+Hopfield, starting from the Hopfield solution (black solid line); and mirror descent (MD), starting from the Hopfield solution (red dashed line). For the simulation, a learning rate $\alpha = 0.001$ is used for GD, NGD, and MD. (b) Accuracy of inference. The parameter of the Hopfield model is inferred from GD (black), NGD (green), MD (red), and the Hopfield solution (gray). True $\theta_i$ is sampled from the standard normal distribution $\mathcal{N}(0,1)$.

actual computation time is not beneficial because the computation of the curvature and its inversion requires additional computation. After sufficient iterations, we examined the accuracy of inferred $\theta$, compared with true $\theta$, and confirmed that MD provides the most accurate inference (see Figure 2b).

The inversion of covariance matrix $C$ has no problems for a small dimension $n$ and small values of $\theta$. However, when $n$ and/or $\theta$ get larger, the inversion starts to be fragile with poor inference. To make the inversion robust, we modified the covariance matrix as $\tilde{C} = C + \epsilon I$, where $I$ is an identity matrix. Nevertheless, NGD that starts from random initial $\theta^0$ is still very unstable because some eigenvalues $\lambda$ of $\tilde{C}^{-1}$ are very large to make the update of $\theta^{t+1}$ divergent. Then we frequently observed that the loss of NGD easily diverges. The divergence of loss can be suppressed by using a smaller learning rate (i.e., inverse of the largest $\lambda$). The addition of $\epsilon I$ regularizes the largest $\lambda$ to be bounded to an order of $1/\epsilon$. Then the update of $\theta^{t+1}$ stays finite for $\alpha/\epsilon < 1$. Unlike the curvature evaluated at the random initial $\theta^0$, the curvature at the Hopfield solution of $\theta^0 = \mathbb{E}_{P}[O]$ shows quite robust inversion of $\tilde{C}$. This allows the stable update of $\mu^{t+1}$ even with a marginal learning rate (e.g., $\alpha \sim 0.1$).
Figure 3: Mirror descent with a fixed curvature. (a) Loss curves of gradient descent (GD) starting from the Hopfield solution with a learning rate $\alpha = 0.1$ (black solid line); mirror descent (MD) with updating curvatures and $\alpha = 0.001$ and $\epsilon = [10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}]$ (with increasing $\epsilon$ values, the six dashed curves move closer to the black solid line, and their colors change from bright red to dark red); and MD with a fixed curvature and $\alpha = 0.001$ and $\epsilon = [10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}]$ (With increasing $\epsilon$ values, the six dotted curves move closer to the black solid line, and their colors change from bright green to dark green). (b) Learning paths of parameters are projected on the principal component analysis (PCA) plane. Each path departs from the Hopfield solution (top right square) and approaches a global optimum (bottom left triangle). True $\theta_i$ is sampled from $\mathcal{N}(0, 1.5)$.

MD and NGD show more effective parameter updates because they use the curvature information of the loss. The curvature is evaluated at $P(x; \theta)$ with equation 3.10 in MD. This means that the curvature should be obtained from the updated model parameters for every iteration. However, to reduce the computational cost, one may consider using a fixed curvature $C^0$ evaluated at the data distribution $\hat{P}(x)$ as a proxy of updating curvatures at every step. The loss curves for these two schemes are compared in Figure 3a. Overall, MD with updating curvatures (red dashed lines) converges faster than MD with the fixed curvature (green dotted lines). However, both MD
methods outperform GD (black solid line) in spite of the 100 times smaller learning rate compared to the learning rate for GD.

Indeed the fixed curvature leads to different updating directions due to the discrepancy from the actual curvature evaluated at the updating $\theta_t$. This “detouring” takes extra steps toward a global optimum. To visualize the different learning paths on a two-dimensional space, we adopt principal component analysis (PCA). We first make a contour map of $L(\theta)$ for $\theta$ estimated at different (PC1, PC2). Then we put the learning paths of GD, MD with updating curvatures and MD with the fixed curvature (see Figure 3b). Every path starts at the Hopfield solution (top right square) and approaches a single global minimum (bottom left triangle) of the exponential-family model. Due to the information loss under the dimension reduction of PCA, some overshoot is observed in the learning paths on the PCA plane. We demonstrated two important points. First, MD with the fixed curvature shows a large detour to the global minimum. Nevertheless, once it approaches the minimum, the fixed curvature becomes consistent with the curvature evaluated at the global minimum because the model distribution approaches the data distribution. Second, increasing the regularization hyperparameter $\epsilon$ allows approximating the curvature matrix $\tilde{C} = C + \epsilon I$ as an identity matrix. Then, MD approaches GD. Indeed, as we increase $\epsilon$, the learning path approaches the path of GD.

5 Conclusion

Our study reveals an intriguing connection between the mirror descent and the Hopfield solution. The mirror descent offers an elegant formalism for achieving natural gradient descent without explicitly calculating the curvature of loss functions through the use of dual structure in parametric models. Our contribution lies in recognizing the relationship between the conjugate variables of exponential family models in dual space and the expectation values of empirical data distributions. We focused specifically on the Hopfield model, a well-known prototype of neural networks falling under the exponential family models category. The solution to the Hopfield model parameters is proposed as the mean and correlations of empirical data distributions. Through the application of mirror descent, we can explain the Hopfield solution as conjugate variables in dual space, allowing for data-driven parameter initialization.

The data-driven parameter initialization is an important subject in machine learning (Seuret et al., 2017; Das et al., 2021; Chumachenko et al., 2022). We emphasize that our approach is fundamentally different from the previous popular approaches, such as Xavier and He initialization (Glorot & Bengio, 2010; He et al., 2015), which adjust initial scales of parameters depending on network size. Those approaches do not incorporate any information from data. Furthermore, the mirror descent enjoys the second-order method that considers the loss curvature of updating model parameters. In
summary, the mirror descent can provide a nice optimization scheme with data-driven parameter initialization and updated training neural networks. Cachi et al. (2020) recently demonstrated that using input data as initial weights in spiking neural networks results in a tenfold reduction in training time without sacrificing accuracy. This finding suggests that our idea has potential beyond proof of concept. However, for practical machine learning models, several challenges must be addressed before the general application of our approach. These include the incorporation of hidden variables that do not appear in data and the generalization of the idea of data-driven parameter initialization beyond exponential families. Although our approach shows promise, further work is required to overcome these challenges and fully realize its potential in practical machine learning applications.

Appendix: Natural Gradient-Descent Method

Optimization problems can be formulated to find a minimum or maximum point $\theta^*$ of a designed objective function $L(\theta)$. Note that the model parameter $\theta = (\theta_1, \theta_2, \ldots)$ is a vector in general. Here, if $L(\theta)$ is a convex function, iterative updates of $\theta$, 

$$
\theta_i^{t+1} = \theta_i^t - \alpha \frac{\partial L(\theta^t)}{\partial \theta_i^t},
$$

(A.1)

which follow the steepest direction of $L(\theta)$, guarantee reaching $\theta^*$. The learning rate $\alpha$ controls the degree of updates. This is the gradient-descent (GD) method. This method has some unsatisfactory points. First, it does not say anything about a starting position of $\theta^0$. Second, the learning rate is homogeneous ($\alpha_i = \alpha$) for every direction, which is not necessary. Third, the intuitive formula of equation A.1 is not covariant under a transformation of $\theta$. Suppose that we double $\theta_i \rightarrow 2\theta_i$. Then, $\theta_i^{t+1}$ and $\theta_i^t$ doubles, but $\partial L/\partial \theta_i$ halves.

To make the formula covariant, the natural gradient-descent (NGD) method is developed:

$$
\theta_i^{t+1} = \theta_i^t - \alpha \sum_j \left[ \frac{\partial^2 L(\theta^t)}{\partial \theta_i \partial \theta_j} \right]^{-1} \frac{\partial L(\theta^t)}{\partial \theta_j},
$$

(A.2)

where the Hessian matrix, or the curvature, of $L(\theta)$ is considered for more effective updates. It is important to note that in order to make NGD covariant, it is not necessary for the curvature matrix to be the Hessian matrix. For instance, the Fisher information matrix is another option that can be used. NGD is a second-order method that requires computing the Hessian matrix and its inverse. This additional computation is costly for updating
high-dimensional $\theta$. For practical implementation, many smart numerical techniques, such as Broyden-Fletcher-Goldfarb-Shanno (BFGS) (Davidon, 1991), have been developed to approximate the inverse of the Hessian matrix. NGD has also been implemented for deep learning with Kronecker-factored approximate curvature (K-FAC) (Martens & Grosse, 2015). Recently, a method called FishLeg (Garcia et al., 2023) has been developed to address the inversion of the Fisher information matrix using Legendre-Fenchel duality.

Acknowledgments

This work was supported in part by the Creative-Pioneering Researchers Program through Seoul National University, a National Research Foundation of Korea grant (2022R1A2C1006871) (to J. J.), and the Summer Internship Program of Artificial Intelligence Institute of Seoul National University (to D. K.).

References

Ackley, D. H., Hinton, G. E., & Sejnowski, T. J. (1985). A learning algorithm for Boltzmann machines. *Cognitive Science, 9*(1), 147–169. 10.1207/s15516709cog0901_7
Azizan, N., Lale, S., & Hassibi, B. (2021). Stochastic mirror descent on overparameterized nonlinear models. *IEEE Transactions on Neural Networks and Learning Systems, 33*(12).
Beck, A., & Teboulle, M. (2003). Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters, 31*(3), 167–175. 10.1016/S0167-6377(02)00231-6
Cachi, P. G., Ventura, S., & Cios, K. J. (2020). Fast convergence of competitive spiking neural networks with sample-based weight initialization. In *Proceedings of Information Processing and Management of Uncertainty in Knowledge-Based Systems, 18th International Conference* (pp. 773–786).
Chan, T.-H., Jia, K., Gao, S., Lu, J., Zeng, Z., & Ma, Y. (2015). PCANet: A simple deep learning baseline for image classification? *IEEE Transactions on Image Processing, 24*(12), 5017–5032. 10.1109/TIP.2015.2475625, PubMed: 26340772
Chumachenko, K., Iosifidis, A., & Gabbouj, M. (2022). Feedforward neural networks initialization based on discriminant learning. *Neural Networks, 146*, 220–229. 10.1016/j.neunet.2021.11.020, PubMed: 34902796
Coates, A., & Ng, A. Y. (2012). Learning feature representations with K-means.
Das, D., Bhalgat, Y., & Porikli, F. (2021). *Data-driven weight initialization with Sylvester solvers*. arXiv:2105.10335.
Davidon, W. C. (1991). Variable metric method for minimization. *SIAM Journal on Optimization, 1*(1), 1–17. 10.1137/0801001
Garcia, J. R., Freddi, F., Fotiadis, S., Li, M., Vakili, S., Bernachia, A., & Hennequin, G. (2023). Fisher-Legendre (FishLeg) optimization of deep neural networks. In *Proceedings of the International Conference on Learning Representations*. 
Gentile, C. (2003). The robustness of the p-norm algorithms. *Machine Learning*, 53, 265–299 (pp. 1–11).

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feed-forward neural networks. *Journal of Machine Learning Research—Proceedings Track*, 9, 249–256.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 1026–1034).

Hebb, D. O. (2005). *The organization of behavior: A neuropsychological theory*. Psychology Press.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. In *Proceedings of the National Academy of Sciences*, 79(8), 2554–2558. 10.1073/pnas.79.8.2554

Lan, G. (2023). Policy mirror descent for reinforcement learning: Linear convergence, new sampling complexity, and generalized problem classes. *Mathematical Programming*, 198(1), 1–48. 10.1007/s10107-022-01816-5

Martens, J., & Grosse, R. (2015). Optimizing neural networks with Kronecker-factored approximate curvature. In *Proceedings of the International Conference on Machine Learning* (pp. 2408–2417).

Masden, M., & Sinha, D. (2020). *Linear discriminant initialization for feed-forward neural networks*. arXiv:2007.12782.

Narkhede, M. V., Bartakke, P. P., & Sutaone, M. S. (2022). A review on weight initialization strategies for neural networks. *Artificial Intelligence Review*, 55(1), 291–322. 10.1007/s10462-021-10033-z

Nesterov, Y. (2018). *Lectures on convex optimization*. Springer.

Nocedal, J., & Wright, S. (2006). *Numerical optimization* (2nd ed.). Springer.

Raskutti, G., & Mukherjee, S. (2015). The information geometry of mirror descent. *IEEE Transactions on Information Theory*, 61(3), 1451–1457. 10.1109/TIT.2015 .2388583

Seuret, M., Alberti, M., Liwicki, M., & Ingold, R. (2017). PCA-initialized deep neural networks applied to document image analysis. In *Proceedings of the 14th IAPR International Conference on Document Analysis and Recognition* (pp. 877–882).

Strubell, E., Ganesh, A., & McCallum, A. (2019). *Energy and policy considerations for deep learning in NLP*. arXiv:1906.02243.

Yam, Y.-F., Leung, C.-T., Tam, P. K., & Siu, W.-C. (2002). An independent component analysis based weight initialization method for multilayer perceptrons. *Neurocomputing*, 48(1-4), 807–818. 10.1016/S0925-2312(01)00674-9

Zhan, W., Cen, S., Huang, B., Chen, Y., Lee, J. D., & Chi, Y. (2021). *Policy mirror descent for regularized reinforcement learning: A generalized framework with linear convergence*. arXiv:2105.11066.

Received December 14, 2022; accepted May 7, 2023.