Parameter-Efficient Fine-Tuning via Partially Decomposable Loss Analysis and Sharing

Anonymous authors
Paper under double-blind review

Abstract

Large language model (LLM) has become a crucial tool for many machine learning research and applications. Due to the large parameter count of these models and the enormous amount of training data, large language models are usually strong at general tasks. For most applications, however, one would like a smaller, more parameter-efficient model that is specialized in a particular field. This motivates the design of fine-tuning, which tunes a pre-trained LLM for a few iterations on a dedicated dataset for specific tasks. If not handled correctly, the fine-tuning process would create another LLM that has a comparable amount of parameters, significantly slowing down any downstream applications.

One of the most widely-known ideas for resolving this issue is the Low-Rank Adaptation (LoRA) framework, where one assumes the fine-tuning weights are low-rank therefore the number of parameters together with the inference time is drastically improved. While performing well in practice, LoRA method is still a heuristic and lacks theoretical guarantees even though the loss function might inherit certain structures. Moreover, when fine-tuning multiple similar tasks in parallel, LoRA requires one to learn a pair of distinct low-rank matrices for each task, ignoring possible shared structure between tasks.

In this work, we design a framework that further reduces parameter count compared to LoRA and enables parameter sharing across different parallel fine-tuning tasks. When the number of parallel fine-tuning tasks grows larger, we cut the parameter count almost in half compared to LoRA. Moreover, we prove why our approach and more generally, LoRA works for a large class of loss functions. We empirically verify the effectiveness of our method on various benchmark models and datasets, demonstrating much-improved parameter count while retaining similar performance as LoRA.

1 Introduction

Large language models (LLMs) such as GPT-4 (OpenAI, 2023) are becoming increasingly important in the realm of artificial intelligence and information technology, serving a multitude of functions across various sectors (Chiang et al., 2023; Touvron et al., 2023a;b). Their ability to understand, generate, and interact with human language in a nuanced manner makes them invaluable tools in everything from customer service and data analysis to content creation and decision support systems. Beyond automating tasks, LLMs contribute to the development of conversational agents that can assist with mental health (Nori et al., 2023), offer educational tutoring (Nori et al., 2023; Phung et al., 2023; Kasneci et al., 2023), and provide specialized advice in legal or medical fields (Bommarito II & Katz, 2022; Wang et al., 2023). These models can process and analyze vast amounts of data far more quickly than humans, making them particularly useful in sifting through large datasets to identify trends or insights. Thus, LLMs are not only reshaping our interaction with technology but also have the potential to significantly impact how we solve complex problems, improve efficiency, and enhance the quality of life.

Fine-tuning is an essential step in harnessing the full potential of Large language models (Touvron et al., 2023b; Wang et al., 2023), tailoring their generalized capabilities to meet specific needs or goals. While these models are trained on a broad range of data to perform various tasks, they often require further customization to excel in specialized applications. Fine-tuning allows businesses, researchers, and developers to adapt LLMs for particular industries, such as healthcare (Bommarito II & Katz, 2022; Wang et al., 2023), finance (Yang et al., 2023; Wu et al., 2023), or law (Bommarito II & Katz, 2022), thereby optimizing their performance and making them more effective and reliable tools. This customization not only improves the model's utility but also helps in mitigating biases, ensuring ethical use, and meeting compliance standards. In essence, fine-tuning is the bridge between a model's generalized abilities and its application in solving real-world, domain-specific problems, making it a critical element in the deployment of LLMs across diverse settings. Moreover, fine-tuning is crucial for commercial deployment of LLMs (OpenAI, 2023; Touvron et al., 2023a;b), as they offer a simple, lightweighted and efficient approach to perform fast inference for dedicated tasks. For example, fine-tuning LLMs has been consequential for Enterprise co-pilots (E.g., github co-pilot) that are being deployed widely.

The process of fine-tuning a Large language model involves several methods, each with its unique advantages, depending on the application and goals. One common approach is data augmentation (Shorten et al., 2021; Feng et al., 2021; Yoo et al., 2021), where the existing dataset is expanded by adding variations of the data to increase diversity and reduce overfitting. Another method is curriculum learning (Xu et al., 2020; Bengio et al., 2009), which involves progressively training the model on increasingly complex tasks, allowing it to build up its expertise gradually. Transfer learning (Chronopoulou et al., 2019; Houlsby et al., 2019) is also widely used, taking a pre-trained model and adapting it for a specific task by training it further on a specialized dataset. Feature-based fine-tuning involves extracting certain layers or "features" from the pre-trained model and incorporating them into a new model designed for the specific task. Hyperparameter tuning, where settings like learning rate or batch size are adjusted, is also crucial for optimizing performance. Additionally, multi-task learning (Sanh et al., 2021; Liu et al., 2019a) can be employed to fine-tune the model on several related tasks simultaneously, thereby enhancing its generalizability. These methods can be used individually or in combination to ensure that the model performs optimally in its designated role, making fine-tuning a versatile and indispensable step in the deployment of large language models.

One of the key aspects to address is the parameter count of fine-tuning (Houlsby et al., 2019; Mangrulkar et al., 2022; Ding et al., 2023). Let $W_0 \in \mathbb{R}^{d\times m}$ denote the pre-training model weight, note that if done naively, even fine-tuning on a single data point will end up with a model as large as W_0, as the $\Delta W \in \mathbb{R}^{d\times m}$ is without any structure if no further assumptions are imposed. On the other hand, fine-tuning should be parameter-efficient and highly structured, as most of the technical heavy-lifting has been handled by the time- and parameter-consuming pre-trained model W_0. Drawing inspiration from deep learning theory, where the gradients of model weights are usually low-rank due to over-parametrization, Hu et al. (2021) proposes the Low-Rank Adaptation (LoRA) framework, where they assume the fine-tuning weights, ΔW admits a rank-r factorization for a hyperparameter r. While d and m can be as large as 10^4 to 10^5 and lead to a pre-trained model with trillions of parameters, the LoRA approach allows one to pick $r = 50$ or 100, reduces the parameter count by more than $100\times$ fold. Moreover, by performing the fine-tuning process on a low-rank model $\Delta W = AB$ for $A \in \mathbb{R}^{d\times r}$ and $B \in \mathbb{R}^{r\times m}$, they also effectively improve the inference time, as multiplying a vector with ΔW can be performed by first multiplying with B then computing the matrix-vector product using A and the resulting vector, a runtime improvement from $O(md)$ to $O(mr + dr)$. Due to these advantages, LoRA has become an important building block for the fine-tuning procedure of many LLMs, including GPT-3 (Brown et al., 2020).

Despite its impressive empirical performance, LoRA does have several drawbacks. The method itself is still a heuristic, as the work Hu et al. (2021) does not provide convergence guarantees and they instead motivate the effectiveness of LoRA from the perspective of subspace similarity Hamm & Lee (2008). From an algorithmic perspective, LoRA requires each individual fine-tuning task to learn a distinct pair of low-rank matrices A_i, B_i. This ignores the potential relevance between different tasks, for example, in the AEP Copilot fine-tuning pipeline, the content and citation retrieval are highly correlated, and are fine-tuned on the same LLM. One would expect the similarity and relevance of these two tasks can be exploited in a manner that further reduces the number of parameters required for fine-tuning. With these considerations in mind, we ask the following question:

Can we develop a framework that further reduces the parameter count for parallel fine-tuning, and provide theoretical guarantees for it?

In this work, we provide a positive answer to the above question. In particular, we propose a framework called *Multiple Parallel Low-Rank Adaptation* (`Multi-LoRA`) that enables one to fine-tune k tasks with the same pre-trained model in a more parameter-efficient manner than LoRA. Moreover, we prove that for a large class of loss functions, our method exhibits a good convergence speed under mild assumptions on the gradient. Our analysis, surprisingly, draws inspiration from the convergence analysis of FedAvg (McMahan et al., 2017) and its variant (Song et al., 2023). We also empirically verify the effectiveness of our proposed method on publicly available Roberta (Liu et al., 2019b) and GPT-2 (Radford et al., 2019) models. We achieve similar performance as LoRA while significantly reduce the parameter count. We summarize our contributions as follows:

- We propose `Multi-LoRA`, a framework that enables fine-tuning multiple tasks in parallel and reduces parameter count. Specifically, for k tasks, LoRA would require $O(kdr + kmr)$ parameters, while Multi-LoRA only requires $O(dr + kmr)$ parameters. If d and m are in the same order and k is large, this almost cuts the number of parameters in half.
- We provide theoretical guarantees for our framework via a partially decomposable loss analysis. In particular, we show that whenever the individual task loss has a good local structure (such as Lipschitzness, smoothness, and convexity), it can be propagated to the global loss.
- We empirically verify our algorithm by performing experiments on Roberta and GPT2 on various benchmarks as in Hu et al. (2021). We achieve comparable performances as Hu et al. (2021), while only using $\leq 60\%$ of parameters compared to LoRA.

1.1 Our Results

We start with the formulation of our loss,

Definition 1.1 (A mathematical interpretation of the model). *Let $L : \mathbb{R}^d \times \mathbb{R}^m \to \mathbb{R}$ denote a loss function. Let $k \geq 1$ denote a positive integer. Define the global loss function $\mathcal{L} : \mathbb{R}^d \times \mathbb{R}^{mk} \to \mathbb{R}$ as follows $\mathcal{L}(x, y) := \sum_{i=1}^k L(x, y_i)$ where $y \in \mathbb{R}^{mk}$ is a vector that concatenates all k parameters for each y_i: $y := \begin{bmatrix} y_1^\top & y_2^\top & \cdots & y_k^\top \end{bmatrix}^\top$.*

Lemma 1.2 (Informal of Lemma 4.4, lipschitz). *If L is γ-Lipschitz, then $\mathcal{L}$ is (γk)-Lipschitz.*

Lemma 1.3 (Informal of Lemma 4.6, smoothness). *If L is β-smooth, then $\mathcal{L}$ is $(k\beta)$-smooth.*

Our results for Lipschitzness and smoothness don't require additional assumptions. However, for strong convexity, extra structural assumptions are needed (for details see Lemma 4.8) as otherwise counterexample exists.

Lemma 1.4 (Informal of Lemma 4.8, strongly convex). *If L is α-strongly convex and under certain assumptions, then $\mathcal{L}$ is (0.5α)-strongly convex.*

Consequentially, standard first-order optimization methods can be applied directly to `Multi-LoRA` and convergence can be obtained.

2 Related Work

Practical/Empirical LLMs In the landscape of practical and empirical studies on Large Language Models (LLMs), a plethora of research has emerged that scrutinizes various facets of these computational behemoths. Studies often focus on the utility of LLMs in specific sectors such as healthcare, where they have been deployed for diagnostic assistance and drug discovery, or in the financial sector for risk assessment and fraud detection. Additionally, benchmarking papers have explored the raw performance of these models, evaluating them across various natural language processing tasks like machine translation, text summarization, and question-answering. BERT (Devlin et al., 2018) presents a novel approach in NLP by pre-training deep bidirectional representations on unlabeled text, which enables fine-tuning with minimal architectural modification to achieve state-of-the-art results across various language processing tasks. Across the GPT (Radford et al., 2018; 2019; Brown et al., 2020; OpenAI, 2023) series, the progression of GPT models is evident, each showcasing advancements in natural language understanding and generative capabilities. Starting with a focus on generative pre-training to improve task-specific fine-tuning in diverse NLP tasks, the series evolves towards the development of GPT-4 (OpenAI, 2023), a multimodal, large-scale model capable of human-level performance in professional and academic benchmarks, accepting both image and text inputs, and exhibiting advancements in zero-shot and few-shot learning, scalability, and applicability across a myriad of NLP tasks and real-world scenarios., T5 (Raffel et al., 2020) innovates in transfer learning for NLP by introducing a unified, text-to-text framework to harmonize diverse language problems and by comparing various elements like pre-training objectives and architectures. The PaLM series (Chowdhery et al., 2022; Anil et al., 2023) develops large-scale Transformer language models, with PaLM (Chowdhery et al., 2022) leveraging a novel ML system, Pathways to achieve SOTA few-shot learning and breakthrough performance in multi-step reasoning and multilingual tasks. PaLM 2 (Anil et al., 2023) builds upon this by improving multilingual capabilities, reasoning, and compute efficiency. FLAN (Wei et al., 2021)introduces a model that enhances zero-shot learning in language models through instruction tuning, a process of fine-tuning models on tasks described via instructions, showing superior performance over models like GPT-3 in various NLP tasks by utilizing natural language instructions, model scale, and diversified fine-tuning datasets. A significant body of work has also been dedicated to the fine-tuning methods that optimize LLMs for specialized tasks, offering critical insights into effective techniques such as data augmentation, transfer learning, and hyperparameter tuning. For example, RoBERTa (Liu et al., 2019b) refines BERT pretraining approach, revealing through a meticulous replication study that by optimizing key hyperparameters and training data size, BERT was significantly undertrained and could achieve state-of-the-art results on several NLP benchmarks, thus emphasizing the pivotal role of design choices in model performance. Ethical considerations, particularly those related to bias and fairness, have also seen growing empirical examination. These research efforts collectively contribute to the ongoing refinement of LLMs, enhancing their practicality and ethical standards, and ensuring their effective deployment across a wide range of applications.

Theoretical LLMs A number of work have studied the Large Language Models. Several work have tried to explain the LLMs in different kind of angles Panigrahi et al. (2023a); Sanford et al. (2023); Tarzanagh et al. (2023); Arora & Goyal (2023); Malladi et al. (2023b); Panigrahi et al. (2023b). There are also a number of work studied the attention scheme in computation level, such as Kitaev et al. (2020); Wang et al. (2020); Zandieh et al. (2023); Alman & Song (2023); Deng et al. (2023); Brand et al. (2023); Malladi et al. (2023a)

3 `Multi-LoRA`: A Parameter-Sharing Framework for Efficient Fine-Tuning

In this section, we present `Multi-LoRA`, a framework that enables multiple fine-tuning tasks to share a same low-rank module while maintaining their identities. This results in a nearly 50% parameter reduction compared to LoRA. To make the following discussion clear, for matrices A, B, we use subscript A_i, B_i to denote the low-rank modules for the i-th task, and we use $A(t), B(t)$ to denote the low-rank modules for the t iteration. For example, $B_i(t)$ would denote the low-rank module for the i-th task at iteration t.

At a high level, `Multi-LoRA` resembles that of LoRA: the algorithm tunes two low-rank modules iteratively. The key difference here is that `Multi-LoRA` further freezes the module A and shares it across all different tasks. This approach seems to weaken the LoRA framework, as it reduces the expressivity of the model by freezing A and further, if the tasks are highly unrelated, then sharing A would intuitively be a bad idea. We show both theoretically and empirically that this is not the case. Theoretically, we provide a mathematical formulation of our `Multi-LoRA` framework and observe a partially decomposable structure of this formulation. This enables one to transmit good local structure to the global loss, therefore explaining the good performance of `Multi-LoRA`. Empirically, we observe `Multi-LoRA` exhibits comparable performance as LoRA while reducing number of parameters dramatically.

4 Mathematical Formulation of Multiple Parallel Fine-Tuning

In this section, we provide a mathematical model for `Multi-LoRA`from the perspective of loss functions. Specifically, we show that by our formulation of global loss, the local structure can be propagated to global.

Algorithm 1 Parameter reduction through sharing a low-rank module across all fine-tuning tasks. We use W_0 to denote the pre-trained model weights, r to denote the rank parameter of low-rank modules, L to denote the loss function that takes in a pre-trained weight, a low-rank factorization of fine-tuning weight and an m-dimensional data point. k denotes the total number of tasks to be fine-tuned, and $X_1, \dots, X_k$ denote specific datasets for each task. Finally, let T denote the total number of fine-tuning episodes.

```
1: procedure MULTI-LORA($W_0 \in \mathbb{R}^{d\times m}, r, L : \mathbb{R}^{d\times m} \times \mathbb{R}^{d\times r} \times \mathbb{R}^{r\times m} \times \mathbb{R}^m \to \mathbb{R}, k, X_1 \in \mathbb{R}^{m\times n_1}, \dots, X_k \in \mathbb{R}^{m\times n_k}, T$)
2:     /* Initialization stage: initialize $A(0)$ with random Gaussian and $B_i(0)$ to $\mathbf{0}_{r\times m}$. */
3:     Initialize each entry of $A(0)$ as independent $\mathcal{N}(0, 1)$
4:     for $i = 1 \to k$ do
5:         $B_i(0) \leftarrow \mathbf{0}_{r\times m}$
6:     end for
7:     /* Fine-tuning stage: freeze $W_0$ and $A(0)$, while tuning for each $B_i(t)$. */
8:     for $t = 1 \to T$ do
9:         $A(t) \leftarrow A(t-1)$                                    ▷ Freeze shared module $A$.
10:        for $i = 1 \to k$ do
11:            Update $B_i(t)$ using $B_i(t-1)$ and $\nabla_B L(W_0, A(t), B_i(t-1), X_i)$
12:        end for
13:    end for
14:    return $A(T), \{B_i(T)\}_{i=1}^k$
15: end procedure
```

4.1 Notations

We use $[n]$ for a positive integer n to denote the set $\{1, 2, \dots, n\}$. Given a vector x, we use $\|x\|_2$ to denote its ℓ_2 norm. Given a matrix A, we use $\|A\|$ to denote its spectral norm and $\|A\|_F$ to denote its Frobenious norm. Given a matrix A, we use $A_{i,*}$ to denote its i-th row and $A_{*,j}$ to denote its j-th column. We use $\|A\|_0$ to denote the ℓ_0 semi-norm of A that measures the number of nonzero entries in A.

Given a rank-k, $m \times n$ real matrix A, we use $\kappa(A)$ to denote its condition number: $\kappa(A) = \frac{\sigma_1(A)}{\sigma_k(A)}$ where $\sigma_1(A), \dots, \sigma_k(A)$ are singular values of A sorted in magnitude. When A is clear from context, we often use κ directly.

For a $n \times n$ matrix A, we say it is positive semi-definite (PSD, $A \succeq 0$) if for all x, $x^\top Ax \geq 0$. We say it is positive definite (PD, $A \succ 0$) if for all $x \in \mathbb{R}^n \backslash \mathbf{0}_n$, we have $x^\top Ax > 0$.

4.2 Assumptions and Definitions

In this section, we provide a mathematical formulation of the multiple parallel fine-tuning tasks model studied in this paper.

Definition 4.1 (A mathematical interpretation of the model, restatement of Defintion 1.1)**.** *Let $L : \mathbb{R}^d \times \mathbb{R}^m \to \mathbb{R}$ denote a loss function. Let $k \geq 1$ denote a positive integer.*

Define the global loss function $\mathcal{L} : \mathbb{R}^d \times \mathbb{R}^{mk} \to \mathbb{R}$ as follows

$$\mathcal{L}(x, y) := \sum_{i=1}^k L(x, y_i)$$

where $y \in \mathbb{R}^{mk}$ is a vector that concatenates all k parameters for each y_i:

$$y := \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_k \end{bmatrix}.$$

We would like to highlight the connection between this formulation and the fine-tuning scheme we are to introduce in this paper. Our algorithm would resemble that of LoRA, with the significant difference that if there are k tasks, LoRA would maintain k pairs of matrices A_i, B_i, but our algorithm would *share* a single A across all k tasks while only let each task vary their B_i's. In the above formulation, one can view x as a shared parameter across all local losses and y_i's are the customized parameters that are unique to each task. The global loss function is heavily inspired by federated learning and the FedAvg paradigm McMahan et al. (2017). We would also like to remark that we can vary the loss function by replacing L with L_i and our results still hold. For the simplicity of presentation, we stick to the same loss across all tasks.

The gradient of the loss function can be compactly expressed in the following way:

Claim 4.2. *We have*

$$\frac{\mathrm{d}\mathcal{L}(x,y)}{\mathrm{d}(x,y)} = \begin{bmatrix} \sum_{i=1}^k \frac{\mathrm{d}L(x,y_i)}{\mathrm{d}x} \\ \frac{\mathrm{d}L(x,y_1)}{\mathrm{d}y_1} \\ \vdots \\ \frac{\mathrm{d}L(x,y_k)}{\mathrm{d}y_k} \end{bmatrix}$$

Proof. The proof of this statement follows straightforward by using chain rule. □

The form of the gradient enables us to perform a "partial decomposition": we can decompose it into a term contributed by the shared term, and k terms correspond to individual parameters. This motivates us to prove structural property of the global loss based on individual loss. Below, we show several standard properties of loss functions, such as Lipschitzness, smoothness and (strong) convexity can be readily propagated from individual loss to global loss.

4.3 LIPSCHITZ PROPERTY

Definition 4.3. *We say function $L : \mathbb{R}^d \times \mathbb{R}^m \to \mathbb{R}$ is γ-Lipschitz if*

$$|L(x,y_i) - L(\widetilde{x},\widetilde{y}_i)| \leq \gamma \cdot \| \begin{bmatrix} x \\ y_i \end{bmatrix} - \begin{bmatrix} \widetilde{x} \\ \widetilde{y}_i \end{bmatrix} \|_2$$

Lemma 4.4 (Formal version of Lemma 1.2)**.** *If $L : \mathbb{R}^d \times \mathbb{R}^m \to \mathbb{R}$ is γ-Lipschitz, then $\mathcal{L} : \mathbb{R}^d \times \mathbb{R}^{mk} \to \mathbb{R}$ is (γk)-Lipschitz.*

Proof. We can show

$$\begin{aligned} |\mathcal{L}(x,y) - \mathcal{L}(\widetilde{x},\widetilde{y})|^2 &= |\sum_{i=1}^k (L(x,y_i) - L(\widetilde{x},\widetilde{y}_i))|^2 \\ &\leq k \sum_{i=1}^k |L(x,y_i) - L(\widetilde{x},\widetilde{y}_i)|^2 \\ &\leq k \sum_{i=1}^k \gamma^2 \| \begin{bmatrix} x \\ y_i \end{bmatrix} - \begin{bmatrix} \widetilde{x} \\ \widetilde{y}_i \end{bmatrix} \|_2^2 \\ &= k\gamma^2 \sum_{i=1}^k (\|x-\widetilde{x}\|_2^2 + \|y_i - \widetilde{y}_i\|_2^2) \\ &\leq k^2\gamma^2 (\|x-\widetilde{x}\|_2^2 + \sum_{i=1}^k \|y_i - \widetilde{y}_i\|_2^2) \\ &= k^2\gamma^2 (\|x-\widetilde{x}\|_2^2 + \|y-\widetilde{y}\|_2^2) \\ &= k^2\gamma^2 \cdot \| \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} \widetilde{x} \\ \widetilde{y} \end{bmatrix} \|_2^2 \end{aligned}$$

where the first step follows from definition of $\mathcal{L}$, the second step follows from Cauchy-Schwarz inequality, the third step follows from loss function L is γ-Lipschitz, the forth step follows definition of ℓ_2 norm, the fifth step follows from simple algebra, the sixth step follows from definition of y.

Taking the square root of both sides of the above equation, then we have

$$|\mathcal{L}(x,y) - \mathcal{L}(\widetilde{x},\widetilde{y})| \leq k\gamma \cdot \| \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} \widetilde{x} \\ \widetilde{y} \end{bmatrix} \|_2$$

Thus, we complete the proof. □

4.4 SMOOTHNESS

Definition 4.5. *We say function $L : \mathbb{R}^d \times \mathbb{R}^m \to \mathbb{R}$ is β-smooth if*

$$\|\nabla L(x,y_i) - \nabla L(\widetilde{x},\widetilde{y}_i)\|_2 \leq \beta \cdot \| \begin{bmatrix} x \\ y_i \end{bmatrix} - \begin{bmatrix} \widetilde{x} \\ \widetilde{y}_i \end{bmatrix} \|_2$$

Here $\nabla L(x,y_i)$ denote the gradient.

Lemma 4.6 (Formal version of Lemma 1.3)**.** *If function L is β-smooth, then $\mathcal{L}$ is $(k\beta)$-smooth.*

Proof. We can show

$$\begin{aligned}
\|\nabla \mathcal{L}(x,y) - \nabla \mathcal{L}(\widetilde{x},\widetilde{y})\|_2^2 &= \| \sum_{i=1}^k (\nabla L(x,y_i) - \nabla L(\widetilde{x},\widetilde{y}_i))\|_2^2 \\
&\leq k \sum_{i=1}^k \|\nabla L(x,y_i) - \nabla L(\widetilde{x},\widetilde{y}_i)\|_2^2 \\
&\leq k \sum_{i=1}^k \beta^2 \| \begin{bmatrix} x \\ y_i \end{bmatrix} - \begin{bmatrix} \widetilde{x} \\ \widetilde{y}_i \end{bmatrix} \|_2^2 \\
&= k\beta^2 \sum_{i=1}^k (\|x-\widetilde{x}\|_2^2 + \|y_i - \widetilde{y}_i\|_2^2) \\
&\leq k^2\beta^2 (\|x-\widetilde{x}\|_2^2 + \sum_{i=1}^k \|y_i - \widetilde{y}_i\|_2^2) \\
&= k^2\beta^2 (\|x-\widetilde{x}\|_2^2 + \|y - \widetilde{y}\|_2^2) \\
&= k^2\beta^2 \cdot \| \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} \widetilde{x} \\ \widetilde{y} \end{bmatrix} \|_2^2
\end{aligned}$$

where the first step follows from definition of $\mathcal{L}$, the second step follows from Cauchy-Schwarz inequality, the third step follows from loss function L is β-smooth, the forth step follows definition of ℓ_2 norm, the fifth step follows from simple algebra, the sixth step follows from definition of y.

Taking the square root of both sides of the above equation, then we have

$$\|\nabla \mathcal{L}(x,y) - \nabla \mathcal{L}(\widetilde{x},\widetilde{y})\|_2 \leq k\beta \cdot \| \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} \widetilde{x} \\ \widetilde{y} \end{bmatrix} \|_2$$

Thus, we complete the proof. □

4.5 CONVEXITY

Definition 4.7. *We say function $L : \mathbb{R}^d \times \mathbb{R}^m \to \mathbb{R}$ is α-strongly convex if*

$$\nabla^2 L(x,y_i) \succeq \alpha \cdot I_{d+m}$$

Without any assumption on α_0, α, k relationship, the hessian $\nabla^2 \mathcal{L}(x,y)$ might not be psd due to certain counter-example (see Section A).

Lemma 4.8 (Formal version of Lemma 1.4)**.** *If the following conditions hold*

- *Let $k \geq 1$ denote a positive integer.*

- $\nabla_x^2 L(x, y_i) \preceq \alpha_0 \cdot I_d$, *where* $\alpha_0 > 0$.
- $L : \mathbb{R}^d \times \mathbb{R}^m \to \mathbb{R}$ *is* α*-strongly convex, where* $\alpha > 0$.
- *Let* $(k-1)(\alpha_0 - \alpha) \leq \alpha/2$.

Then we have $\mathcal{L} : \mathbb{R}^d \times \mathbb{R}^{km} \to \mathbb{R}$ *is* $\alpha/2$*-strongly convex.*

Proof. Note that

$$\nabla^2 \mathcal{L}(x, y) \in \mathbb{R}^{(d+km)\times(d+km)}$$

We can consider the Hessian matrix as $(k+1) \times (k+1)$ blocks.

For convenient, let us rewrite $\nabla^2 \mathcal{L}(x, y)$ as follows

$$\nabla^2 \mathcal{L}(x, y) = \begin{bmatrix} H_{0,0} & H_{0,1} & \cdots & H_{0,k} \\ H_{1,0} & H_{1,1} & \cdots & H_{1,k} \\ \vdots & \vdots & \ddots & \vdots \\ H_{k,0} & H_{k,1} & \cdots & H_{k,k} \end{bmatrix}$$

where $H_{0,0} \in \mathbb{R}^{d\times d}, H_{i,i} \in \mathbb{R}^{m\times m}$ for all $i \in [k]$, $H_{0,i} \in \mathbb{R}^{m\times d}, H_{i,0} \in \mathbb{R}^{d\times m}$ and $H_{i,j} \in \mathbb{R}^{m\times m}$. Since y_i and y_j has no correlation in loss function, thus $H_{i,j} \in \mathbf{0}_{m\times m}$ is an all zero matrix.

From assumption we know L is α-strongly convex,thus for all $i \in [k]$, we know

$$\begin{bmatrix} H_{0,0} & H_{0,i} \\ H_{i,0} & H_{i,i} \end{bmatrix} \succeq \alpha \cdot I_{m+d}$$

Let us pick up a vector $z \in \mathbb{R}^{d+mk}$, let $z_0 \in \mathbb{R}^d$, let $z_i \in \mathbb{R}^m$ for all $i \in [k]$. Then we have

$$\begin{aligned} z^\top \nabla^2 \mathcal{L}(x, y) z &= z_0^\top H_{0,0} z_0 + \sum_{i=1}^k z_i^\top H_{i,i} z_i + \sum_{i=1}^k z_i^\top H_{i,0} z_0 + z_0^\top H_{0,i} z_i^\top \\ &= k \cdot z_0^\top H_{0,0} z_0 + \sum_{i=1}^k z_i^\top H_{i,i} z_i + \sum_{i=1}^k z_i^\top H_{i,0} z_0 + z_0^\top H_{0,i} z_i^\top - (k-1) \cdot z_0^\top H_{0,0} z_0 \\ &= \sum_{i=1}^k (z_0^\top H_{0,0} z_0 + z_i^\top H_{i,i} z_i + z_i^\top H_{i,0} z_0 + z_0^\top H_{0,i} z_i) - (k-1) \cdot z_0^\top H_{0,0} z_0 \\ &\geq \sum_{i=1}^k \alpha(\|z_0\|_2^2 + \|z_i\|_2^2) - (k-1) z_0^\top H_{0,0} z_0 \\ &\geq \sum_{i=1}^k \alpha(\|z_0\|_2^2 + \|z_i\|_2^2) - (k-1)\alpha_0 \|z_0\|_2^2 \\ &= \alpha\|z\|_2^2 - (k-1) \cdot (\alpha_0 - \alpha)\|z_0\|_2^2 \\ &\geq \alpha\|z\|_2^2 - (k-1) \cdot (\alpha_0 - \alpha)\|z\|_2^2 \\ &\geq \alpha\|z\|_2^2 - 0.5\alpha\|z\|_2^2 \\ &= 0.5\alpha\|z\|_2^2 \end{aligned}$$

where the first follows from definition of Hessian, the second step follows from simple algebra, the third step follows from simple algebra, the forth step follows from L is α-strongly convex, the fifth step follows from $\nabla_x^2 L(x, y_i) \preceq \alpha_0 \cdot I_d$, the sixth step follows from simple algebra, the seventh step follows from $\|z_0\|_2^2 \leq \|z\|_2^2$, the eight step follows from assumption of α_0 and α, the ninth step follows from simple algebra. □

Table 1: Performance of RoBERTa$_{\text{base}}$ with various adaptation techniques on the GLUE benchmark. For MNLI we provide the overall (matched and mismatched) accuracy, for CoLA we present Matthew's correlation, and for STS-B we report Pearson correlation. For the remaining tasks we disclose the accuracy. For all these metrics, a higher value is preferable. Numbers marked with * are referenced from previously published works. The best in **bold** and the second best $\underline{\text{underlined}}$.

Method	# of Trainable Parameters	MNLI	SST-2	MRPC	CoLA	QNLI	QQP	RTE	STS-B	Avg.
Pre-trained	0M	$33.5_{\pm 1.3}$	$50.5_{\pm .8}$	$61.0_{\pm 16.5}$	$0.0_{\pm .0}$	$50.3_{\pm .4}$	$42.1_{\pm 11.8}$	$48.4_{\pm 2.4}$	$-2.2_{\pm 5.9}$	35.5
Fine-Tuning*	1000.0M	**87.6**	$\underline{94.8}$	**90.2**	**63.6**	$\underline{92.8}$	**91.9**	78.7	$\underline{91.2}$	$\underline{86.4}$
LoRA*	2.4M	$\underline{87.5}_{\pm .3}$	$\mathbf{95.1}_{\pm \mathbf{.2}}$	$\underline{89.7}_{\pm .7}$	$\underline{63.4}_{\pm 1.2}$	$\mathbf{93.3}_{\pm \mathbf{.3}}$	$\underline{90.8}_{\pm .1}$	$\mathbf{86.6}_{\pm \mathbf{.7}}$	$\mathbf{91.5}_{\pm \mathbf{.2}}$	**87.2**
`Multi-LoRA`	1.3M	$86.1_{\pm .2}$	$94.7_{\pm .4}$	$88.4_{\pm .7}$	$57.9_{\pm .8}$	$92.0_{\pm .2}$	$89.6_{\pm .1}$	$\underline{81.8}_{\pm 2.2}$	$90.5_{\pm .3}$	85.1

Table 2: BLEU score (↑) of GPT-2 medium (M) with various adaptation techniques on the E2E NLG Challenge, DART and WebNLG. Numbers marked with * are referenced from previously published works. The best numbers are in **bold** and the second best ones are $\underline{\text{underlined}}$.

Method	# of Trainable Parameters	E2E NLG Challenge	DART	WebNLG	Avg.
Fine-Tuning*	1065M	$\underline{68.2}$	$\underline{46.2}$	46.5	53.6
LoRA	1.1M	**68.9**	**46.2**	**54.9**	**56.7**
`Multi-LoRA`	0.7M	67.3	42.2	$\underline{51.6}$	$\underline{53.7}$

5 EXPERIMENTS

RoBERTa on NLU. We conduct experiments on the base model of RoBERTa (Liu et al., 2019b) for natural language understanding (NLU) tasks. RoBERTa refined the original pre-training methodology set forth in BERT (Devlin et al., 2018), enhancing its performance on various tasks without significantly increasing the number of parameters. We assess the effectiveness of various efficiency-focused adaptation methods on GLUE tasks using RoBERTa base (125M) from the HuggingFace Transformers library. Additionally, we replicate the configurations specified in studies by Houlsby et al. (2019). To maintain an equitable comparison, we alter two key elements in our evaluation approach for LoRA. Firstly, we maintain a consistent batch size across all tasks and fix the sequence length at 128. Secondly, the model is initialized using pre-trained settings for MRPC, RTE, and STS-B, as opposed to starting with a model pre-adapted to MNLI like in the fine-tuning baseline. To implement our algorithm, we utilize the same LoRA$_A$ layers (randomly initialized by default configuration) for all the 8 tasks and only tune LoRA$_B$ layers on each task respectively. We set the dimension of the low-rank matrices $r = 8$ and the scaling factor for the weight matrices LoRA$_\alpha = 16$ for all the experiments. the maximum sequence length is chosen as 512, the warmup ratio is selected as 0.06.

We compare our method with pre-trained RoBERTa$_{\text{base}}$ model, RoBERTa$_{\text{base}}$ fully fine-tuned, and RoBERTa$_{\text{base}}$ fine-tuned with LoRA. Since the results of Fine-Tuning and LoRA are borrowed from literature and run 5 times independently, we also run our experiments (our method and the evaluation of Pre-trained model) 5 times using random seed $0, 1, 2, 3, 4$. As shown in Tab. 1, our proposed `Multi-LoRA` can achieve comparable results with state-of-the-art fine-tuning approaches with significantly smaller parameter size in total.

GPT-2 on NLG. Having demonstrated `Multi-LoRA`'s effectiveness in comprehensive fine-tuning on NLU, we aim to investigate whether `Multi-LoRA` maintains its superiority on NLG (Natural Language Generation) tasks. We conduct experiments of GPT-2 medium models on E2E NLG Challenge, WebNLG and DART. We set the dimension of the low-rank matrices $r = 4$, the scaling factor for the weight matrices LoRA$_\alpha = 32$ and dropout probability of the LoRA layers as 0.1 for all the experiments. The maximum sequence length is chosen as 512. As shown in Tab. 2, our `Multi-LoRA` is able to maintain high NLG performance with extremely small parameter size and is better than fully fine-tuning which training parameter size is thousands of times more than ours.

REFERENCES

Josh Alman and Zhao Song. Fast attention requires bounded entries. In *NeurIPS*. arXiv preprint arXiv:2302.13214, 2023.

Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*, 2023.

Sanjeev Arora and Anirudh Goyal. A theory for emergence of complex skills in language models. *arXiv preprint arXiv:2307.15936*, 2023.

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

Michael Bommarito II and Daniel Martin Katz. Gpt takes the bar exam. *arXiv preprint arXiv:2212.14402*, 2022.

Jan van den Brand, Zhao Song, and Tianyi Zhou. Algorithm and hardness for dynamic attention maintenance in large language models. *arXiv preprint arXiv:2304.02207*, 2023.

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

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023. URL `https://lmsys.org/blog/2023-03-30-vicuna/`.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

Alexandra Chronopoulou, Christos Baziotis, and Alexandros Potamianos. An embarrassingly simple approach for transfer learning from pretrained language models. *arXiv preprint arXiv:1902.10547*, 2019.

Yichuan Deng, Zhihang Li, and Zhao Song. Attention scheme inspired softmax regression. *arXiv preprint arXiv:2304.10411*, 2023.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 5(3):220–235, 2023.

Steven Y Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. A survey of data augmentation approaches for nlp. *arXiv preprint arXiv:2105.03075*, 2021.

Jihun Hamm and Daniel D. Lee. Grassmann discriminant analysis: A unifying view on subspace-based learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pp. 376–383, New York, NY, USA, 2008. Association for Computing Machinery.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pp. 2790–2799. PMLR, 2019.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Enkelejda Kasneci, Kathrin Seßler, Stefan Küchemann, Maria Bannert, Daryna Dementieva, Frank Fischer, Urs Gasser, Georg Groh, Stephan Günnemann, Eyke Hüllermeier, et al. Chatgpt for good? on opportunities and challenges of large language models for education. *Learning and individual differences*, 103:102274, 2023.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504*, 2019a.

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

Sadhika Malladi, Tianyu Gao, Eshaan Nichani, Alex Damian, Jason D Lee, Danqi Chen, and Sanjeev Arora. Fine-tuning language models with just forward passes. *arXiv preprint arXiv:2305.17333*, 2023a.

Sadhika Malladi, Alexander Wettig, Dingli Yu, Danqi Chen, and Sanjeev Arora. A kernel-based view of language model fine-tuning. In *International Conference on Machine Learning*, pp. 23610–23641. PMLR, 2023b.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, and Sayak Paul. Peft: State-of-the-art parameter-efficient fine-tuning methods. `https://github.com/huggingface/peft`, 2022.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pp. 1273–1282. PMLR, 2017.

Harsha Nori, Nicholas King, Scott Mayer McKinney, Dean Carignan, and Eric Horvitz. Capabilities of gpt-4 on medical challenge problems. *arXiv preprint arXiv:2303.13375*, 2023.

OpenAI. Gpt-4 technical report. *CoRR, abs/2303.08774*, 2023.

Abhishek Panigrahi, Sadhika Malladi, Mengzhou Xia, and Sanjeev Arora. Trainable transformer in transformer. *arXiv preprint arXiv:2307.01189*, 2023a.

Abhishek Panigrahi, Nikunj Saunshi, Haoyu Zhao, and Sanjeev Arora. Task-specific skill localization in fine-tuned language models. *arXiv preprint arXiv:2302.06600*, 2023b.

Tung Phung, Victor-Alexandru Pădurean, José Cambronero, Sumit Gulwani, Tobias Kohn, Rupak Majumdar, Adish Singla, and Gustavo Soares. Generative ai for programming education: Benchmarking chatgpt, gpt-4, and human tutors. *International Journal of Management*, 21(2):100790, 2023.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

Clayton Sanford, Daniel Hsu, and Matus Telgarsky. Representational strengths and limitations of transformers. *arXiv preprint arXiv:2306.02896*, 2023.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*, 2021.

Connor Shorten, Taghi M Khoshgoftaar, and Borko Furht. Text data augmentation for deep learning. *Journal of big Data*, 8:1–34, 2021.

Zhao Song, Yitan Wang, Zheng Yu, and Lichen Zhang. Sketching for first order method: efficient algorithm for low-bandwidth channel and vulnerability. In *International Conference on Machine Learning (ICML)*, pp. 32365–32417. PMLR, 2023.

Davoud Ataee Tarzanagh, Yingcong Li, Christos Thrampoulidis, and Samet Oymak. Transformers as support vector machines. *arXiv preprint arXiv:2308.16898*, 2023.

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

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.

Guangyu Wang, Guoxing Yang, Zongxin Du, Longjun Fan, and Xiaohu Li. Clinicalgpt: Large language models finetuned with diverse medical data and comprehensive evaluation. *arXiv preprint arXiv:2306.09968*, 2023.

Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*, 2021.

Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David Rosenberg, and Gideon Mann. Bloomberggpt: A large language model for finance. *arXiv preprint arXiv:2303.17564*, 2023.

Benfeng Xu, Licheng Zhang, Zhendong Mao, Quan Wang, Hongtao Xie, and Yongdong Zhang. Curriculum learning for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 6095–6104, 2020.

Hongyang Yang, Xiao-Yang Liu, and Christina Dan Wang. Fingpt: Open-source financial large language models. *arXiv preprint arXiv:2306.06031*, 2023.

Kang Min Yoo, Dongju Park, Jaewook Kang, Sang-Woo Lee, and Woomyoung Park. GPT3Mix: Leveraging large-scale language models for text augmentation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pp. 2225–2239, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-emnlp.192. URL https://aclanthology.org/2021.findings-emnlp.192.

Amir Zandieh, Insu Han, Majid Daliri, and Amin Karbasi. Kdeformer: Accelerating transformers via kernel density estimation. *arXiv preprint arXiv:2302.02451*, 2023.

APPENDIX

A COUNTEREXAMPLE FOR STRONG CONVEXITY

The following counterexample indicates that we need extra assumption to make sure that $\mathcal{L}$ has strongly convex property.

Lemma A.1 (Counterexample). *Let $H \in \mathbb{R}^{(k+1)\times(k+1)}$ denote the following matrix*

$$H = \begin{bmatrix} 4 & -3 & -3 & -3 & \cdots & -3 \\ -3 & 4 & 0 & 0 & \cdots & 0 \\ -3 & 0 & 4 & 0 & \cdots & 0 \\ -3 & 0 & 0 & 4 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ -3 & 0 & 0 & 0 & \cdots & 4 \end{bmatrix}$$

We know that

- **Part 1.** $\langle H, \mathbf{1}_{(k+1)\times(k+1)} \rangle = -2k + 4$
- **Part 2.** $\begin{bmatrix} 4 & -3 \\ -3 & 4 \end{bmatrix} \succeq 0$

Proof. **Proof of Part 1.**

We can show

$$\begin{aligned} \langle H, \mathbf{1}_{(k+1)\times(k+1)} \rangle = & ~ 4(k+1) - 6k \\ = & ~ -2k + 4 \end{aligned}$$

Proof of Part 2.

We have

$$\begin{aligned} \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 4 & -3 \\ -3 & 4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} & = 4x^2 + 4y^2 - 6xy \\ & = x^2 + y^2 + 3(x^2 - 2xy + y^2) \\ & = x^2 + y^2 + 3(x-y)^2 \\ & \geq 0 \end{aligned}$$

□

B SOFTMAX REGRESSION: A CASE STUDY

As a toy example, we consider the softmax regression problem Deng et al. (2023). This problem can be viewed as a simplification of two-layer transformer network. To better understand how effectiveness our method would be, we empirically verify various parameters for this problem.

Definition B.1 (Softmax regression). *Given $A \in \mathbb{R}^{n\times d}$ and $b \in \mathbb{R}^n$, we define loss function $L : \mathbb{R}^d \to \mathbb{R}$ as*

$$L(x) := 0.5 \| \langle \exp(Ax), \mathbf{1}_n \rangle^{-1} \exp(Ax) - b \|_2^2$$

Here

- $\exp(Ax)_i = \exp((Ax)_i)$ *for all $i \in [n]$*
- $\mathbf{1}_n \in \mathbb{R}^n$ *denote the length-n vector where all the entries are ones*
- $\langle a, b \rangle = \sum_{i=1}^n a_i b_i$.

Definition B.2. *Let* $f(x) = \langle \exp(Ax), \mathbf{1}_n \rangle^{-1} \exp(Ax)$. *Let* $c(x) = f(x) - b$.

Lemma B.3 (see Definition 5.7 in page 14 in Deng et al. (2023))**.** *Let* $g(x) \in \mathbb{R}^d$ *denote the gradient of softmax regression loss (see Definition B.1), then we have*

$$g(x) = A^\top(-f(x)\langle c(x), f(x)\rangle + \mathrm{diag}(f(x))c(x))$$

Lemma B.4 (see Definition 6.1 in page 28 in Deng et al. (2023))**.** *Let* $H(x) \in \mathbb{R}^{d\times d}$ *denote the Hessian of softmax regression loss (see Definition B.1), we have*

$$H(x) = \underbrace{A^\top}_{d\times n} \underbrace{B(x)}_{n\times n} \underbrace{A}_{n\times d}$$

where

$$\begin{aligned} B(x) := & \langle 3f(x) - 2b, f(x)\rangle f(x)f(x)^\top \\ & + (b \circ f(x))f(x)^\top + f(x)(b\circ f(x))^\top \\ & + \langle f(x) - b, f(x)\rangle \cdot \mathrm{diag}(f(x)) \\ & + \mathrm{diag}((2f(x) - b)\circ f(x)) \end{aligned}$$

Here

- $(a \circ b)_i = a_i b_i$ *for all* $i \in [n]$

Remark B.5. *Note that* $B(x)$ *is a constructed as three rank-*1 *matrices with two diagonal matrices.*

- Function is α-Strongly convex
$$H(x) \succeq \alpha \cdot I_d$$
- Function is β-Smoothness (this is equivalent to gradient of function is β-Lipschitz)
$$\|H(x)\| \le \beta \iff \|g(x) - g(y)\|_2 \le \beta \cdot \|x - y\|_2$$
- Hessian of Function is M-Lipschitz
$$\|H(x) - H(y)\| \le M \cdot \|x - y\|_2$$
- Function is γ-Lipschitz (this is equivalent to gradient of function is bounded)
$$\|L(x) - L(y)\|_2 \le \gamma \cdot \|x - y\|_2 \iff \|g(x)\|_2 \le \gamma$$

Table 3: Numerical estimations of softmax regression problem for various choices of (n, d). We perform these experiments by randomly sampling point x and constructing the Hessian matrix, then performing estimations. We report the average over 10000 random samples.

n	d	α	β	M	γ
$n = 10$	$d = 5$	0.00	148.03	26.83	0.60
$n = 10$	$d = 10$	0.00	272.52	23.65	0.74
$n = 10$	$d = 20$	0.00	540.98	21.49	0.89
$n = 20$	$d = 5$	0.00	447.95	74.38	0.54
$n = 20$	$d = 10$	0.00	361.00	26.47	0.77
$n = 20$	$d = 20$	0.00	1309.38	52.23	0.91
$n = 20$	$d = 40$	0.00	2511.67	58.29	1.24