Unlocking Black-Box Prompt Tuning Efficiency via Zeroth-Order Optimization

Heshen Zhan[12], **Congliang Chen**[1], **Tian Ding**[2], **Ziniu Li**[1], **Ruoyu Sun**[†12]
[1]The Chinese University of Hong Kong, Shenzhen, China
[2]Shenzhen International Center For Industrial And Applied Mathematics,
Shenzhen Research Institute of Big Data
{heshenzhan,congliangchen,ziniuli}@link.cuhk.edu.cn
dingtian@sribd.cn, sunruoyu@cuhk.edu.cn

Abstract

Prompt optimization emerges as an important technique for adapting Large Language Models (LLMs) to specific tasks. Unfortunately, LLM proprietors often limit access to models' internal weights, confining users to inference API services. This restriction poses a significant challenge for prompt optimization, as conventional optimization-based algorithms rely heavily on gradient information, which is unavailable via inference APIs. Addressing this challenge, this paper presents the Zeroth-Order Tuning (ZOT) approach, which enables efficient prompt tuning solely via inference APIs. ZOT adopts the zeroth-order optimization framework, utilizing finite differences to approximate gradient information. We further incorporate ZOT with gradient clipping and momentum techniques to enhance the tuning effectiveness. Experimental results show that ZOT outperforms existing black-box prompt tuning methods in terms of both task-specific performance and convergence speed. Furthermore, we provide a theoretical explanation for the unexpectedly strong performance of zeroth-order methods on LLM prompt tuning. By introducing the concept of effective dimension, we establish a strong connection between the inherently low effective dimension of prompt spaces and the superior convergence speed of zeroth-order methods. Our code is available at `https://github.com/ZhanHeshen/ZOT`.

1 Introduction

Prompts serve as essential tools for adapting Large Language Models (LLMs) to specific downstream tasks and aligning them with human objectives (Gao et al., 2020; Liu et al., 2023b; Schick and Schütze, 2020b; Li and Liang, 2021; Liu et al., 2023a). Research has demonstrated that carefully crafted prompts can significantly boost LLM performance across various applications, such as facilitating creative writing (Dang et al., 2023), streamlining question-answering processes (Ye et al., 2024), and improving fairness by minimizing bias in content generation (Ma et al., 2023). A notable example is the "Let's think step by step" prompt (Kojima et al., 2022), which enabled InstructionGPT (Ouyang et al., 2022) to achieve an impressive accuracy increase of over 30% on the GSM8K task (Cobbe et al., 2021). Despite these advances, creating effective prompts manually may involve extensive trial and error and demand specialized knowledge. These challenges has steered recent studies towards the development of prompt tuning algorithms, aiming at automating the discovery of effective prompts (Sun et al., 2022b; Diao et al., 2022).

Driven by commercial and security concerns, LLM owners often restrict access to the models' underlying weights, offering services solely through inference APIs (Brown et al., 2020). In such scenarios, LLMs are perceived as “black boxes” by users. This poses a crucial challenge to prompt tuning because gradient information, which is a cornerstone for standard optimization algorithms, is no longer accessible.

Black-Box Tuning (BBT) method (Sun et al., 2022b) is among the first to perform automatic prompt tuning by only accessing the LLM inference API. Specifically, BBT adopts a black-box optimization algorithm termed CMA-ES (Covariance Matrix Adaptation Evolution Strategy) (Hansen, 2016) to refine the continuous embedding of prompts, known as “soft prompts”. As a population-based algorithm, CMA-ES samples a considerable group of candidate solutions in each iteration. This process can lead to a significant increase in API calls, thereby incurring substantial time and financial costs. Furthermore, despite CMA-ES's empirical efficacy across diverse applications, its theoretical convergence properties have yet to be justified. These limitations underscore the potential for future research on more efficient and theoretically grounded black-box tuning methods.

Findings of the Association for Computational Linguistics: EACL 2024, pages 14825–14838
November 12-16, 2024

This paper harnesses the potential of zeroth-order optimization methods to enhance black-box prompt tuning. Zeroth-order methods stand as the gradient-free analogs to first-order methods like gradient descent and stochastic gradient descent. By approximating gradients through finite differences, these methods necessitate only function value evaluations, circumventing the need for direct gradient access. In this work, we introduce **Z**eroth-**O**rder **T**uning(**ZOT**). ZOT extends the zeroth-order stochastic gradient descent (ZO-SGD) framework by integrating gradient clipping (Bengio et al., 2017) and momentum techniques (Polyak, 1964; Nesterov, 1983) to improve the algorithm performance. To counteract potential diminishment in momentum at the initial stage, we apply a bias correction mechanism. In the experiments, ZOT achieves over 2x speed-up compared to BBT with comparable or even better performance across various prompt tuning tasks.

The success of zeroth-order methods in prompt tuning presents a notable surprise, offering a compelling deviation from established optimization theory. Although Nesterov's seminal work (Nesterov and Spokoiny, 2017) theoretically proved that zeroth-order gradient descent can converge, it also highlighted a significant caveat: their theoretical convergence speed is expected to be d times worse than that of standard gradient descent, where d is the dimensionality of the optimization problem. This slowdown effect, characterized by the number of samples evaluated, suggests a seemingly prohibitive inefficiency of zeroth-order methods in high-dimensional problems, including the LLM prompt tuning problem.

Upon revisiting and refining the proof in Nesterov and Spokoiny (2017), we uncover that the anticipated slowdown of zeroth-order methods can be significantly mitigated if the sum of the eigenvalue of the Hessian matrix remains small. This observation leads us to introduce the concept of effective dimension D_e, defined by

$$D_e = \frac{\sup_{x\in\mathbb{R}^d} \sum_i \lambda_i(\nabla^2 f(x))}{L}, \tag{1}$$

where f denotes the loss function, $\lambda_i(\nabla^2 f(\cdot))$ is the i-th largest eigenvalue of $\nabla^2 f(\cdot)$, L is the Lipschitz constant of ∇f. Our analysis reveals that the reduction in the convergence speed of zeroth-order gradient descent is more accurately characterized by D_e, rather than d. This finding recalibrates the expected efficiency of zeroth-order methods and highlights a scenario where zeroth-order methods can approach the speed of their first-order counterparts. Experiment results further validate our theoretical insight, showing that the effective dimension of LLM prompt spaces is notably smaller than the ambient dimension.

Our contributions are summarized as follows:

- **Zeroth-Order Approach**: We introduce ZOT, a zeroth-order algorithm tailored for the efficiently tuning LLM prompts. ZOT operates by only leveraging loss function values, thereby circumventing the need for direct gradient information. This method enables prompt tuning in constrained scenarios, such as the case where the access to LLMs is restricted to inference APIs.

- **Empirical Advances**: Our experimental results show that ZOT achieves at least a 2x training speed-up compared to existing black-box tuning methods. Across 9 datasets, we also report an average accuracy improvement of 3.23%. These results underscore the practical effectiveness of our method.

- **Theoretical Insights**: We provide a theoretical explanation for the success of zeroth-order methods in LLM prompt tuning by introducing the concept of effective dimension. We show that the anticipated inefficiencies of zeroth-order methods can be significantly mitigated if the optimization problem exhibits a low effective dimension. Our empirical investigation further demonstrates a strong correlation between the characteristic of low effective dimension and the enhanced performance achieved by zeroth-order methods

2 Problem Formulation

In this section, we provide the necessary background on prompt tuning and zeroth-order optimization, followed by the introduction of our proposed algorithm, ZOT.

2.1 Prompt Tuning

Prompting is a technique designed to augment input data with carefully crafted phrases or vectors, thereby enabling LLMs to more effectively tackle downstream tasks. For training data (X_{in}, Y), a language model processes the text input X_{in} alongside a prompt x, subsequently producing logits over the label Y. Using these logits, we compute the loss value. The whole process can be formalized as the optimization problem:

$$x^* = \arg\min_x f(x; (X_{in}, Y)) \tag{2}$$

where f denotes the composition of model inference and loss computation.

In hard prompt tuning, the prompt x is a discrete word or character. The input data is concatenated with x, and then the model performs inference with the new input. In soft prompt tuning, the input data is first mapped into an embedding vector. The prompt x, also represented as a vector, is either concatenated or added to the embedding of the input data. The model then performs inference with the new embedding vector.

2.2 Zeroth-Order Optimization Methods

Zeroth-order optimization methods involve gradient estimation using query feedback. A commonly used gradient estimator is given below.

Definition 2.1 (Randomized Gradient Estimation)**.** For a differentiable function f, its gradient can be estimated with two stochastic queries:

$$\hat{\nabla} f(x) = \frac{f(x + \rho z) - f(x - \rho z)}{2\rho} z, \tag{3}$$

where $z \sim \mathcal{N}(0, \mathbf{I}_d)$ is a random vector, $\rho > 0$ is a small number.

Remark 2.2. The gradient estimator, as defined in Eq. (3) relies on the random variable z, rendering it also randomized. Research by (Duchi et al., 2015) and (Nesterov and Spokoiny, 2017) have showed that $\mathbb{E}[\hat{\nabla} f(x)] = \nabla f(x) + o(\rho)$. Here the term $o(\rho)$ is the gradient estimation error, diminishing to 0 as ρ goes to 0. Consequently, the gradient can be approximated well with a sufficiently small ρ and enough estimations. This context also motivates the concept of the n*-points* estimator, which averages the outcomes from n-times stochastic estimation in Eq. (3).

With the gradient estimator in Eq. (3), one can easily apply the idea of gradient descent for optimization. We outline this idea in Algorithm 2.

2.3 ZOT: Zeroth-Order Tuning

The direct application of ZO-SGD in prompt tuning could fail due to the variance issue associated with stochastic gradients. To address this challenge, we propose integrating two techniques: gradient norm clipping (Zhang et al., 2019), and Polyak's momentum method (Polyak, 1964). Gradient norm clipping effectively mitigates the impact of stochastic noise, while Polyak's momentum strategy enhances gradient estimation by employing an exponential moving average. Our experiments in the next section will show that these two techniques are essential for achieving good performance. The resulting algorithm named ZOT (Zeroth-Order Tuning) is presented in Algorithm 3.

Algorithm 1 Randomized_Grad

Input: loss function f, prompt x, the number of points for gradient estimation n, perturbation scale ρ, data $\mathcal{D}$
for $i = 1$ **to** n
  sample $z_i \sim \mathcal{N}(0, \mathbf{I}_d)$
  $f_{i,1} = f(x + \rho z_i; \mathcal{D})$
  $f_{i,2} = f(x - \rho z_i; \mathcal{D})$
  $\tilde{\nabla} f_i = \frac{f_{i,1} - f_{i,2}}{2\rho} z_i$
end for
$\hat{\nabla} f = \frac{\sum_{i=1}^{n} \tilde{\nabla} f_i}{n}$
return $\hat{\nabla} f$

Algorithm 2 ZO-SGD: Zeroth-Order Stochastic Gradient Descent.

Input: loss function f, learning rate γ, total iterations T, and the number points for gradient estimation n, full data $\mathcal{D}$
Initialization set soft prompt $x_0 = 0$
for $t = 1$ **to** T **do**
  Sample mini batch of data $B \subset \mathcal{D}$
  $g_{t-1} =$ Randomized_Grad(f, x_{t-1}, n, ρ, B)
  $x_t = x_{t-1} - \gamma g_{t-1}$
end for
return x_T

3 Experiments

3.1 Setup

Dataset We run experiments across a variety of text processing tasks, including sentiment analysis (SST-2 (Socher et al., 2013) and Yelp polarity (Zhang et al., 2015)), content classification (AG's News (Zhang et al., 2015)), entity classification (DBPedia (Zhang et al., 2015)), paraphrase identification (MRPC (Dolan and Brockett, 2005)), natural language inference (SNLI (Bowman et al., 2015) and RTE (Wang et al., 2018)), question answering (COPA (Roemmele et al., 2011)) and linguistic acceptability (CoLA (Warstadt et al., 2018)). Among

Algorithm 3 ZOT: Zeroth-Order Tuning.

Input: loss function f, learning rate γ, total iterations T, and the number points for gradient estimation n, perturbation scale ρ, clipping threshold c, momentum factor β, full data $\mathcal{D}$

Initialization set soft prompt $x_0 = 0$, momentum $m_0 = 0$

for $t = 1$ **to** T **do**

Sample mini batch of data $B \subset \mathcal{D}$

$g_{t-1} = \text{Randomized\_Grad}(f, x_{t-1}, n, \rho, B)$

$g_{t-1} = \min(1, \frac{c}{||g_{t-1}||})g_{t-1}$

$m_t = \beta m_{t-1} + (1-\beta)g_{t-1}$

$\hat{m}_t = \frac{m_t}{1-\beta^t}$

$x_t = x_{t-1} - \gamma \hat{m}_t$

end for

return x_T

them, SST-2, Yelp Polarity, AG's News, DBPedia and CoLA are single sentence dataset, MRPC, SNLI, RTE and COPA are sentence pair dataset.

Backbone Model We implement our methodology on an encoder-only model and two decoder-only models. For the encoder-only model, we use RoBERTa$_{\text{LARGE}}$ (Liu et al., 2019), which has about 355 million parameters and demonstrates superior performance and robustness in a wide range of NLP tasks, such as text classification, sentiment analysis, and named entity recognition. For the decoder-only models, we use GPT2$_{\text{LARGE}}$ (Liu et al., 2019) and LLaMA-7B (Touvron et al., 2023). GPT2$_{\text{LARGE}}$ has about 774 million parameters, renowned for its deep understanding of language context and generation capabilities. LLaMA-7B boasting about 7 billion parameters, is designed to offer a compelling balance between model size and performance and represents the cutting edge in language model efficiency and scalability. For every model, we only access the word embedding, and add soft prompt to the word embedding, leaving the rest structure as a total black-box.

Few-shot Setting In reality, the accessible data may often be scarce. Leveraging the inherent few-shot learning capabilities of the pre-trained foundation model (Brown et al., 2020), we adopt a setting tailored for a few-shot scenario. We follow Zhang et al. (2020); Gao et al. (2020); Gu et al. (2021), selecting a constrained number of examples from each category within the datasets. Specifically, we randomly select k samples from each class to form the k-shot training set, denoted as $\mathcal{D}_{train}$. The parameter k represents the number of instances per class, effectively limiting the scope of our training data. We construct a development set $\mathcal{D}_{dev}$ by randomly selecting k samples for each class from the remaining dataset. As suggested by Perez et al. (2021), let $|\mathcal{D}_{train}| = |\mathcal{D}_{dev}|$. For evaluation purposes, the original development set is utilized as the test set. In cases where a development set is not available, the standard test set is used, ensuring a significantly larger test set ($|\mathcal{D}_{test}| \gg |\mathcal{D}_{train}| = |\mathcal{D}_{dev}|$). For systematically evaluating few-shot performance, we randomly sample 3 different splits of $\mathcal{D}_{train}$ and $\mathcal{D}_{dev}$ and measure the average performance of these 3 splits.

Hyper-parameter We adopt a batch size of 64 for training; however, if the training dataset comprises fewer than 64 samples, we utilize the entire dataset. Consistent with most datasets, the prompt is trained for 2000 steps to achieve optimal performance. Specifically for DBPedia, in line with recommendations from (Sun et al., 2024), we extend the training to 8000 steps to ensure convergence. The perturbation scale of zeroth-order gradient is searched within $\{0.1, 0.2\}$. We do the grid search of learning rate on $\{0.01, 0.03, 0.1, 0.3, 1, 3\}$, clipping rate on $\{8, 10, 15, 50, 100\}$, momentum decay rate on $\{0.8, 0.9, 0.93, 0.96\}$. We test the momentum on both settings: with bias correction and without bias correction. We use Cosine Annealing scheduler with minimal learning rate equal to a ratio times learning rate. We test the ratio by grid search on $\{0.1, 0.2, 0.3, 0.5\}$. Without loss of generality, we set the number of soft prompts to 50. The statistics of all the hyper-parameter can be summarized in Table 1.

Hyper-parameter	**Default**
# Training steps	2000
Batch size	64
Learning rate	{0.01, 0.03, 0.1, 0.3, 1, 3}
Clipping rate	{8, 10, 15, 50, 100}
Momentum decay rate	{0, 0.8, 0.9, 0.93, 0.96}
Scheduler eta_min	lr*{0.1, 0.2, 0.3, 0.5, 1}

Table 1: Default configuration of hyper-parameter.

3.2 Results

Comparison of Performance Table 2 presents the performance on the RoBERTa$_{\text{LARGE}}$ and GPT2$_{\text{LARGE}}$. Compared to manual prompting, ZOT achieves significant improvements across all 9 datasets, demonstrating its ability of adapting model to specific tasks. Compared to BBT, which tunes the soft prompt using CMA-ES, ZOT demonstrates better performance on 7 out of 9 datasets.

For LLaMA-7B, we choose 4 different kinds of tasks: sentiment analysis (SST-2 and Yelp P.), content classification (AG's News), paraphrase identification (MRPC) and natural language inference (SNLI and RTE). Table 3 displays the results on LLaMA-7B, which shows great improvements in the cutting edge model. ZOT can make more improvements than BBT. ZOT gains substantial improvement compared to manual prompts and BBT on RoBERTa$_{\text{LARGE}}$, GPT2$_{\text{LARGE}}$ and LLaMA-7B.

Comparison of Efficiency We evaluate the training efficiency of ZOT versus BBT on the RoBERTa$_{\text{LARGE}}$ model, examining their performance across datasets with varying numbers of classes. Specifically, SST-2 comprises two classes, AG's News consists of four classes, and DBPedia includes fourteen classes. For each dataset, we select k samples from each class and perform updates for an identical number of total steps. As illustrated in Table 4, ZOT does not only outperform BBT in terms of speed but also shows a greater advantage as the number of classes increases. For the SST-2 dataset, ZOT is at least twice as fast as BBT, while for DBPedia, the improvement is even more pronounced, with ZOT being at least four times faster than BBT.

BBT employs projection to map high-dimensional soft prompts into a lower-dimensional space, aiming to reduce the complexity of the optimization problem. This dimensionality reduction can aid algorithms like CMA-ES, which may struggle with the original high-dimensional space of soft prompts but potentially compromises the representational capacity of the soft prompts. In contrast, our method bypasses the need for projection by directly optimizing the soft prompts. This approach not only preserves the full representational power of the soft prompts but also ensures efficiency, thereby resulting in superior performance.

3.3 Ablation Study

In this section, we test the contribution of learning rate, clipping, momentum, bias, and scheduler. To explore the effect of each mechanism while avoiding the influence of hyper-parameters, we test the performance by running the grid search on the default configuration suggested by Table 1. The results are shown in Table 5. We can see that ZOT can achieve better results.

4 Discussion

In this section, we aim to explain why the zeroth-order method is effective in prompt tuning. In particular, we focus on how it can mitigate the curse of dimensionality.

4.1 Effective Dimension

The classical optimization theory suggests that the convergence rate of zeroth-order optimization methods depends on the dimension of the optimization variable (Duchi et al., 2015; Nesterov and Spokoiny, 2017). In particular, the convergence becomes slow for high-dimensional problems. This is the famous curse of dimensionality. To verify the problem of the curse of dimensionality, we introduce a new computable concept called effective dimension:

Definition 4.1 (Effective Dimension)**.** The effective dimension of f is

$$D_e(f) = \frac{\sup_{x \in e} \sum_i \lambda_i(\nabla^2 f(x))}{L}, \tag{4}$$

where e is the variable space, λ_i are the i-th largest eigenvalue of the Hessian matrix $\nabla^2 f(x)$, and L is the Lipschitz continuous parameter, i.e.,

$$||\nabla f(x) - \nabla f(y)|| \leq L||x-y||, \forall x, y \in \mathbb{R}^d. \tag{5}$$

We will later argue that the convergence rate of zero-order optimization actually depends on the effective dimension, which could be much smaller than the original problem dimension. As such, we can explain the observed superior results in the previous section.

The eigenvalues of Hessian matrix can measure the local changes of gradients. Therefore, if D_e is large, there exists at least one region, gradient will change in plenty of dimensions; if D_e is small, the gradient only change in a small number of dimension among the whole space. Hence, D_e characterizes the geometric properties of f. A small D_e means a good loss landscape, which can bring benefits to optimization. Motivated by (Malladi et al., 2024), we prove the convergence of zeroth-order gradient descent with effective dimension and show that our bound is tighter than that of (Malladi et al., 2024).

Method	SST-2 acc	Yelp P. acc	AG's News acc	DBPedia acc	MRPC F1	SNLI acc	RTE acc	COPA acc	CoLA acc	Avg.
Results on RoBERTa$_{LARGE}$										
Manual Prompt	79.82	89.65	76.96	41.33	67.40	31.11	51.62	44.00	30.87	56.97
BBT	$\mathbf{89.56_{0.25}}$	$91.50_{0.16}$	$81.51_{0.79}$	$87.80_{1.53}$	$61.56_{4.34}$	$\mathbf{46.58_{1.33}}$	$52.59_{2.21}$	$45.00_{2.65}$	$46.56_{7.07}$	66.96
ZOT	$88.59_{1.90}$	$\mathbf{91.76_{0.76}}$	$\mathbf{83.63_{0.01}}$	$\mathbf{90.25_{0.44}}$	$\mathbf{79.11_{0.03}}$	$43.70_{4.12}$	$\mathbf{53.19_{1.92}}$	$\mathbf{52.33_{3.06}}$	$\mathbf{48.96_{5.43}}$	**70.17**
Results on GPT2$_{LARGE}$										
Manual Prompt	51.03	68.44	72.00	22.75	5.41	33.70	46.21	45.00	30.87	41.71
BBT	$75.53_{1.98}$	$84.55_{3.48}$	$77.63_{1.89}$	$73.51_{9.56}$	$\mathbf{73.48_{3.87}}$	$\mathbf{43.25_{3.44}}$	$51.86_{1.11}$	$46.00_{1.73}$	$54.75_{6.07}$	64.51
ZOT	$\mathbf{78.78_{0.75}}$	$\mathbf{87.31_{2.55}}$	$\mathbf{78.24_{1.52}}$	$\mathbf{82.29_{2.51}}$	$71.32_{0.02}$	$41.56_{3.23}$	$\mathbf{54.27_{3.89}}$	$\mathbf{54.67_{4.16}}$	$\mathbf{65.45_{1.44}}$	**68.21**

Table 2: Comparison of results on RoBERTa$_{\text{LARGE}}$ and GPT2$_{\text{LARGE}}$. We report the mean and standard deviation of performance on 3 data splits. All experiments are on the 16-shot setting. We see that in both RoBERTa$_{\text{LARGE}}$ and GPT2$_{\text{LARGE}}$, ZOT makes improvements on average across seven datasets.

Method	SST-2 acc	Yelp P. acc	AG's News acc	MRPC f1	SNLI acc	RTE acc	Avg.
Results on LLaMA-7B							
Manual Prompt	52.18	55.11	27.71	15.58	32.23	48.38	38.53
BBT	$53.78_{2.76}$	$74.31_{3.32}$	$60.38_{18.59}$	$65.81_{12.10}$	$\mathbf{34.80_{0.83}}$	$49.46_{2.53}$	56.42
ZOT	$\mathbf{69.42_{4.81}}$	$\mathbf{85.14_{3.26}}$	$\mathbf{74.22_{2.15}}$	$\mathbf{69.17_{1.84}}$	$32.50_{0.50}$	$\mathbf{50.42_{4.31}}$	**63.48**

Table 3: Experiments on LLaMA-7B. We test the performance on 4 different kinds of tasks: sentiment analysis (SST-2 and Yelp P.), content classification (AG's News), paraphrase identification (MRPC) and natural language inference (SNLI and RTE).

	SST-2 (2 classes)	AG's News (4 classes)	DBPedia (14 classes)
BBT	19.83 mins	59.53 mins	602.49 mins
ZOT	8.96 mins	23.84 mins	142.51 mins

Table 4: Training time comparison for black-box method. We select datasets have different number of classes: 2 classes (SST-2), 4 classes (AG's News) and 14 classes (DBPedia).

	SST-2 (acc)	AG's News (acc)	RTE (acc)
ZO-SGD	$85.82_{0.26}$	$82.47_{1.51}$	$51.86_{0.42}$
ZO-SGD(Clipping)	$85.89_{0.12}$	$83.34_{0.01}$	$51.87_{0.00}$
ZO-SGD(Mom)	$85.93_{0.02}$	$83.52_{0.01}$	$52.35_{0.02}$
ZO-SGD(Mom+bias)	$87.00_{0.01}$	$83.63_{0.01}$	$51.87_{0.00}$
ZO-SGD(Scheduler)	$85.78_{0.22}$	$82.30_{0.01}$	$52.71_{0.01}$
ZOT	$88.59_{1.90}$	$83.63_{0.01}$	$53.19_{1.92}$

Table 5: Results of ablation study on clipping, momentum, bias and scheduler. Mom: momentum without bias correction; Mom+bias: momentum with bias correction; Scheduler: cosine annealing learning rate scheduler

4.2 Convergent Order Determined by Effective Dimension

We first introduce PL-Inequality (Polyak, 1963), which is usually considered in the optimization theory to prove the global convergence.

Definition 4.2 (PL-Inequality)**.** Let $f^* = \min_{\mathbf{x}} f(\mathbf{x})$, we say a function f satisfies PL-Inequality if the following holds for some $\mu > 0$:

$$\frac{1}{2}||\nabla f(x)||^2 \geq \mu(f(x) - f^*). \tag{6}$$

Now we show that under the PL-Inequality, the Zeroth-Order gradient descent can converge with an order proportional to D_e:

Theorem 4.3 (Convergence)**.** *Let f satisfies PL-Inequality with $\mu < L$, and $\{x_t\}_{t=1}^{\infty}$ is generated by ZO-SGD. Then for a learning rate $\gamma = \frac{n}{(D_e+n+1)L}$, $e = \mathbb{R}^d$, after t steps we will have:*

$$\mathbb{E}[f(\mathbf{x_t})] - f^* \leq (1 - \frac{n}{D_e+n+1}\frac{\mu}{L})^t (f(\mathbf{x_0}) - f^*), \tag{7}$$

which suggests that after iterations in the order of

$$O((1 + \frac{D_e+1}{n}) \underbrace{\frac{L}{\mu}\log(\frac{1}{\epsilon})}_{\textit{First Order GD}}) \tag{8}$$

ZO-GD can achieve ϵ-optimization accuracy.

The proof can be found in A.1. The effectiveness of the zeroth-order method in the prompt spaces of large language models is not straightforward. Studies such as (Duchi et al., 2015; Wibisono et al., 2012) have shown that, in the absence of additional structures, the convergence rate of the zeroth-order method linearly depends on the dimension d, implying that, in the worst-case scenario, it would require d times more iterations to achieve the same accuracy as the first-order method. Considering the large dimensions of prompt spaces in LLMs, often exceeding 10^5, the application of the zeroth-order method seems impractical. However, our results indicate that the zeroth-order method can converge with a dimension-independent order of

$O((1+\frac{D_e+1}{n})\frac{L}{\mu}log(\frac{1}{\epsilon}))$. This suggests that, in the worst case, the method needs only $1+\frac{D_e+1}{n}$ times more iterations than the first-order method, making it less dependent on the dimension. Our experiments have found that $1+\frac{D_e+1}{n}$ is significantly smaller than d, indicating that applying the zeroth-order method for prompt tuning may be more feasible than previously thought.

Proposition 4.4 (Local Effective Dimension)**.** *For a sequence $\{x_t\}_{t=1}^{\infty}$, we define the local effective dimension by setting*

$$e = \bigcup_{t=1}^{\infty}\{x \mid ||x - x_t|| \leq \gamma d||\nabla f(x_t)||\}. \quad (9)$$

(Malladi et al., 2024) presents the concept of "local r-effective rank", they showed that zeroth-order methods can be effective when this measure is small. We demonstrate that the "local effective dimension" provides a tighter metric than the "local r-effective rank". Proof can be found in Appendix A.4.

4.3 Verification of Convergence Order with Effective Dimension

We assess whether the convergence order delineated in Theorem 4.3 aligns with empirical observations. Initially, our evaluation focuses on quadratic functions. We generate several quadratic functions, ensuring that their D_e remains on the same scale, while progressively increasing the dimensionality of the variables. Our comparison encompasses first-order gradient descent and zeroth-order gradient descent methods, considering $n = 1, \frac{D_e}{2}, D_e$ and d dimensions. As shown by Figure 1, although the dimension of variable becomes larger and larger, the zeroth-order gradient with $n = \frac{D_e}{2}, D_e, d$ can still catch up the speed of first order method, which demonstrate that the convergent speed of zeroth-order descent can be dimension-free, the effective dimension can characterize it convergent speed.

Secondly, our verification extends to RoBERTa$_{\text{BASE}}$, a transformer-based model equipped with approximately 125 million parameters, featuring a soft prompt dimensionality of 38,400. To facilitate our analysis, we initially sample 500 points along the trajectory of the first-order gradient, subsequently estimating the effective dimension D_e, based on these points. In this context, our experimental findings indicate an estimated D_e of 131. We proceed to implement both first-order gradient descent and zeroth-order gradient descent methodologies, with the zeroth-order gradient descent executed under two distinct settings: $n = 1$ and $n = 200$. The outcomes reveal that, within the parameter space where $d = 38,400 \gg n = 200 > D_e$, the zeroth-order gradient descent method demonstrates a capability to nearly match the convergence speed of first-order gradient descent. This comparative performance is illustrated in Figure 2, showcasing the loss trajectories for each method.

These experiments demonstrate that the bounds established by Theorem 4.3 are tight enough to reflect the disparity in convergence speeds between zeroth-order and first-order gradient descent methods. Furthermore, the results reveal that the effective dimension within the prompt space of the language model is significantly smaller than d (specifically, $131 \ll 38,400$ in this instance), suggesting that the efficiency of zeroth-order gradient descent may have been underestimated previously, highlighting its potential for prompt tuning in language models with high-dimensional parameter spaces.

5 Related Work

Prompt tuning is an effective method for adapting Large Language Models (LLMs) to downstream tasks and aligning them with human intentions. Prompts can be manually designed (Brown et al., 2020; Schick et al., 2020; Schick and Schütze, 2020a), automatically generated (Gao et al., 2020), or tuned by gradients (Shin et al., 2020; Li and Liang, 2021; Wang et al., 2023; Jiang et al., 2023). A popular approach involves tuning a set of embeddings as soft prompts (Lester et al., 2021; Li and Liang, 2021; Vu et al., 2021; Gu et al., 2021; Liu et al., 2023b; Mokady et al., 2021; Qian et al., 2022; An et al., 2022), leveraging their advantageous optimization properties.

Recently, a new line of methods has emerged, which discovers soft prompts without accessing model weights. To address the challenges posed by high-dimensional space in black-box optimization, BBT (Sun et al., 2022b) projects the soft prompt into a lower-dimensional space and optimizes it using a black-box solver like CMA-ES. However, the convergence speed of BBT is relatively slow. An advanced version, BBTv2 (Sun et al., 2022a), optimizes multiple vectors in the middle layers of LLMs and uses a CMA-ES solver for each layer, thereby speeding up convergence. Nonetheless, BBTv2 requires interaction with the middle layers of the model. There are also some works (Diao

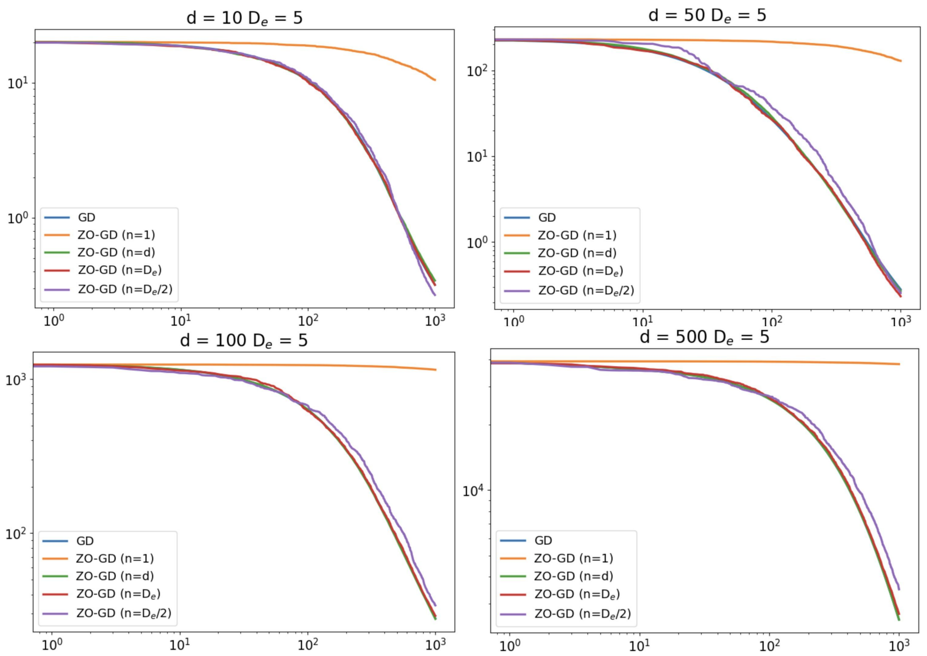

Figure 1: Experiment results on quadratic function. The horizontal axis and longitudinal axis stand for training steps and loss value respectively. Throughout these experiments, the effective dimension D_e is held constant at 5, while the overall dimension d of the problem space is varied, specifically, $d = 10, 50, 100, 500$. The results demonstrate that Zeroth-Order Gradient Descent (ZO-GD), with $n = \frac{D_e}{2}, D_e, d$, successfully matches the convergence speed of traditional Gradient Descent (GD). This parity in convergence speed is observed consistently, irrespective of the variations in d.

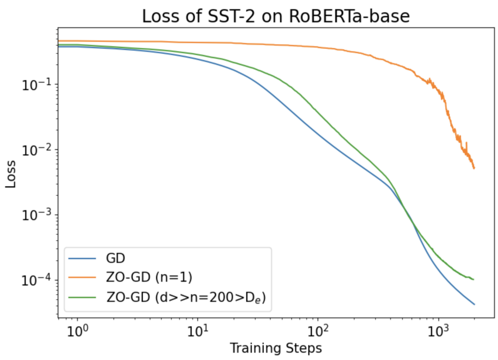

Figure 2: Experiments of SST-2 on RoBERTa-base.

et al., 2022; Deng et al., 2022; Cheng et al., 2023) discover hard prompts in black-box setting, but the discrete nature of hard prompts will cost more resources.

Black-box optimization methods have a long-standing history. Most of these methods, however, struggle with the challenges brought by high dimensionality. For instance, it has been shown in various studies that zeroth-order gradient methods have a dimensional-dependent convergence rate (Jamieson et al., 2012; Braun et al., 2017; Raginsky and Rakhlin, 2011; Duchi et al., 2015; Shamir, 2017; Nemirovskij and Yudin, 1983). Some works like (Wibisono et al., 2012; Duchi et al., 2015) suggest that without a more specific structure of the problem, the rate is at least proportional to the dimensionality d. (Malladi et al., 2024) applied zeroth-order methods to fine-tune LLMs with low memory requirements and demonstrated that the convergence rate of ZO-SGD depends on the "local r-effective rank". Our work introduces a metric called "effective dimension" and demonstrates that soft-prompt tuning tends to have a small effective dimension. This implies that zeroth-order optimization for soft-prompt tuning can also achieve rapid convergence.

6 Conclusion

In this study, we have introduced Zeroth-Order Tuning (ZOT), a black-box optimization method for efficiently tuning the LLM prompts. ZOT is designed to operate without direct gradient access, facilitating prompt tuning through model inference APIs alone. Our experimental analyses across diverse models and datasets have demonstrated that ZOT not only improves accuracy by an average of 3.23% but also doubles the convergence speed compared to existing black-box tuning methods.

Such performance is unexpected given the conventional theories which suggest that the efficiency of zeroth-order methods may be significantly hampered by the curse of dimensionality. Through a refinement of classical proofs, we have identified that the presence of a low effective dimension can substantially alleviate the anticipated slow convergence rates of zeroth-order methods. Further empirical investigation validates the connection between the low effective dimension of prompt spaces and the unexpectedly fast convergence of zeroth-order methods in prompt tuning.

Limitations

In comparison with manual prompt engineering: the training phase of black-box prompt tuning involves numerous API calls to large language model (LLM) service providers, incurring further expenses.

In comparison with white-box prompt tuning methods: In scenarios where full model tuning is feasible (the model weights are accessible for user modifications), white-box tuning methods may offer superior outcomes compared to ZOT.

Ethics Statement

There are no ethics-related issues in this paper. All data and other related resources used are open-source and commonly-used by many existing work.

Acknowledgements

This paper is supported by NSFC (No. 12326608); Hetao Shenzhen-Hong Kong Science and Technology Innovation Cooperation Zone Project (No.HZQSWS-KCCYB-2024016); University Development Fund UDF01001491, the Chinese University of Hong Kong, Shenzhen; Guangdong Provincial Key Laboratory of Mathematical Foundations for Artificial Intelligence (2023B1212010001).

References

Shengnan An, Yifei Li, Zeqi Lin, Qian Liu, Bei Chen, Qiang Fu, Weizhu Chen, Nanning Zheng, and Jian-Guang Lou. 2022. Input-tuning: Adapting unfamiliar inputs to frozen pretrained models. *arXiv preprint arXiv:2203.03131*.

Yoshua Bengio, Ian Goodfellow, and Aaron Courville. 2017. *Deep learning*, volume 1. MIT press Cambridge, MA, USA.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

Gábor Braun, Cristóbal Guzmán, and Sebastian Pokutta. 2017. Lower bounds on the oracle complexity of nonsmooth convex optimization via information theory. *IEEE Transactions on Information Theory*, 63(7):4709–4724.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Jiale Cheng, Xiao Liu, Kehan Zheng, Pei Ke, Hongning Wang, Yuxiao Dong, Jie Tang, and Minlie Huang. 2023. Black-box prompt optimization: Aligning large language models without model training. *arXiv preprint arXiv:2311.04155*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Hai Dang, Sven Goller, Florian Lehmann, and Daniel Buschek. 2023. Choice over control: How users write with large language models using diegetic and non-diegetic prompting. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pages 1–17.

Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric P Xing, and Zhiting Hu. 2022. Rlprompt: Optimizing discrete text prompts with reinforcement learning. *arXiv preprint arXiv:2205.12548*.

Shizhe Diao, Zhichao Huang, Ruijia Xu, Xuechun Li, Yong Lin, Xiao Zhou, and Tong Zhang. 2022. Black-box prompt learning for pre-trained language models. *arXiv preprint arXiv:2201.08531*.

Bill Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Third International Workshop on Paraphrasing (IWP2005)*.

John C Duchi, Michael I Jordan, Martin J Wainwright, and Andre Wibisono. 2015. Optimal rates for zero-order convex optimization: The power of two function evaluations. *IEEE Transactions on Information Theory*, 61(5):2788–2806.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2020. Making pre-trained language models better few-shot learners. *arXiv preprint arXiv:2012.15723*.

Yuxian Gu, Xu Han, Zhiyuan Liu, and Minlie Huang. 2021. Ppt: Pre-trained prompt tuning for few-shot learning. *arXiv preprint arXiv:2109.04332*.

Nikolaus Hansen. 2016. The cma evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*.

Kevin G Jamieson, Robert Nowak, and Ben Recht. 2012. Query complexity of derivative-free optimization. *Advances in Neural Information Processing Systems*, 25.

Song Jiang, Zahra Shakeri, Aaron Chan, Maziar Sanjabi, Hamed Firooz, Yinglong Xia, Bugra Akyildiz, Yizhou Sun, Jinchao Li, Qifan Wang, et al. 2023. Resprompt: Residual connection prompting advances multi-step reasoning in large language models. *arXiv preprint arXiv:2310.04743*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023a. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2023b. Gpt understands, too. *AI Open*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Huan Ma, Changqing Zhang, Yatao Bian, Lemao Liu, Zhirui Zhang, Peilin Zhao, Shu Zhang, Huazhu Fu, Qinghua Hu, and Bingzhe Wu. 2023. Fairness-guided few-shot prompting for large language models. *arXiv preprint arXiv:2303.13217*.

Sadhika Malladi, Tianyu Gao, Eshaan Nichani, Alex Damian, Jason D Lee, Danqi Chen, and Sanjeev Arora. 2024. Fine-tuning language models with just forward passes. *Advances in Neural Information Processing Systems*, 36.

Ron Mokady, Amir Hertz, and Amit H Bermano. 2021. Clipcap: Clip prefix for image captioning. *arXiv preprint arXiv:2111.09734*.

Arkadij Semenovič Nemirovskij and David Borisovich Yudin. 1983. Problem complexity and method efficiency in optimization.

Yurii Nesterov and Vladimir Spokoiny. 2017. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 17:527–566.

Yurii Evgen'evich Nesterov. 1983. A method of solving a convex programming problem with convergence rate o\bigl(k^2\bigr). In *Doklady Akademii Nauk*, volume 269, pages 543–547. Russian Academy of Sciences.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Ethan Perez, Douwe Kiela, and Kyunghyun Cho. 2021. True few-shot learning with language models. *Advances in neural information processing systems*, 34:11054–11070.

Boris T Polyak. 1963. Gradient methods for the minimisation of functionals. *USSR Computational Mathematics and Mathematical Physics*, 3(4):864–878.

Boris T Polyak. 1964. Some methods of speeding up the convergence of iteration methods. *Ussr computational mathematics and mathematical physics*, 4(5):1–17.

Jing Qian, Li Dong, Yelong Shen, Furu Wei, and Weizhu Chen. 2022. Controllable natural language generation with contrastive prefixes. *arXiv preprint arXiv:2202.13257*.

Maxim Raginsky and Alexander Rakhlin. 2011. Information-based complexity, feedback and dynamics in convex programming. *IEEE Transactions on Information Theory*, 57(10):7036–7056.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. 2011. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *2011 AAAI Spring Symposium Series*.

Timo Schick, Helmut Schmid, and Hinrich Schütze. 2020. Automatically identifying words that can serve as labels for few-shot text classification. *arXiv preprint arXiv:2010.13641*.

Timo Schick and Hinrich Schütze. 2020a. Exploiting cloze questions for few shot text classification and natural language inference. *arXiv preprint arXiv:2001.07676*.

Timo Schick and Hinrich Schütze. 2020b. It's not just size that matters: Small language models are also few-shot learners. *arXiv preprint arXiv:2009.07118*.

Ohad Shamir. 2017. An optimal algorithm for bandit and zero-order convex optimization with two-point feedback. *The Journal of Machine Learning Research*, 18(1):1703–1713.

Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. 2020. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *arXiv preprint arXiv:2010.15980*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Tianxiang Sun, Zhengfu He, Hong Qian, Yunhua Zhou, Xuan-Jing Huang, and Xipeng Qiu. 2022a. Bbtv2: towards a gradient-free future with large language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3916–3930.

Tianxiang Sun, Yunfan Shao, Hong Qian, Xuanjing Huang, and Xipeng Qiu. 2022b. Black-box tuning for language-model-as-a-service. In *International Conference on Machine Learning*, pages 20841–20855. PMLR.

Tianxiang Sun et al. 2024. Black-box tuning github repository. https://github.com/txsun1997/Black-Box-Tuning/tree/main.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Tu Vu, Brian Lester, Noah Constant, Rami Al-Rfou, and Daniel Cer. 2021. Spot: Better frozen model adaptation through soft prompt transfer. *arXiv preprint arXiv:2110.07904*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Qifan Wang, Yuning Mao, Jingang Wang, Hanchao Yu, Shaoliang Nie, Sinong Wang, Fuli Feng, Lifu Huang, Xiaojun Quan, Zenglin Xu, et al. 2023. Aprompt: Attention prompt tuning for efficient adaptation of pre-trained language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9147–9160.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2018. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*.

Andre Wibisono, Martin J Wainwright, Michael Jordan, and John C Duchi. 2012. Finite sample convergence rates of zero-order stochastic optimization methods. *Advances in Neural Information Processing Systems*, 25.

Junyi Ye, Mengnan Du, and Guiling Wang. 2024. Dataframe qa: A universal llm framework on dataframe question answering without data exposure. *arXiv preprint arXiv:2401.15463*.

Jingzhao Zhang, Tianxing He, Suvrit Sra, and Ali Jadbabaie. 2019. Why gradient clipping accelerates training: A theoretical justification for adaptivity. *arXiv preprint arXiv:1905.11881*.

Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q Weinberger, and Yoav Artzi. 2020. Revisiting few-sample bert fine-tuning. *arXiv preprint arXiv:2006.05987*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28.

A Appendix

A.1 Proof of Theorem

Lemma A.1. *Let $\hat{\nabla} f(\mathbf{x})$ be the estimated zeroth-order gradient, we have:*

$$\mathbb{E}[\hat{\nabla} f(\mathbf{x})\hat{\nabla} f(\mathbf{x})^\top] = (1+\frac{1}{n})\nabla f(\mathbf{x})\nabla f(\mathbf{x})^\top + \frac{1}{n}||\nabla f(\mathbf{x})||^2\mathbf{I} \tag{10}$$

Proof:

$$\hat{\nabla} f(\mathbf{x})\hat{\nabla} f(\mathbf{x})^\top = \frac{\sum z_i z_i^\top}{n}\nabla f(\mathbf{x})\hat{\nabla} f(\mathbf{x})^\top \frac{\sum z_j z_j^\top}{n}, \tag{11}$$

where $z_i, z_j \sim \mathcal{N}(0, \mathbf{I})$ are all i.i.d.
If $i \neq j$,

$$\mathbb{E}[z_i z_i^\top \nabla f(\mathbf{x})\nabla f(\mathbf{x})^\top z_j z_j^\top] = \nabla f(\mathbf{x})\nabla f(\mathbf{x})^\top. \tag{12}$$

If $i = j$,

$$\begin{aligned}&\mathbb{E}[z_i z_i^\top \nabla f(\mathbf{x})\nabla f(\mathbf{x})^\top z_i z_i^\top]\\ =&\mathbb{E}[z^{\otimes 4}](\nabla f(\mathbf{x}), \nabla f(\mathbf{x}))\\ =&||\nabla f(\mathbf{x})||^2 I + 2\nabla f(\mathbf{x})\nabla f(\mathbf{x})^\top\end{aligned} \tag{13}$$

Therefore, we have:

$$\begin{aligned}&\mathbb{E}[\hat{\nabla} f(\mathbf{x})\hat{\nabla} f(\mathbf{x})^\top]\\ =&\mathbb{E}[\frac{\sum z_i z_i^\top}{n}\nabla f(\mathbf{x})\hat{\nabla} f(\mathbf{x})^\top \frac{\sum z_j z_j^\top}{n}]\\ =&\frac{1}{n^2}\mathbb{E}[(\sum z_i z_i^\top)\nabla f(\mathbf{x})\hat{\nabla} f(\mathbf{x})^\top(\sum z_j z_j^\top)]\\ =&\frac{1}{n^2}[n(||\nabla f(x)||\mathbf{I} + 2\nabla f(x)\nabla f(x)^\top)\\ &+ (n^2-n)\nabla f(x)\nabla f(x)^\top]\\ =&(1+\frac{1}{n})\nabla f(\mathbf{x})\nabla f(\mathbf{x})^\top + \frac{1}{n}||\nabla f(\mathbf{x})||^2\mathbf{I}\end{aligned} \tag{14}$$

Lemma A.2 (Descent Lemma)**.** *Let f be L-Lipschitz, $\{\mathbf{x_t}\}_{t=0}^{\infty}$ is generated by unbiased randomized gradient with learning rate γ, we have:*

$$\begin{aligned}&\mathbb{E}[f(\mathbf{x_{t+1}})|\mathbf{x_t}] - f(\mathbf{x_t})\\ &\leq (-\gamma + \frac{1}{2}\gamma^2(\frac{\sum_i \lambda_i(\nabla^2 f(\mathbf{x_{t_0}}))}{nL}\\ &+ (1+\frac{1}{n}))L)||\nabla f(\mathbf{x_t})||^2,\end{aligned} \tag{15}$$

Proof: By Taylor expansion, for a $k \in [0, 1]$, we have:

$$\begin{aligned}&f(\mathbf{x_{t+1}}) = f(\mathbf{x_t}) + \nabla f(\mathbf{x_t})^\top(\mathbf{x_{t+1}} - \mathbf{x_t})\\ &+ \int_0^1 k(\mathbf{x_{t+1}} - \mathbf{x_t})^\top \nabla^2 f(k\mathbf{x_{t+1}}\\ &+ (1-k)\mathbf{x_t})(\mathbf{x_{t+1}} - \mathbf{x_t})dk\end{aligned} \tag{16}$$

We know $\mathbf{x_t}$ is update by

$$\mathbf{x_{t+1}} = \mathbf{x_t} - \gamma\hat{\nabla} f(\mathbf{x_t}). \tag{17}$$

Written $\nabla^2 f(k\mathbf{x_{t+1}} + (1-k)\mathbf{x_t})$ as

$$\nabla^2 f(k\mathbf{x_{t+1}} + (1-k)\mathbf{x_t}) = \sum_i \lambda_i v_i v_i^\top, \tag{18}$$

where λ_i is the eigenvalue of $\nabla^2 f(k\mathbf{x_{t+1}} + (1-k)\mathbf{x_t})$, v_i is its corresponding eigenvalue. For any integer $\mathbf{t}$, we have:

$$\begin{aligned}&\mathbb{E}\int_0^1 k(\mathbf{x_{t+1}} - \mathbf{x_t})^\top\\ &\nabla^2 f(k\mathbf{x_{t+1}} + (1-k)\mathbf{x_t})(\mathbf{x_{t+1}} - \mathbf{x_t})dk\\ =&\mathbb{E}\int_0^1 k(-\gamma\hat{\nabla} f(\mathbf{x_t}))^\top\\ &(\sum_i \lambda_i v_i v_i^\top)(-\gamma\hat{\nabla} f(\mathbf{x_t}))dk\\ =&\mathbb{E}\int_0^1 k\gamma^2\hat{\nabla} f(\mathbf{x_t})^\top(\sum_i \lambda_i v_i v_i^\top)\hat{\nabla} f(\mathbf{x_t})dk\\ =&\mathbb{E}\int_0^1 k\gamma^2\sum_i \lambda_i\hat{\nabla} f(\mathbf{x_t})^\top v_i v_i^\top\hat{\nabla} f(\mathbf{x_t})dk\\ =&\mathbb{E}\int_0^1 k\gamma^2\sum_i \lambda_i v_i^\top\hat{\nabla} f(\mathbf{x_t})\hat{\nabla} f(\mathbf{x_t})^\top v_i dk\\ =&\int_0^1 k\gamma^2\sum_i \lambda_i v_i^\top\mathbb{E}[\hat{\nabla} f(\mathbf{x_t})\hat{\nabla} f(\mathbf{x_t})^\top]v_i dk\\ =&\int_0^1 k\gamma^2\sum_i \lambda_i v_i^\top[(1+\frac{1}{n})\nabla f(\mathbf{x_t})\nabla f(\mathbf{x_t})^\top\end{aligned} \tag{19}$$

$$
\begin{aligned}
&+\frac{1}{n}||\nabla f(\mathbf{x_t})||^2\mathbf{I}]v_i dk\\
=&\int_0^1 k\gamma^2[(1+\frac{1}{n})\nabla f(\mathbf{x_t})^\top\\
&(\sum_i \lambda_i v_i v_i^\top)\nabla f(\mathbf{x_t})\\
&+\frac{1}{n}||\nabla f(\mathbf{x_t})||^2(\sum_i \lambda_i)]dk\\
=&\int_0^1 k\gamma^2[(1+\frac{1}{n})\nabla f(\mathbf{x_t})^\top\\
&(\nabla^2 f(k\mathbf{x_{t+1}}+(1-k)\mathbf{x_t}))\nabla f(\mathbf{x_t})\\
&+\frac{1}{n}||\nabla f(\mathbf{x_t})||^2(\sum_i \lambda_i)]dk\\
\le&\int_0^1 k\gamma^2[(1+\frac{1}{n})L||\nabla f(\mathbf{x_t})||^2\\
&+\frac{1}{n}||\nabla f(\mathbf{x_t})||^2(\sum_i \lambda_i)]dk\\
=&\frac{1}{2}\gamma^2[(1+\frac{1}{n})L+\frac{1}{n}(\sum_i \lambda_i)]||\nabla f(\mathbf{x_t})||^2.
\end{aligned} \tag{20}
$$

Plugging in 16 and take expectation, we have:

$$
\begin{aligned}
&\mathbb{E}[f(\mathbf{x_{t+1}})|\mathbf{x_t})]\\
=&f(\mathbf{x_t})+\mathbb{E}[\nabla f(\mathbf{x_t})^\top(\mathbf{x_{t+1}}-\mathbf{x_t})]\\
&+\mathbb{E}[\int_0^1 k(\mathbf{x_{t+1}}-\mathbf{x_t})^\top\\
&\nabla^2 f(k\mathbf{x_{t+1}}+(1-k)\mathbf{x_t})(\mathbf{x_{t+1}}-\mathbf{x_t})dk]
\end{aligned} \tag{21}
$$

$$
\begin{aligned}
\le& f(\mathbf{x_t})-\gamma\mathbb{E}[\nabla f(\mathbf{x_t})^\top zz^\top\nabla f(\mathbf{x_t})]\\
&+\frac{1}{2}\gamma^2[(1+\frac{1}{n})L+\frac{1}{n}(\sum_i \lambda_i)]||\nabla f(\mathbf{x_t})||^2
\end{aligned} \tag{22}
$$

$$
\begin{aligned}
=&f(\mathbf{x_t})-\gamma||\nabla f(\mathbf{x_t})||^2\\
&+\frac{1}{2}\gamma^2[(1+\frac{1}{n})L+\frac{1}{n}(\sum_i \lambda_i)]||\nabla f(\mathbf{x_t})||^2\\
=&f(\mathbf{x_t})\\
&+(-\gamma+\frac{1}{2}\gamma^2[(1+\frac{1}{n})L\\
&+\frac{1}{n}(\sum_i \lambda_i)])||\nabla f(\mathbf{x_t})||^2,
\end{aligned} \tag{23}
$$

i.e., we have

$$
\begin{aligned}
&\mathbb{E}[f(\mathbf{x_{t+1}})|\mathbf{x_t})]-f(\mathbf{x_t})\\
&\le(-\gamma+\frac{1}{2}\gamma^2[\frac{(\sum_i \lambda_i)}{nL}+(1+\frac{1}{n})]L)||\nabla f(\mathbf{x_t})||^2
\end{aligned} \tag{24}
$$

Theorem A.3 (Convergence)**.** *Let f satisfies PL-Inequality with $\mu < L$, $\{x_i\}_{i=1}^\infty$ is generated by ZO-GD. Then for a learning rate $\gamma = \frac{n}{(D_e+n+1)L}$, after t steps we will have:*

$$
\begin{aligned}
&\mathbb{E}[f(\mathbf{x_t})]-f^*\\
&\le(1-\frac{n}{D_e+n+1}\frac{\mu}{L})^t(f(\mathbf{x_0})-f^*),
\end{aligned} \tag{25}
$$

which suggest that after

$$
O((1+\frac{D_e+1}{n})\underbrace{\frac{L}{\mu}log(\frac{1}{\epsilon})}_{\text{First Order GD}}) \tag{26}
$$

iterations ZO-GD can achieve ϵ-optimization accuracy.

Proof: By lemma A.2, we have

$$
\begin{aligned}
&\mathbb{E}[f(\mathbf{x_{t+1}})|\mathbf{x_t})]-f(\mathbf{x_t})\\
&\le[-\gamma+\frac{1}{2}\gamma^2(1+\frac{D_e+1}{n})L]||\nabla f(\mathbf{x_t})||^2.
\end{aligned} \tag{27}
$$

Assume f satisfy the PL-Inequality, when $\gamma < \frac{n}{(D_e+n+1)L}$, we have

$$
\begin{aligned}
&\mathbb{E}[f(\mathbf{x_{t+1}})|\mathbf{x_t})]\\
&\le f(\mathbf{x_t})\\
&+2\mu[-\gamma+\frac{1}{2}\gamma^2(1+\frac{D_e+1}{n})L](f(\mathbf{x_t})-f^*),
\end{aligned} \tag{28}
$$

i.e.,

$$
\begin{aligned}
&\mathbb{E}[f(\mathbf{x_{t+1}})|\mathbf{x_t})]-f(\mathbf{x_0})\\
\le&(f(\mathbf{x_t})-f(\mathbf{x_0}))\\
&+2\mu[-\gamma+\frac{1}{2}\gamma^2(1+\frac{D_e+1}{n})L]\\
&(f(\mathbf{x_t})-f(\mathbf{x_0}))\\
=&(1+2\mu[-\gamma+\frac{1}{2}\gamma^2(1+\frac{D_e+1}{n})L])\\
&(f(\mathbf{x_t})-f^*).
\end{aligned} \tag{29}
$$

Therefore, in expectation we have

$$
\begin{aligned}
&\mathbb{E}[f(\mathbf{x_{t+1}})]-f^*\\
\le&(1+2\mu[-\gamma+\frac{1}{2}\gamma^2(1+\frac{D_e+1}{n})L])^t\\
&(f(\mathbf{x_0})-f^*).
\end{aligned} \tag{30}
$$

Let $\gamma = \frac{n}{(D_e+n+1)L}$, we have:

$$
\begin{aligned}
&\mathbb{E}[f(\mathbf{x_t})]-f^*\\
&\le(1-\frac{n}{D_e+n+1}\frac{\mu}{L})^t(f(\mathbf{x_0})-f^*).
\end{aligned} \tag{31}
$$

which suggest that after

$$O((1+\frac{D_e+1}{n})\underbrace{\frac{L}{\mu}log(\frac{1}{\epsilon})}_{\text{First Order GD}}) \tag{32}$$

iterations ZO-GD can achieve ϵ-optimization accuracy.

Lemma A.4. *Let* $e = \bigcup_{t=1}^{\infty}\{x|||x-x_t|| \leq \gamma d||\nabla f(x_t)||\}$*, assume for every* t*, there exist* $H(x_t)$ *s.t.* $\nabla^2 f(x_t) \preceq H(x_t) \preceq LI_d$ *on* $\{x|||x-x_t|| \leq \gamma d||\nabla f(x_t)||\}$*, then the local effective dimension* D_e *is smaller than local* r*-effective rank on* $H(x_t)$*.*

Proof: By definition, we have $||H(x_t)||_{op} \leq L$ and $\sum_i \lambda_i(\nabla^2 f(x_t)) \leq tr(H(x_t))$, therefore

$$\begin{aligned} D_e(f) &= \frac{\sup_{x\in e}\sum_i \lambda_i(\nabla^2 f(x))}{L} \\ &\leq \frac{tr(H(x_t))}{||H(x_t)||_{op}} \\ &\leq r. \end{aligned} \tag{33}$$

End of the proof.