On the Convergence of Online Mirror Descent(OMD) and Stochastic Mirror Descent(SMD)

Nahom Seyoum NAHOM.SEYOUM@YALE.EDU
Department of Statistics
Yale University

Haoxiang You HAOXIANG.YOU@YALE.EDU
Department of Mechanical Engineering and Material Science
Yale University

Editor: Seyoum, You

Keywords: Online Mirror Descent, Variational Coherence, Quasi Convex Optimization

1 Introduction

Since its genesis in 1983, mirror descent has been particularly popular as it was able to weather some of the pitfalls associated with the traditional gradient descent that preceded it. It did this by adapting to the "geometry of the problem" through introducing a new measure of distance known as the Bregman divergence. Recent scholarship has started to concern itself with specific classes of problems associated with mirror descent as the applications have been very plenty. In this paper, we will be looking at two different types of Mirror Descent algorithms i.e. the Online Mirror Descent and the stochastic mirror descent to establish convergence rates for assessments of effectiveness and robustness.

To do so, we will be surveying the following two papers:

1. Convergence of online mirror descent by Lei, Yunwen and Ding-Xuan Zhou (Lei and Zhou, 2020)
2. On the convergence of mirror descent beyond stochastic convex programming by Zhengyuan Zhou, Panayotis Mertikopoulos, Nicholas Bambos, Stephen Boyd, Peter Glynn (Zhou et al., 2018)

1.1 Motivation for Online Learning

In machine learning problems, our goal is to find an optimal set of parameters represented by the variable (usually referred as weights) $w \in \mathbb{R}^d$. This optimization aims to minimize the empirical loss, which quantifies the discrepancy between the model's predictions and the actual outcomes observed in the data. Mathematically, we express this objective as:

$$\underset{w}{\text{minimize}} \quad \mathbb{E}_z[f(w, z)], \tag{1}$$

where z represents data drawn from some underlying distribution, for instance, in the context of image classification tasks, z consists of pairs (x, y), where x denotes an image and y denotes the corresponding label.

A common way to solve the problem (1) is by using *stochastic gradient descent* (SGD) in the form of

$$w_{t+1} = w_t - \alpha_t g_t. \tag{2}$$

Here $g_t \in \partial_w f(w_t, z_t)$ denotes the sub-differential of f with respect to w. Because f is dependent on the data z_t, which is available after a particular sample is drawn from the distribution. The SGD algorithm can be viewed as an online learning problem to minimize *regret* define following

$$R_\tau(w) = [\sum_{t=1}^{\tau} f_t(w_t)] - \min_w [\sum_{t=1}^{\tau} f_t(w)], \tag{3}$$

where $f_t(w_t) = f(w_t, z_t)$.

The interplay between online learning and optimization algorithms like SGD serves as a driving force for the analytical study conducted in this project.

1.2 Motivation for Online Mirror Descent

Online Mirror Descent (OMD) algorithm is ultimately an extension of the classical online gradient descent. The OMD algorithm uses a mirror map Ψ, instead of the simple quadratic map Ψ^2, to capture more complex data geometric structures beyond those assumed in Hilbert spaces. The OMD operates by updating the sequence of parameters $\{w_t\}_t$ starting from an initial vector w_1 using the update rule:

$$\nabla\Psi(w_{t+1}) = \nabla\Psi(w_t) - \eta_t \nabla_w [f(w_t, z_t)], \tag{4}$$

where η_t is the step size at iteration t. This formulation allows the algorithm to adaptively adjust the parameters based on the structure of the data and the form of the loss function, effectively navigating the parameter space to minimize the loss.

Remark 1 *The above motivation ultimately drives the formulations used in the paper we are evaluating but OMD also comes with a suite of benefits. For example, OMD is particularly useful in high-dimensional settings where data may be sparse. Traditional methods might struggle with the curse of dimensionality, where distances between points become misleadingly uniform. The mirror map can be chosen to emphasize important dimensions and ignore irrelevant ones, thus preserving computational resources and enhancing the focus on significant attributes.*

2 First Paper

The first paper titled " investigates OMD under strongly convex, differentiable, and smooth settings. Here necessary and sufficient conditions are presented in terms of the step size sequence $\{\eta_t\}_t$ for the convergence of an OMD algorithm with respect to the expected Bregman distance induced by the mirror map. Although the original paper relies on a linear rate step size decay ($\eta_t = \frac{\eta_0}{t+1}$) to establish its theoretical proofs, our numerical experiments reveal a more favorable convergence rate when employing a step size scaling

inversely proportional to the square root of the number of iterations ($\eta_t = \frac{\eta_0}{\sqrt{t+1}}$). This step-size scheduling result is consistent with the choices we've encountered for subgradient descent and mirror descent in our classes.

For the remainder of this section, we will introduce the setup and primary theoretical findings of the first paper. Subsequently, we will perform a numerical experiment based on these theoretical results. Finally, we will conclude our examination of the first paper with a discussion. Detailed proofs and code can be found in Appendix A. The extension of these results to more general settings, such as non-convex online mirror descent, will be explored in the study of the second paper.

2.1 Paper Setup and Assumptions

Here, we will outline the problem setup and main assumptions made in the first paper. Within the scope of our study, we assume that the mirror map Ψ is differentiable, σ_Ψ-strongly convex with respect to some norm $|\cdot|$, and L_Ψ-strongly smooth. Additionally, we assume that the objective function $f(w, z)$ is a convex function and L-strongly smooth function with respect to w. Moreover, we assume there exists an $\sigma_F > 0$ such that

$$\langle w^* - w, \nabla F(w^*) - \nabla F(w)\rangle \geq \sigma_F D_\Psi(w^*, w), \forall w \in \mathcal{W}. \tag{5}$$

This is hold when the target $f(w, z)$ is $\frac{L_\Psi \sigma_F}{2}$-strongly convex respect to w for any z. To see this, we have

$$\begin{aligned}\langle w^* - w, \nabla F(w^*) - \nabla F(w)\rangle &\geq (\text{by convexity})\frac{L_\Psi \sigma_F}{2}\|w^* - w\|^2 &(6)\\ &\geq (\text{by smoothness})\frac{L_\Psi \sigma_F}{2}\frac{2}{L_\Psi}D_\Psi(w^*, w) &(7)\\ &= \sigma_F D_\Psi(w^*, w) &(8)\end{aligned}$$

Next, we introduce assumptions regarding the data distribution. In many machine learning applications, the data $z = (x, y)$ comprises pairs of inputs and corresponding labels. In our study, we assume that the input x has a bounded dual norm, denoted as $\sup_{x\in\mathcal{X}} |x|_* < \infty$. Moreover, we assume that the label y has a bounded second-order moment, represented as $\mathbb{E}_Z[Y^2] < \infty$, and that the covariance matrix $C_X = \mathbb{E}_Z[XX^\top]$ is positive definite.

The paper then separates the discussion based on the variance defined by

$$\inf_{w\in\mathcal{W}} \mathbb{E}_Z[|Y - \langle w, X\rangle|\|X\|_*]. \tag{9}$$

The problem is classified as having positive variance if the quantity mentioned above is positive, and as having zero variance when this quantity equals zero. In the scenario of zero variance, employing a constant step size is adequate to ensure almost sure convergence and a linear rate of convergence can be achieved. Conversely, in the case of positive variance, it becomes necessary to decay the step size to achieve convergence, albeit at a slower rate of $O(\frac{1}{T})$. Intuitively, zero variance indicates that the data can be accurately represented by the linear mapping $y = \langle w^*, x\rangle$ for some weight vector w^*. Conversely, the problem exhibits positive variance when either the mapping cannot be accurately described solely by a linear function, or there is some degree of noise present in the data distribution, i.e. $y = \langle w^*, x\rangle + \sigma$, where σ is some random noisy.

2.2 Main Theoretical Results

Here we will present the main theoretical results separate for the positive variance case and zero variance case.

Theorem 2 *Assume* $\inf_{w\in\mathcal{W}} \mathbb{E}_Z[|Y - \langle w, X\rangle| \|X\|_*] > 0$, *then for the OMD algorithm* $\lim_{t\to\infty} \mathbb{E}_{z_1,\ldots,z_{t-1}}[D_\Psi(w^*, w_t)] = 0$ *if and only if.*

$$\lim_{t\to\infty} \eta_t = 0 \quad \textit{and} \quad \sum_{t=1}^{\infty} \eta_t = \infty \tag{10}$$

Moreover: (a) If $\lim_{t\to\infty} \eta_t = 0$, *then there exist some constants* $t_0 \in \mathbb{N}$ *and* $\tilde{C} > 0$ *such that*

$$\mathbb{E}_{z_1,\ldots,z_{T-1}}[D_\Psi(w^*, w_T)] \geq \frac{\tilde{C}}{T - t_0 + 1}, \quad \forall T \geq t_0. \tag{11}$$

(b) If the step size sequence takes the form $\eta_t = \frac{4}{(t+1)\sigma_F}$, *then*

$$\mathbb{E}_{z_1,\ldots,z_{t-T}}[D_\Psi(w^*, w_T)] = O(\frac{1}{T}). \tag{12}$$

Notice, in the above theorem, the step size sequence decays linearly with respect to the iteration number. However, such a decay schedule to choose for the theorem to work, in our later numerical experiment, we show a faster converge rate can be achieved if the decay of the step size is proportional to the square root of the iteration number, i.e. $\eta_t = \frac{\eta_0}{\sqrt{t+1}}$.

Theorem 3 *Assume* $\inf_{w\in\mathcal{W}} \mathbb{E}_Z[|Y - \langle w, X\rangle| \|X\|_*] = 0$, *then for the OMD algorithm* $\lim_{t\to\infty} \mathbb{E}_{z_1,\ldots,z_{t-1}}[D_\Psi(w^*, w_t)] = 0$ *if and only if.*

$$\sum_{t=1}^{\infty} \eta_t = 0 \tag{13}$$

Moreover, if we set step size $\eta < \frac{\sigma_\Psi}{2L}$, *then we have*

$$(1 - 2\sigma_\Psi^{-1} L\eta)^T D_\Psi(w^*, w_1) \leq \mathbb{E}_{z_1,\ldots,z_{t-1}}[D_\Psi(w^*, w_t)] \leq (1 - 2^{-1}\sigma_F\eta)^T D_\Psi(w^*, w_1) \tag{14}$$

2.3 Numerical Experiment

In this subsection, we undertake a numerical experiment to validate the theoretical results outlined in the first paper. Our focus is on minimizing an empirical loss within the probability simplex, which is defined as follows:

$$\underset{w\in\Delta^d}{\text{minimize}} \quad F(w) = \mathbb{E}_{X,Y}[f(w, X, Y)] = \mathbb{E}_{X,Y}[\|\langle w, X\rangle - Y\|_2^2], \tag{15}$$

where $X \sim \mathcal{N}(0, I)$, and $Y = \langle w^*, X \rangle + \sigma Z$, with $Z \sim \mathcal{N}(0, 1)$. Here, σ is a constant controlling the level of noise in the system. We encounter a zero-variance problem when $\sigma = 0$ and we use $\sigma = 0.2$ to construct a positive-variance example. We use KL divergence as Bregman distance and the Ψ is the negative entropy.

In our evaluation under the zero-variance scenario, we compare the performance of online mirror descent with that of the projected gradient descent method, both utilizing a constant step size as recommended in the paper. We observe that the KL divergence between the current iteration and the minimizer decreases roughly linearly in log space. Interestingly, in this setting, online mirror descent converges more quickly than projected gradient descent in terms of the KL distance, especially when adapting a larger step size.

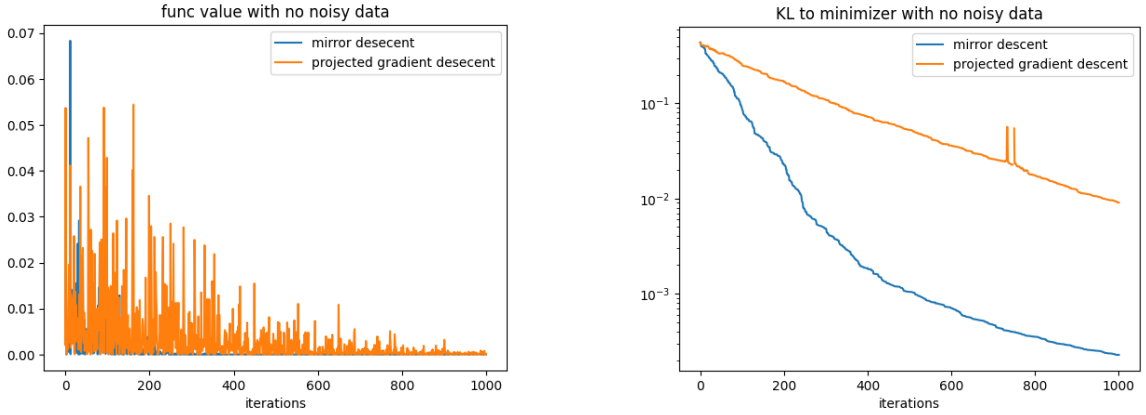

Figure 1: Convergence of online mirror descent under zero-variance case

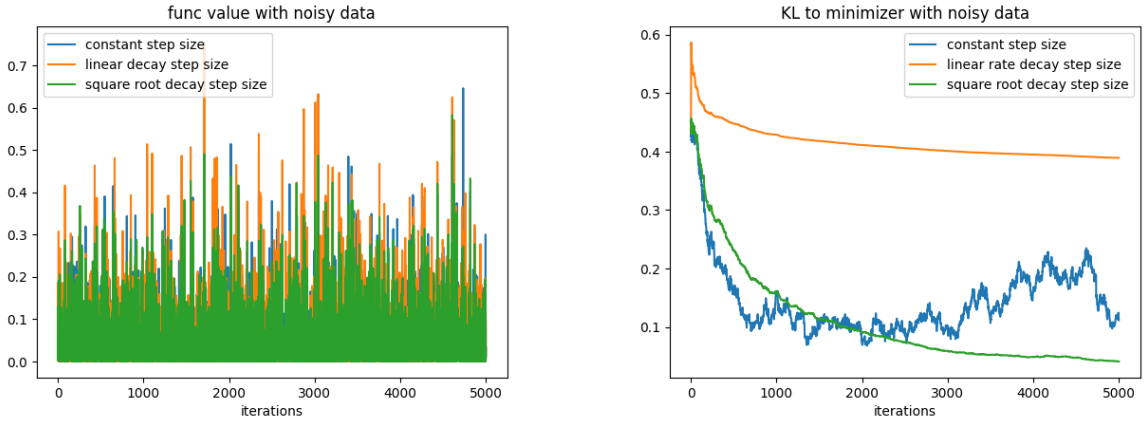

Figure 2: Convergence of online mirror descent under positive-variance case

Next, we assess the convergence of online mirror descent under a positive variance setting. We investigate the convergence behavior of online mirror descent with various step size decay schedules, including:

1. A constant step size.

2. Step size decays linearly with the iteration, i.e., $\eta_t = \frac{\eta_0}{t+1}$, as used in the paper to achieve $O(\frac{1}{T})$ convergence.

3. Step size decay with the square root of the iteration, i.e., $\eta_t = \frac{\eta_0}{\sqrt{t+1}}$, a decay schedule commonly employed in subgradient descent and mirror gradient descent in our classes.

We observed that the KL distance between the current iteration and the global minimizer fluctuated after a certain number of iterations. However, we noticed that w converged to w^* in terms of the KL distance when employing a step size decay proportional to the square root of the iteration. Conversely, when applying the step size decay linearly with iteration as described in the paper, the KL distance increased initially and then converged very slowly. This is not contradict to the paper's theoretical results, however, as the KL distance is not a strongly-smooth distance required by the theoretical proof.

2.4 Discussion

The first paper studies the convergence of OMD under strongly convex and smooth settings. The achieved analysis on the one-step progress of the OMD algorithm and established a lower and upper bound. A necessary and sufficient condition for the convergence is given.

However, the paper has several strict assumptions, for example, they assume the Ψ is strongly smooth. For many applications such as when we do optimization under probability simplex and use KL as Bregman distance, the smooth assumption does not hold anymore. Indeed, the paper mention examples of Ψ as some p-norm, where $1 < p \leq 2$.

Indeed, in the numerical test, we do observe convergence in terms of the KL distance when we apply a decay step size in online mirror descent. This suggests that we might be able to eliminate the smoothness assumptions in certain cases. Additionally, we notice that a better convergence rate can be achieved when employing a step size decay proportional to the square root of the number of iterations, as opposed to decaying linearly as mentioned in the paper. We will explore more general cases when some smooth and convexity conditions do not in our study of the second paper.

3 Second Paper

3.1 Motivation and Previous Results

Historically, the convergence properties of SMD have been well-established in convex settings, with seminal contributions by Nemirovski and Yudin laying the groundwork. In non-convex scenarios, however, the convergence of SMD's last iterate has only recently been explored. Studies by Shamir and Zhang, and Nedic and Lee, have shown that in strongly convex environments, the last iterate of SMD can achieve convergence rates comparable to those of its ergodic average.

Further investigations into non-convex problems, such as those by Jiang and Xu on variational inequalities and phase retrieval, demonstrate SMD's ability to converge to global optima under certain conditions. Moreover, research by Ghadimi and Lan indicates that employing randomized stopping times with SGD can lead to convergence at critical points on average in non-convex settings. These findings suggest that while the convergence behavior

of SMD in non-convex programs is less understood, recent results are promising, pointing towards effective strategies for approaching these complex optimization landscapes.

3.2 Problem Setup

In many practical situations, optimization problems involve decision-making under uncertainty. One common approach to addressing such problems is to consider the expected performance over some uncertainty characterized by a probability distribution. The following outlines a stochastic optimization framework where the goal is to minimize an expected objective function subject to certain constraints:

$$\begin{aligned} \text{minimize} \quad & f(x) \\ \text{subject to} \quad & x \in X, \end{aligned}$$

where the objective function $f(x)$ is defined by the expectation:

$$f(x) = \mathbb{E}[F(x;\omega)],$$

Here, F represents a stochastic objective function mapping from $X \times \Omega \to \mathbb{R}$, with Ω being a complete probability space characterized by $(\Omega, \mathcal{F}, \mathbb{P})$.

For the optimization problem (Opt), they establish two key regularity assumptions necessary for the analysis:

Assumption 1: The function $F(x,\omega)$ is continuously differentiable with respect to x for almost every outcome ω in Ω. This assumption ensures that the derivative with respect to x exists and varies smoothly, except possibly on a set of probability zero. These smoothness assumptions can be relaxed but they contend that it is "cumbersome for analysis".

Assumption 2: The gradient of F, with respect to x, is uniformly bounded in the L^2 norm. Specifically, we assume that:

$$\mathbb{E}[\|\nabla F(x;\omega)\|^2] \leq V^2,$$

for all $x \in X$ and some finite constant $V \geq 0$. This condition implies that the expected square of the norm of the gradient does not exceed V^2, which controls the variability and ensures stability in the optimization process. It also holds trivially if F is uniformly Lipschitz.

3.3 Variational Coherence and Definitions of Quasi Convexity

Variational coherence (VC) is a condition that establishes a geometric and analytical structure in the space of feasible solutions for optimization problems. This condition, as defined in:

$$\langle \nabla f(x), x - x^* \rangle \geq 0,$$

states that the gradient of the objective function f at any point x forms a non-negative angle with the vector pointing towards any optimal point x^*. This geometric interpretation is crucial because it aligns with the principle of gradient methods, where the direction of the

gradient points towards the steepest ascent and, by negation, the descent direction indicates the path towards local minima in optimization landscapes.
In the context of convex optimization, if f is convex and ∇f is monotone, then:

$$\langle \nabla f(x) - \nabla f(x_0), x - x_0 \rangle \geq 0 \text{ for all } x, x_0 \in X.$$

By the first-order optimality conditions, we have:

$$\langle f(x^*), x - x^* \rangle \geq 0 \text{ for all } x \in X.$$

Through monotonicity:

$$\langle \nabla f(x), x - x^* \rangle \geq \langle \nabla f(x^*), x - x^* \rangle \geq 0 \text{ for all } x \in X, x^* \in X^*.$$

By convexity, equality in the above holds if and only if $x \in X^*$. Thus, convex programs inherently satisfy the (VC) condition.
For quasi-convex objectives, the analysis extends:

$$f(x_0) \leq f(x) \implies \langle \nabla f(x), x_0 - x \rangle \leq 0.$$

3.4 Algorithm

Algorithm 1 Stochastic Mirror Descent (SMD)

Require: Mirror map $Q : Y \to X$; step-size sequence $\{\gamma_n > 0\}_{n=1}^{\infty}$

$Y \leftarrow$ choose $Y \in Y \equiv V^*$ ▷ Initialization
for $n = 1, 2, \ldots$ **do**
  $X \leftarrow Q(Y)$ ▷ Set state
  Draw $\omega \in \Omega$ ▷ Gradient sample
  $\hat{v} \leftarrow -\nabla F(X; \omega)$ ▷ Get oracle feedback
  $Y \leftarrow Y + \gamma_n \hat{v}$ ▷ Update score variable
end for
return X ▷ Output

3.5 Intuition and General Insight

SMD differentiates from deterministic optimization algorithms by directly using stochastic gradients, which are approximations of the true gradients derived from samples or simulations. This direct use of noisy gradients allows the algorithm to adapt to scenarios where only imperfect information is available, addressing the variability in the optimization path due to the randomness of the data.

Instead of operating directly on the decision variables, SMD works in the dual space. The dual space's properties to manage constraints and problem geometry more efficiently as it does a good job of transforming complex constraints into simpler forms which generally translates to smoother processes.

The mirror map Q is essential for linking updates in the dual space back to the primal space, where the actual decision variables are situated. After updating the dual variables

(score variable Y), Q translates these adjustments back to the primal space as new points X. This ensures that updates maintain the structural and constraint integrity of the original problem.

Ultimately, SMD improves robustness by averaging the effects of stochastic gradients over time. This method smooths out the noise inherent in gradient estimates through continuous updates of the dual variable, followed by re-mapping to the primal space. Such averaging acts as a noise-dampening mechanism, aiding the algorithm in converging towards a robust solution despite the stochastic nature of the data.

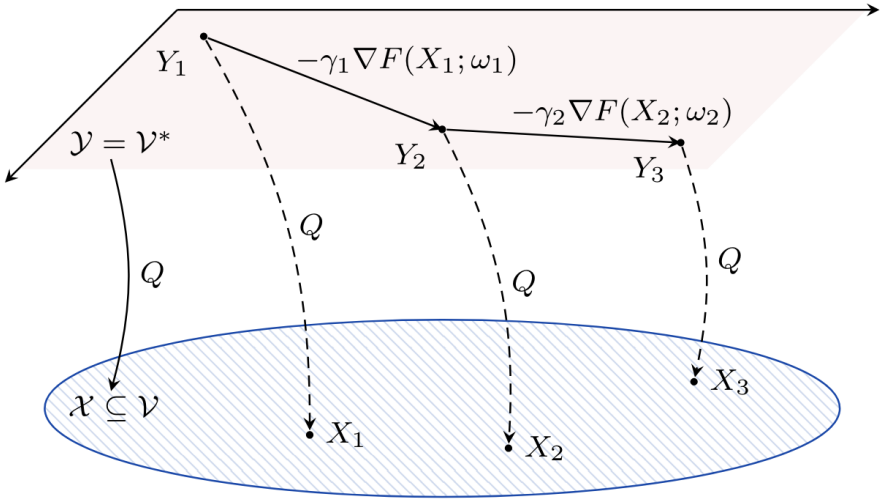

Figure 3: Schematic representation of the Stochastic Mirror Descent (Algorithm 1).

3.6 Global Convergence

Ultimately we are interested in the convergence properties of this algorithm. In this section, we will provide a sketch of the global convergence proof.

We start by letting x^* be a minimum point of the optimization problem (Opt), and define $F_n = F(x^*, Y_n)$, where Y_n represents a sequence of random variables influenced by the iterations of the SMD algorithm.
The function F_n is updated according to the SMD algorithm, leading to the next step:

$$F_{n+1} = F(x^*, Y_{n+1}) = F(x^*, Y_n + \gamma_n \hat{v}_n)$$

Using the properties of the function F and the update mechanism, we can write:

$$F_{n+1} \leq F_n + \gamma_n h(\hat{v}_n, X_n - x^*) + \frac{\gamma_n^2}{2} K \|\hat{v}_n\|^2$$

where $\hat{v}_n$ represents the gradient estimate and K is a constant from the Lipschitz condition.

Because Y_n is predictable with respect to the sigma-algebra generated by the past, F_n is adapted to this filtration. Taking conditional expectations and leveraging the martingale properties, the inequality becomes:

$$\mathbb{E}[F_{n+1}|F_n] \leq F_n + \frac{\gamma_n^2}{2} KV^2$$

where V represents a bound on the expected square of the gradient norm, and F_n denotes the filtration up to step n.
Applying Gladyshev's Lemma (Lemma 4.3, See Appendix), which deals with nonnegative random variables where the expected increment is controlled by a summable series, given the setup:

$$\mathbb{E}[a_{n+1}|a_1, \ldots, a_n] \leq (1 + \delta_n)a_n + \epsilon_n$$

with $\sum \delta_n < \infty$ and $\sum \epsilon_n < \infty$, and $a_n = F_n$ in our case, the conditions of the lemma are satisfied due to the boundedness of γ_n^2.
Using the lemma, F_n converges almost surely to some finite limit F_∞, and since F_n measures the discrepancy at x^*, this implies $F(x^*, Y_n) \to 0$. Consequently, $X_n = Q(Y_n)$ converges to x^* almost surely, where Q represents the projection or mapping used in the SMD, confirming that X_n approaches a minimum point of (Opt).

3.7 Local Convergence

In this section, our goal is to extend the convergence analysis of the previous section to account for optimization problems that are only "locally" coherent. In the paper, they are defined as follows.

Let x^* be a local minimizer within C, the coherent set under consideration. Define the sequence $\{X_n\}$ as the iterates generated by the algorithm starting from $X_1 \in U$, an open neighborhood of x^*. Assume $\{\gamma_n\}$ is the step-size sequence.

Consider the stochastic process:

$$\xi_n = \langle \nabla f(X_n) - \nabla F(X_n; \omega_n), X_n - x^* \rangle$$

Summing the martingale sequence given by $\gamma_n \xi_n$ and recognizing the submartingale behavior of the squared gradient norms, we decompose the error into two parts:

$$\text{Martingale term: } S_n = \sum_{k=1}^{n} \gamma_k \xi_k,$$

$$\text{Submartingale term: } R_n = \sum_{k=1}^{n} \gamma_k^2 \|v_k\|^2.$$

By applying Doob's maximal inequality, we control both terms as follows:

$$P\left(\sup_{k \leq n} S_k \geq \epsilon\right) \leq \frac{\delta}{2}, \quad P\left(\sup_{k \leq n} R_k \geq \epsilon\right) \leq \frac{\delta}{2}.$$

The conditions of the inequalities are satisfied through careful selection of $\{\gamma_n\}$, ensuring $\sum_{n=1}^{\infty} \gamma_n^2$ is sufficiently small.

To guarantee convergence, we combine the bounds on the martingale and submartingale sequences to conclude:

$$P\left(\sup_n \max\{S_n, R_n\} \leq \epsilon\right) \geq 1 - \delta.$$

Given X_1 starts in U, the function $F(x^*, Y_1)$ begins below a threshold ϵ. By the established inequalities and the properties of the processes, the iterates $\{X_n\}$ are guaranteed to remain close to x^* and within the neighborhood U with high probability.

Using the recursive relation and the control over S_n and R_n, we conclude that $\{X_n\}$ converges to x^* in U with at least probability $1 - \delta$. This follows by ensuring each iterate does not exceed the bounds of the defined errors, leading to convergence to the minimizer.

3.8 Numerical Experiments

Ultimately, the paper makes a lot of significant claims about SMD being able to globally convergence on non-convex problems. In the paper, they provide various numerical experiments to prove this.

In this section, we will try to see if the SMD converges on the infamous Rosenbrock setup.

The setup is ultimately a quadratic:

$$f(x) = \frac{1}{2} \sum_{i,j=1}^{d} Q_{ij} x_i x_j + \sum_{i=1}^{d} b_i x_i$$

where Q is a negative-definite matrix. This setup is interesting because, despite Q being negative-definite—which would typically suggest a concave landscape—when combined with the constraints of the unit simplex, the problem does not simply remain trivial. It also has a well defined global minimum we can hearken to.

Furthermore, the function's relationship to the maximum weight clique problem, known for its computational complexity, lets us test SMD under challenging conditions. If SMD can quickly find reasonable solutions here, it's a good sign that it can handle other complex problems effectively.

We ran the experiments on the two dimensional Rosenbrock and the three dimensional Rosenbrock.

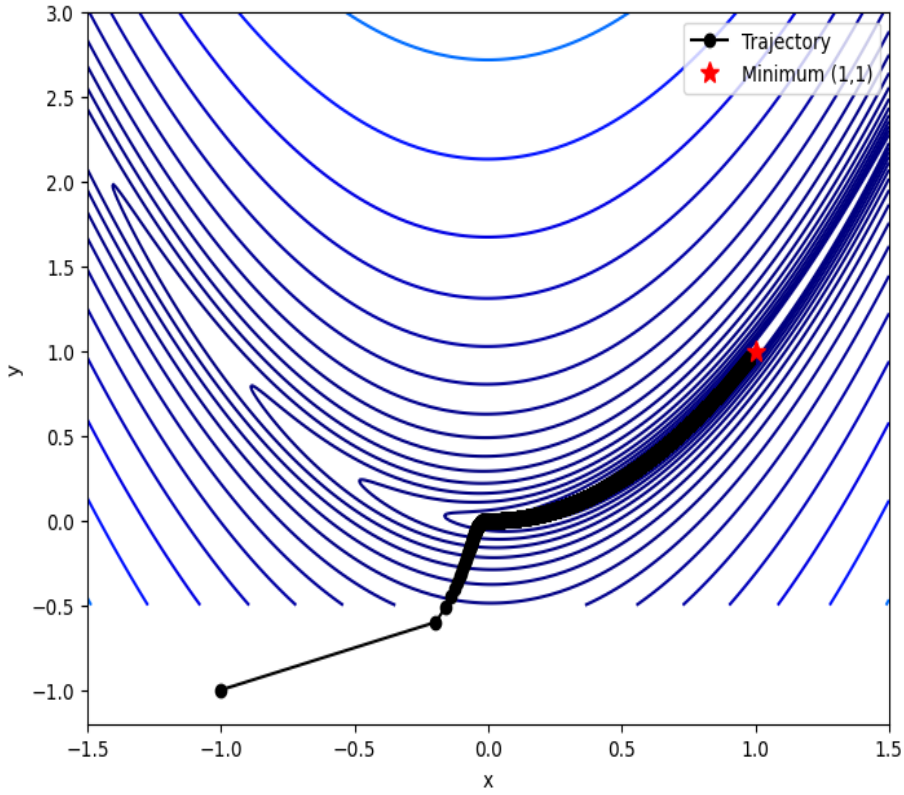

Figure 4: Stochastic Convergence on 2 Dimensional Rosenbrock

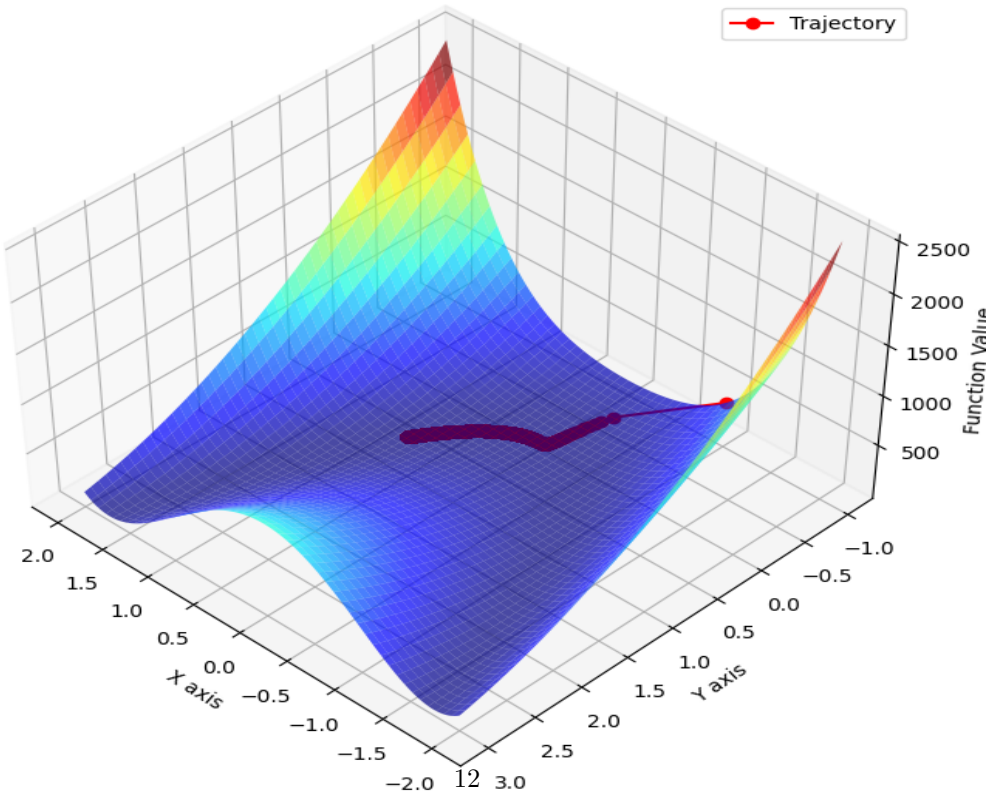

Figure 5: Stochastic Convergence on 3 Dimensional Rosenbrock

In Figure 4, the trajectory from the starting point on the left towards the global minimum at (1,1) is smooth and direct. This indicates that the algorithm effectively handles the Rosenbrock function's notorious narrow valley. The path almost hitting the mark at (1,1) suggests that the algorithm isn't just wandering around but is pretty targeted in its approach.

Moving up a dimension in Figure 5, things get more complex with the added dimension, yet the algorithm still seems to hold its own. The trajectory here meanders towards a low point, and the color gradient from blue to red nicely illustrates the function values dropping along the way. This behavior is encouraging because it shows that the algorithm can scale up and still perform well even as the problem space expands.

We also wanted to show the convergence rates so we decided to include the rates in the 2 dimensional case which, as it turns out, were incredibly encouraging.

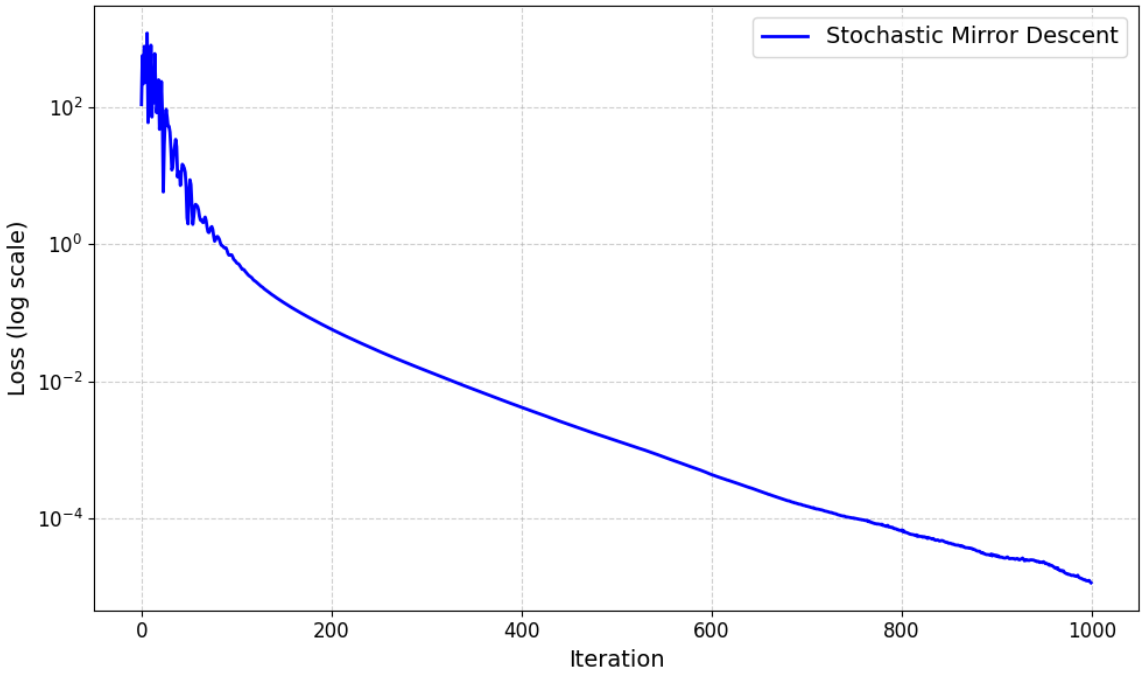

Figure 6: Convergence of SMD on Rosenbrock

3.9 Discussion and Further Developments

The convergence behavior of SMD in non-convex scenarios, as presented in the second paper, highlights a significant advancement in understanding how stochastic methods can adapt to complex optimization landscapes. Unlike traditional gradient descent methods that may falter in such environments due to local minima and saddle points, SMD incorporates randomness in a manner that potentially allows it to escape these traps.

Our numerical experiments with the Rosenbrock function fundamentally agree with this point. SMD not only adapted its approach in the face of a non-trivial, non-convex landscape but also demonstrated robustness by converging towards the global minimum effectively.

However, the findings also raise some contentions, particularly around the optimality and efficiency of SMD in broader non-convex settings. The reliance on specific step-size schedules and initial conditions may limit the general applicability and scalability of the approach and further research should definitely be done. Additionally, while the Rosenbrock function serves as a useful testbed due to its well-understood properties and challenging topology, real-world applications often present even more complex landscapes and higher-dimensional spaces that may not be adequately represented by this or similar benchmark functions. Extending the analysis to include a wider variety of non-convex functions, would lead to more generative points of insight.

References

Yunwen Lei and Ding-Xuan Zhou. Convergence of online mirror descent. *Applied and Computational Harmonic Analysis*, 48(1):343–373, 2020. ISSN 1063-5203.

Zhengyuan Zhou, Panayotis Mertikopoulos, Nicholas Bambos, Stephen Boyd, and Peter Glynn. On the convergence of mirror descent beyond stochastic convex programming, 2018.

Appendix A. First paper theoretical proof and codes

A.1 Proof sketch of the theorem

In this section, we will provide a proof sketch of the main results from the first paper. We will strive to formulate our proof sketch based on our understanding of the original paper. However, there may be certain lemmas, conditions, and propositions that we have not fully comprehended yet. Therefore, we will precisely follow the original paper to prove those parts.

We first are going to prove Theorem 2. We want to show the necessity i.e. $\lim_{t\to\infty}\mathbb{E}_{z_1,\ldots,z_{t-1}}[D_\Psi(w^*,w_t)]=0$ implies condition (10). By the σ_Ψ-strong convexity of Ψ, we have $\|w^*-w_t\|^2\le\frac{2}{\sigma_\Psi}D_\Psi(w^*,w_t)$. So the condition $\lim_{t\to\infty}\mathbb{E}_{z_1,\ldots,z_{t-1}}[D_\Psi(w^*,w_t)]=0$ implies the $\lim_{t\to\infty}\mathbb{E}_{z_1,\ldots,z_{t-1}}[\|w^*-w_t\|^2]=0$. Then, by continuity of $\nabla\Psi$ (embed with L_ψ-smooth assumption) we have

$$\lim_{t\to\infty}\mathbb{E}_{z_1,\ldots,z_{t-1}}[\|\nabla\Psi(w_t)-\nabla\Psi(w^*)\|_*]=0 \tag{16}$$

By OMD algorithm (4), we have

$$\eta_t\|\nabla_w[f(w_t,z_t)]\|_*=\|\nabla\Psi(w_t)-\nabla\Psi(w_{t+1})\|_*\le\|\nabla\Psi(w_t)-\nabla\Psi(w^*)\|_*+\|\nabla\Psi(w_{t+1})-\nabla\Psi(w^*)\|_* \tag{17}$$

Due to positive variance, we have $\inf_{w\in W}\mathbb{E}_Z[\|\nabla_w[f(w,Z)]\|_*]>0$. Denote $\sigma=\inf_{w\in W}\mathbb{E}_Z[\|\nabla_w[f(w,Z)]\|_*]$, then by taking expectation of equation (17) respect to z_t, yields

$$\eta_t\sigma\le\eta_t\mathbb{E}[\|\nabla_w[f(w_t,z_t)]\|_*]\le\|\nabla\Psi(w_t)-\nabla\Psi(w^*)\|_*+\mathbb{E}_{z_t}[\|\nabla\Psi(w_{t+1})-\nabla\Psi(w^*)\|_*] \tag{18}$$

$$\eta_t\sigma\le\mathbb{E}_{z_1,\ldots z_{t-1}}[\|\nabla\Psi(w_t)-\nabla\Psi(w^*)\|_*]+\mathbb{E}_{z_1,\ldots,z_t}[\|\nabla\Psi(w_{t+1})-\nabla\Psi(w^*)\|_*]. \tag{19}$$

Since left hand goes to zero as $t\to\infty$ and $\sigma>0$, we have $\lim_{t\to\infty}\eta_t=0$.

We then try to prove $\sum_{t=1}^{\infty}\eta_t=\infty$. We first introduce the following lemma

Lemma 4 *The following identity holds for $t\in\mathbb{N}$*

$$\mathbb{E}_{z_t}[D_\Psi(w^*,w_{t+1})]-D_\Psi(w^*,w_t)=\eta_t\langle w^*-w_t,\nabla F(w_t)\rangle+\mathbb{E}_{z_t}[D_\Psi(w_t,w_{t+1})]. \tag{20}$$

Proof By the points lemma of Bregman distance, we have the following identity

$$\begin{aligned}
&D_\Psi(w^*,w_{t+1})-D_\Psi(w^*,w_t) && (21)\\
=&-D_\Psi(w_{t+1},w_t)+\langle w^*-w_{t+1},\nabla\Psi(w_t)-\nabla\Psi(w_{t+1})\rangle && (22)\\
=&-D_\Psi(w_{t+1},w_t)+\langle w^*-w_t,\nabla\Psi(w_t)-\nabla\Psi(w_{t+1})\rangle+\langle w_t-w_{t+1},\nabla\Psi(w_t)-\nabla\Psi(w_{t+1})\rangle && (23)\\
=&-D_\Psi(w_{t+1},w_t)+\eta_t\langle w-w_t,\nabla_w[f(w_t,z_t)]\rangle+\langle w_t-w_{t+1},\nabla\Psi(w_t)-\nabla\Psi(w_{t+1})\rangle && (24)\\
=&D_\Psi(w_t,w_{t+1})+\eta_t\langle w-w_t,\nabla_w[f(w_t,z_t)]\rangle && (25)
\end{aligned}$$

Taking expectations $\mathbb{E}_{z_t}$ on both sides and noting that w_t is independent of z_t, we have the stated identity in the lemma. ■

By L smoothness of F and σ_Ψ-strongly convex of Ψ, we have

$$\langle w^*-w_t,\nabla F(w_t)\rangle\ge-L\|w^*-w_t\|^2\ge-\frac{2L}{\sigma_\Psi}D_\Psi(w^*,w_t). \tag{26}$$

Then we apply the lemma 4 and take expectations on both sides give

$$\mathbb{E}_{z_1,\dots z_t}[D_\Psi(w^*, w_{t+1})] \geq (1 - a\eta_t)\mathbb{E}_{z_1,\dots,z_{t-1}}[D_\Psi(w^*, w_t)] + \mathbb{E}_{z_1,\dots,z_t}[D_\Psi(w_t, w_{t+1})], \quad (27)$$

where $a = 2L\sigma_\Psi^{-1}$.

Since $\lim_{t\to\infty} \eta_t = 0$, we cab find some integer $t_0 \in \mathbb{N}$ such that $\eta_t \leq (3a)^{-1}$ for $t \geq t_0$. Noting $1 - \alpha\eta \geq \exp -2\alpha\eta$ when $\eta \in (0, (3a)^{-1}]$ and $\mathbb{E}_{z_1,\dots,z_t}[D_\Psi(w_t, w_{t+1})] \geq 0$, we have following

$$\mathbb{E}_{z_1,\dots,z_t}[D_\Psi(w^*, w_{t+1})] \geq \exp(-2a\eta_t)\mathbb{E}_{z_1,\dots z_{t-1}}[D_\Psi(w^*, w_t)], \quad \forall t \geq t_0. \quad (28)$$

Applying this inequality iteratively yields

$$\mathbb{E}_{z_1,\dots,z_T}[D_\Psi(w^*, w_{T+1})] \geq \prod_{t=t_0+1}^{T} \exp(-2a\eta_t)\mathbb{E}_{z_1,\dots,z_{t_0}}[D_\Psi(w^*, w_{t_0+1})]. \quad (29)$$

$$= \exp(-2\alpha \sum_{t=t_0+1}^{T})\eta_t \mathbb{E}_{z_1,\dots,z_{t_0}}[D_\Psi(w^*, w_{t_0+1})]. \quad (30)$$

We have $\mathbb{E}_{z_1,\dots,z_{t_0}}[D_\Psi(w^*, w_{t_0+1}) > 0$ and $\lim_{T\to\infty} \mathbb{E}_{z_1,\dots,z_T}[D_\Psi(w^*, w_{T+1})] = 0$. Hence $\sum_{t=1}^{\infty} \eta_t = \infty$.

We then will prove the sufficiency. Here, we first introduce two lemmas from an original paper without further proof:

Lemma 5 *Let $g : \mathbb{R}^d \to \mathbb{R}$ be continuous and convex. Let $\beta > 0$. Then g is β-strongly convex with respect to the norm $\|\cdot\|$ if and only if g^* is $\frac{1}{\beta}$-strongly smooth with respect to the dual norm $\|\cdot\|_*$.*

If g is differentiable and strongly convex, then there holds

$$D_g(w, \tilde{w}) = D_{g^*}(\nabla g(\tilde{w}), \nabla g(w)), \quad \forall w, \tilde{w} \in \mathcal{W}. \quad (31)$$

Lemma 6 *Let $\alpha \in (0, 1]$ and $g : \mathcal{W} \to \mathbb{R}$ be a differentiable and convex function. If there exists some constant $L > 0$ such that*

$$D_g(w, \tilde{w}) \leq \frac{L}{1+\alpha}\|w - \tilde{w}\|^{1+\alpha}, \quad \forall w, \tilde{w} \in \mathcal{W}, \quad (32)$$

then we have

$$\frac{2L^{-\frac{1}{\alpha}}\alpha}{1+\alpha}\|\nabla g(w) - \nabla g(\tilde{w})\|_*^{\frac{1+\alpha}{\alpha}} \leq \langle w - \tilde{w}, \nabla g(w) - \nabla g(\tilde{w}), \quad \forall w, \tilde{w} \in \mathcal{W}. \quad (33)$$

. By lemma 5, and the σ_Ψ-strong convexity of Ψ, we have

$$\mathbb{E}_{z_t}\big[D_{\Psi^*}(\nabla\Psi(w_{t+1}), \nabla\Psi(w_t))\big] \leq \frac{1}{2\sigma_\Psi}\mathbb{E}_{z_t}\big[\|\nabla\Psi(w_{t+1}) - \nabla\Psi(w_t)\|_*^2\big]$$

$$= \frac{\eta_t^2}{2\sigma_\Psi}\mathbb{E}_{z_t}\big[\|\nabla_w[f(w_t, z_t)]\|_*^2\big]. \quad (34)$$

We bound $\big[\|\nabla_w[f(w_t,z_t)]\|_*^2\big]$ by $2\big[\|\nabla_w[f(w_t,z_t)]-\nabla_w[f(w^*,z_t)]\|_*^2\big]+2\big[\|\nabla_w[f(w^*,z_t)]\|_*^2\big]$. Then we apply Lemma 6 with $w=w^*, \tilde{w}=w_t, g=f(\cdot,z_t)$ and $\alpha=1$. By the L-strong smoothness of $f(\cdot,z)$, we know that

$$\mathbb{E}_{z_t}\Big[\|\nabla_w[f(w_t,z_t)]-\nabla_w[f(w^*,z_t)]\|_*^2\Big] \le L\mathbb{E}_{z_t}\Big[\big\langle w_t-w^*, \nabla_w[f(w_t,z_t)]-\nabla_w[f(w^*,z_t)]\big\rangle\Big] \tag{35}$$

$$= L\langle w^*-w_t, \nabla F(w^*)-\nabla F(w_t)\rangle. \tag{36}$$

Then by lemma 4 we have

$$\mathbb{E}_{z_t}[D_\Psi(w^*,w_{t+1})]-D_\Psi(w^*,w_t) \le \\ -\left(1-\frac{L\eta_t}{\sigma_\Psi}\right)\eta_t\langle w^*-w_t,\nabla F(w^*)-\nabla F(w_t)\rangle+\frac{\eta_t^2}{\sigma_\Psi}\mathbb{E}_{z_t}\big[\|\nabla_w[f(w^*,z_t)]\|_*^2\big]. \tag{37}$$

Since $\lim_{t\to\infty}\eta_t=0$, there exists some $t_1\in\mathbb{N}$ such that $\frac{L}{\sigma_\Psi}\eta_t\le\frac{1}{2}$ for $t\ge t_1$ which implies

$$\mathbb{E}_{z_t}[D_\Psi(w^*,w_{t+1})]-D_\Psi(w^*,w_t) \le \\ -\frac{\eta_t}{2}\langle w^*-w_t,\nabla F(w^*)-\nabla F(w_t)\rangle+\frac{\eta_t^2}{\sigma_\Psi}\mathbb{E}_{z_t}\big[\|\nabla_w[f(w^*,z_t)]\|_*^2\big]. \tag{38}$$

With the condition (5), we have

$$\mathbb{E}_{z_t}[D_\Psi(w^*,w_{t+1})] \le D_\Psi(w^*,w_t)-\frac{\eta_t\sigma_F}{2}D_\Psi(w^*,w_t)+b\eta_t^2, \tag{39}$$

where $b=\frac{1}{\sigma_\Psi}\mathbb{E}_Z[\|\nabla_w[f(w^*,Z)]\|_*^2]$. By taking expectation, we have

$$\mathbb{E}_{z_1,\dots,z_t}[D_\Psi(w^*,w_{t+1})] \le (1-\frac{\eta_t\sigma_F}{2})\mathbb{E}_{z_1,\dots,z_{t-1}}[D_\Psi(w^*,w_t)]+b\eta_t^2. \tag{40}$$

Let $A_t=\mathbb{E}_{z_1,\dots,z_{t-1}}[D_\Psi(w^*,w_t)]$, we have

$$A_{t+1}\le(1-\frac{\eta_t\sigma_F}{2})A_t+b\eta_t^2 \tag{41}$$

We now try to show $\lim_{t\to\infty}A_t=0$. We claim the following proposition

Proposition 7 *For any $\gamma\in(0,1)$ the following arguments hold when we have condition* (10)

$$\sup t\in\mathbb{N}: A_t\le\gamma=\infty \tag{42}$$

Proof We prove this by contradiction. Since $\lim_{t\to\infty}\eta_t=0$, we have

$$\eta_t\le\frac{\sigma_F\gamma}{4b},\quad \forall t\ge t_\gamma \tag{43}$$

Suppose the above argument does not hold, which implies there exists $t'_\gamma\ge t_\gamma$ such that

$$A_t\ge\gamma, \forall t\ge t'_\gamma. \tag{44}$$

Additional with the inequality (41), we have

$$A_{t+1} \leq (1 - \frac{\eta_t \sigma_F}{2}) A_t + b\eta_t^2 \tag{45}$$
$$= b\eta_t^2 - \frac{\eta_t \sigma_F}{2} A_t + A_t \tag{46}$$
$$\leq b\frac{\sigma_F \gamma}{4b}\eta_t - \frac{\eta_t \sigma_F}{2} A_t + A_t \tag{47}$$
$$\leq \frac{\sigma_F \eta_t}{4} A_t - \frac{\eta_t \sigma_F}{2} A_t + A_t \tag{48}$$
$$\leq -\frac{\sigma_F \eta_t}{4} A_t + A_t \tag{49}$$
$$\leq -\frac{\sigma_F \eta_t}{4}\gamma + A_t \tag{50}$$

By applying the above inequality iteratively, we obtain

$$A_T \leq -\frac{\sigma_F \gamma}{4} \sum_{t=t'_\gamma+1}^{T} \eta_t + A_{t'_\gamma} \tag{51}$$

Then we have

$$\lim_{T\to\infty} A_T \leq A_{t'_\gamma} - \frac{\sigma_F \gamma}{4} \lim_{T\to\infty} \sum_{t=t'_\gamma+1}^{T} \eta_t = -\infty, \tag{52}$$

which contradicts to the fact

$$A_t \geq \gamma, \forall t \geq t'_\gamma. \tag{53}$$

Hence, we complete the proof. ■

Now we are going to prove the sufficient condition through induction. With proposition 7, there exists $t_\gamma" > t_\gamma$, such that $A_t \leq \gamma, \forall t > t_\gamma"$.

Then

$$A_{t+1} \leq b\eta_t^2 - \frac{\eta_t \sigma_F}{2} A_t + A_t \tag{54}$$
$$\leq \max\left\{ b(\frac{\sigma_F \gamma}{4b})^2 - \frac{\sigma_F^2}{8b}\gamma A_t + A_t, A_t \right\} \tag{55}$$
$$\leq \max\left\{ (1 - \frac{\sigma_F^2 \gamma}{8b}) A_t + \frac{\sigma_F^2 \gamma^2}{16}, A_t \right\} \tag{56}$$
$$\leq \begin{cases} \max\left\{ \frac{\sigma_F^2 \gamma^2}{16b}, A_t \right\} & \text{if } 1 - \frac{\sigma_F^2 \gamma}{8b} < 0 \\ \max\left\{ \gamma - \frac{\sigma_F^2 \gamma^2}{16b}, A_t \right\} & \text{if } 1 - \frac{\sigma_F^2 \gamma}{8b} \geq 0 \end{cases} \tag{57}$$

Let $\gamma \in (0, \frac{8b}{\sigma_F^2}]$, then we always have $A_{t+1} \leq \max\left\{ \gamma - \frac{\sigma_F^2 \gamma^2}{16b}, A_t \right\} \leq \gamma$. Therefore, we have $\lim_{t\to\infty} A_t \leq \gamma$. Since γ can be arbitrary close to zero, we prove the $\lim_{t\to\infty} A_t = \lim_{t\to\infty} [\mathbb{E}_{z_1,\ldots,z_{t-1}}[D_\Psi(w^*, w_t)] = 0$.

Now we try to prove the converge rate give $\eta_t = \frac{4}{(t+1)\sigma_F}$.
We have

$$A_{t+1} \leq A_t - \frac{2}{t+1}A_t + \frac{16b}{(t+1)^2\sigma_F^2} \tag{58}$$

$$t(t+1)A_{t+1} \leq (t-1)tA_t + \frac{16b}{\sigma_F^2} \tag{59}$$

Applying this iteratively, we obtain

$$(T-1)TA_T \leq (t_1-1)t_1A_{t_1} + \frac{16b(T-t_1)}{\sigma_F^2} \tag{60}$$

$$A_T \leq \frac{(t_1-1)t_1A_{t_1}}{(T-1)T} + \frac{16b}{T\sigma_F^2} \tag{61}$$

$$\mathbb{E}_{z_1,\ldots,z_{T-1}}[D_\Psi(w^*, w_T) \leq \frac{(t_1-1)t_1\mathbb{E}_{z_1,\ldots,z_{t_1-1}}[D_\Psi(w^*, w_{t_1})}{(T-1)T} + \frac{16b}{T\sigma_F^2} \tag{62}$$

The proof for the zero variance case is very similar to the proof of the positive variance case, where we have the relationship

$$A_{t+1} \leq (1 - \frac{\eta_t\sigma_F}{2})A_t \tag{63}$$

from where we immediately get linear converge rate

A.2 Codes for empirical results

```
import numpy as np
import jax.numpy as jnp
import jax
import matplotlib.pyplot as plt
import cvxpy as cp

np.random.seed(0)

n_dim = 50
w_true = np.random.uniform(low=0, high=1, size=n_dim)
w_true = w_true / np.sum(w_true)
num_data_no_noisy = 1000
num_data_noisy = 5000

def generate_dataset(num_data=1000, noise_scale=0):
    x = np.random.multivariate_normal(
        mean=np.zeros(n_dim), cov=np.eye(n_dim), size=num_data)
    y = x @ w_true + noise_scale * np.random.randn(num_data)
    return x, y

xs_no_noise, ys_no_nonise = generate_dataset(num_data=num_data_no_noisy, noise_scale=0)
```

```
xs_noise, ys_nonise = generate_dataset(num_data=num_data_noisy, noise_scale=0.2)
w0 = np.random.uniform(low=0, high=1, size=n_dim)
w0 = w0 / np.sum(w0)

def get_KL(w_true, w_current):
    return np.sum(w_true * np.log(w_true / w_current))

print("Ground Truth w:", w_true)
print("w0:", w0)
print("The KL between inital guess and ground true:", get_KL(w_true=w_true, w_current=w0))

def get_function_value(w, x, y):
    return (jnp.dot(w,x) - y)**2

def project_to_simplex_cvxpy(w):
    n = len(w)
    w_var = cp.Variable(n)
    objective = cp.Minimize(cp.sum_squares(w_var - w))
    constraints = [cp.sum(w_var) == 1, w_var >= 0]
    prob = cp.Problem(objective, constraints)
    prob.solve()
    return w_var.value

def projected_gradient_descent(xs, ys, w0, step_size=0.1, decay=True, decay_rate=0.5):
    w = w0
    func_values = []
    KLs_to_minimizer = [get_KL(w_true=w_true, w_current=w)]

    for i, (x, y) in enumerate(zip(xs, ys)):

        func_value, grad = jax.value_and_grad(get_function_value)(w, x, y)
        func_values.append(func_value)

        if decay:
            alpha = step_size / (i + 1)**decay_rate
        else:
            alpha = step_size

        w = w - alpha * grad
        w = project_to_simplex_cvxpy(w)

        KLs_to_minimizer.append(get_KL(w_true=w_true, w_current=w))

    return w, func_values, KLs_to_minimizer
```

```
def mirror_descent(xs, ys, w0, step_size=0.1, decay=True, decay_rate=0.5):
    w = w0
    func_values = []
    KLs_to_minimizer = [get_KL(w_true=w_true, w_current=w)]

    for i, (x, y) in enumerate(zip(xs, ys)):

        func_value, grad = jax.value_and_grad(get_function_value)(w, x, y)
        func_values.append(func_value)

        if decay:
            alpha = step_size / (i + 1)**decay_rate
        else:
            alpha = step_size

        # Mirror descent update
        w = w * np.exp(-alpha * grad)
        w = w / np.sum(w)

        KLs_to_minimizer.append(get_KL(w_true=w_true, w_current=w))

    return w, func_values, KLs_to_minimizer

w_md, func_values_md, KLs_to_minimizer_md = mirror_descent(
xs=xs_no_noise, ys=ys_no_nonise, w0=w0, step_size=0.5, decay=False)
w_pgd, func_values_pgd, KLs_to_minimizer_pgd = projected_gradient_descent(
xs=xs_no_noise, ys=ys_no_nonise, w0=w0, step_size=0.001, decay=False)

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.plot(func_values_md, label="mirror desecent")
ax.plot(func_values_pgd, label="projected gradient desecent")
ax.legend()
ax.set_title("func value with no noisy data")
ax.set_xlabel("iterations")

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.plot(KLs_to_minimizer_md, label="mirror descent")
ax.plot(KLs_to_minimizer_pgd, label="projected gradient descent")
ax.legend()
ax.set_yscale("log")
ax.set_title("KL to minimizer with no noisy data")
ax.set_xlabel("iterations")
```

```
plt.show()

w_md, func_values_md, KLs_to_minimizer_md = mirror_descent(
xs=xs_noise, ys=ys_nonise, w0=w0, step_size=0.1, decay=False)
w_md_linear_decay, func_values_md_linear_decay, KLs_to_minimizer_md_linear_decay = \
mirror_descent(xs=xs_noise, ys=ys_nonise, w0=w0, step_size=1, decay=True, decay_rate=1)
w_md_square_root_decay, func_values_md_square_root_decay, \
KLs_to_minimizer_md_square_root_decay = mirror_descent(
xs=xs_noise, ys=ys_nonise, w0=w0, step_size=0.5, decay=True, decay_rate=0.5)

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.plot(func_values_md, label="constant step size")
ax.plot(func_values_md_linear_decay, label="linear decay step size")
ax.plot(func_values_md_square_root_decay, label="square root decay step size")
ax.legend()
ax.set_title("func value with noisy data")
ax.set_xlabel("iterations")

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.plot(KLs_to_minimizer_md, label="constant step size")
ax.plot(KLs_to_minimizer_md_linear_decay, label="linear rate decay step size")
ax.plot(KLs_to_minimizer_md_square_root_decay, label="square root decay step size")
ax.legend()
ax.set_title("KL to minimizer with noisy data")
ax.set_xlabel("iterations")

plt.show()
```

Appendix B.

In this appendix we prove the following theorem from Section 6.2:

Theorem *Let u, v, w be discrete variables such that v, w do not co-occur with u (i.e., $u \neq 0 \;\Rightarrow\; v = w = 0$ in a given dataset $\mathcal{D}$). Let N_{v0}, N_{w0} be the number of data points for which $v = 0, w = 0$ respectively, and let I_{uv}, I_{uw} be the respective empirical mutual information values based on the sample $\mathcal{D}$. Then*

$$N_{v0} \;>\; N_{w0} \;\;\Rightarrow\;\; I_{uv} \;\leq\; I_{uw}$$

with equality only if u is identically 0. ■

Proof. We use the notation:

$$P_v(i) \;=\; \frac{N_v^i}{N}, \quad i \neq 0; \quad P_{v0} \;\equiv\; P_v(0) \;=\; 1 - \sum_{i \neq 0} P_v(i).$$

These values represent the (empirical) probabilities of v taking value $i \neq 0$ and 0 respectively. Entropies will be denoted by H. We aim to show that $\frac{\partial I_{uv}}{\partial P_{v0}} < 0$....

Appendix C. Code for Empirical Results (Paper 2)

```
import numpy as np
import matplotlib.pyplot as plt

def rosenbrock(x, y, a=1, b=100):
    return (a - x)**2 + b*(y - x**2)**2

def rosenbrock_grad(x, y, a=1, b=100):
    grad_x = -2*(a - x) - 4*b*x*(y - x**2)
    grad_y = 2*b*(y - x**2)
    return np.array([grad_x, grad_y])

def smd_rosenbrock(init_pos, iterations, step_size_func, a=1, b=100):
    trajectory = [init_pos]
    current_pos = np.array(init_pos)

    for i in range(1, iterations):
        grad = rosenbrock_grad(current_pos[0], current_pos[1], a, b)
        step_size = step_size_func(i)
        current_pos = current_pos - step_size * grad
        trajectory.append(current_pos)

    return np.array(trajectory)

def step_size_func(n, gamma_0=0.001):
    return gamma_0 / np.sqrt(n)

init_pos = (-1, -1)
iterations = 10000000

trajectory = smd_rosenbrock(init_pos, iterations, step_size_func)
x_vals = np.linspace(-1.5, 1.5, 400)
y_vals = np.linspace(-0.5, 3, 400)
X, Y = np.meshgrid(x_vals, y_vals)
Z = rosenbrock(X, Y)

fig, ax = plt.subplots(figsize=(8, 6))
CS = ax.contour(X, Y, Z, levels=np.logspace(-0.5, 3.5, 20), cmap='jet')
ax.plot(trajectory[:, 0], trajectory[:, 1], marker='o', color='black', label='Trajectory')
ax.plot(1, 1, 'r*', markersize=10, label='Minimum (1,1)')
```

```
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_title('SMD on Rosenbrock Function')
ax.legend()
plt.show()

from mpl_toolkits.mplot3d import Axes3D

def rosenbrock(x, y, a=1, b=100):
    return (a - x)**2 + b * (y - x**2)**2

def rosenbrock_grad(x, y, a=1, b=100):
    grad_x = -2 * (a - x) - 4 * b * x * (y - x**2)
    grad_y = 2 * b * (y - x**2)
    return np.array([grad_x, grad_y])

def smd_rosenbrock(init_pos, iterations, step_size_func, a=1, b=100):
    trajectory = [init_pos]
    current_pos = np.array(init_pos)

    for i in range(1, iterations):
        grad = rosenbrock_grad(current_pos[0], current_pos[1], a, b)
        step_size = step_size_func(i)
        current_pos = current_pos - step_size * grad
        trajectory.append(current_pos)

    return np.array(trajectory)

def step_size_func(n, gamma_0=0.001):
    """Define the decreasing step size function."""
    return gamma_0 / np.sqrt(n)

init_pos = (-1, -1)
iterations = 1000000

trajectory = smd_rosenbrock(init_pos, iterations, step_size_func)

x_vals = np.linspace(-2, 2, 400)
y_vals = np.linspace(-1, 3, 400)
X, Y = np.meshgrid(x_vals, y_vals)
Z = rosenbrock(X, Y)

fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')
```

```
surf = ax.plot_surface(X, Y, Z, cmap='jet', alpha=0.7, edgecolor='none')

ax.plot(trajectory[:, 0], trajectory[:, 1], rosenbrock(trajectory[:, 0], trajectory[:, 1]),

ax.view_init(elev=45, azim=134)

ax.set_xlabel('X axis')
ax.set_ylabel('Y axis')
ax.set_zlabel('Function Value')
ax.set_title('3D View of SMD on Rosenbrock Function')
ax.legend()

plt.show()

def rosenbrock(x):
    return sum(100.0 * (x[1:] - x[:-1]**2)**2 + (1 - x[:-1])**2)

def grad_rosenbrock(x):
    grad = np.zeros_like(x)
    grad[0] = -400 * x[0] * (x[1] - x[0]**2) - 2 * (1 - x[0])
    grad[-1] = 200 * (x[-1] - x[-2]**2)
    grad[1:-1] = -400 * x[1:-1] * (x[2:] - x[1:-1]**2) - 2 * (1 - x[1:-1]) + 200 * (x[1:-1]
    return grad

def stochastic_mirror_descent(x_init, lr=0.01, num_steps=1000, beta=0.9, noise_scale=0.01):
    x = x_init.copy()
    v = np.zeros_like(x)
    losses = []

    for step in range(num_steps):
        noise = np.random.normal(scale=noise_scale, size=x.shape)
        grad = grad_rosenbrock(x) + noise
        v = beta * v + (1 - beta) * grad
        x -= lr * v
        loss = rosenbrock(x)
        losses.append(loss)

    return x, losses

x_init = np.array([-1.5, -0.5, 0.5, 1.5])

x_final, losses = stochastic_mirror_descent(x_init)

plt.figure(figsize=(10, 6))
plt.plot(losses, label='Stochastic Mirror Descent', color='blue', linewidth=2)
```

```
plt.yscale('log')
plt.xlabel('Iteration', fontsize=14)
plt.ylabel('Loss (log scale)', fontsize=14)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.legend(fontsize=14)
plt.grid(True, linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()
```