Can Looped Transformers Learn to Implement Multi-step Gradient Descent for In-context Learning?

Khashayar Gatmiry [1] **Nikunj Saunshi** [2] **Sashank Reddi** [2] **Stefanie Jegelka** [1] **Sanjiv Kumar** [2]

Abstract

The remarkable capability of Transformers to show reasoning and few-shot abilities, without any fine-tuning, is widely conjectured to stem from their ability to implicitly simulate a multi-step algorithms – such as gradient descent – with their weights in a single forward pass. Recently, there has been progress in understanding this complex phenomenon from an expressivity point of view, by demonstrating that Transformers can express such multi-step algorithms. However, our knowledge about the more fundamental aspect of its learnability, beyond single layer models, is very limited. In particular, *can training Transformers enable convergence to algorithmic solutions*? In this work we resolve this for in-context linear regression with linear looped Transformers – a multi-layer model with weight sharing that is conjectured to have an inductive bias to learn fix-point iterative algorithms. More specifically, for this setting we show that the global minimizer of the population training loss implements multi-step preconditioned gradient descent, with a preconditioner that adapts to the data distribution. Furthermore, we show a fast convergence for gradient flow on the regression loss, despite the non-convexity of the landscape, by proving a novel gradient dominance condition. To our knowledge, this is the first theoretical analysis for multi-layer Transformer in this setting. We further validate our theoretical findings through synthetic experiments.

1. Introduction

Transformers (Vaswani et al., 2017) have completely revolutionized the field of machine learning and have led to state-of-the-art models for various natural language and vision tasks. Large scale Transformer models have demonstrated remarkable capabilities to solve many difficult problems, including those requiring multi-step reasoning through large language models (Brown et al., 2020; Wei et al., 2022b). One such particularly appealing property is their few-shot learning ability, where the functionality and predictions of the model adapt to additional context provided in the input, without having to update the model weights. This ability of the model, typically referred to as "in-context learning", has been crucial to their success in various applications. Recently, there has been a surge of interest to understand this phenomenon, particularly since Garg et al. (2022) empirically showed that Transformers can be *trained* to solve many in-context learning problems based on linear regression and decision trees. Motivated by this empirical success, Von Oswald et al. (2023); Akyürek et al. (2022) theoretically showed the following intriguing expressivity result: multi-layer Transformers with linear self-attention can implement gradient descent for linear regression where each layer of Transformer implements one step of gradient descent. In other words, they hypothesize that the in-context learning ability results from approximating gradient-based few-shot learning within its forward pass. Panigrahi et al. (2023), further, extended this result to more general model classes.

While such an approximation is interesting from the point of view of expressivity, it is unclear if the Transformer model can *learn* to implement such algorithms. To this end, Ahn et al. (2023); Zhang et al. (2023) theoretically show, in a Gaussian linear regression setting, that the global minimizers of a one-layer model essentially simulate a single step of preconditioned gradient descent, and that gradient flow converges to this solution. Ahn et al. (2023) further show for the multi-layer case that a single step of gradient descent can be implemented by some stationary points of the loss. However, a fundamental characterization of all the stationary points for multi-layer Transformer, and the convergence to a stationary point that implements multi-step gradient descent, remains a challenging and important open question.

In this work, we focus our attention on the *learnability* of such multi-step algorithms by Transformer models. Instead of multi-layer models, we consider a closely related but

[1]MIT [2]Google Research. Correspondence to: Khashayar Gatmiry <gatmiry@mit.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

different class of models called *looped Transformers*, where the same Transformer block is looped multiple times for a given input. Since the expectation from multi-layer models is to simulate an iterative procedure like multi-step gradient descent, looped models are a fairly natural choice to implement this. There is growing interest in looped models with recent results (Giannou et al., 2023) theoretically showing that the iterative nature of the *looped Transformer* model can be used to simulate a programmable computer, thus allowing looped models to solve problems requiring arbitrarily long computations. Looped Transformer models are also conceptually appealing for learning iterative optimization procedures — the sharing of parameters across different layers, in principle, can provide a better inductive bias than multi-layer Transformers for learning iterative-optimization procedures. In fact, by employing a regression loss at various levels of looping, Yang et al. (2023) empirically find that looped Transformer models can be trained to solve in-context learning problems, and that looping on an example for longer and longer at test time converges to a desirable fixed-point solution, thus leading them to conjecture that *looped models can learn to express iterative algorithms*[1].

Despite these strong expressivity results for looped models and their empirically observed inductive bias towards simulating iterative algorithms, very little is known about the optimization landscape of looped models, and the theoretical convergence to desirable and interpretable iterative procedures. In fact, a priori it is not clear why training should even succeed given that looped models heavily use weight sharing and thus do not enjoy the optimization benefits of overparameterization that has been well studied (Buhai et al., 2020; Allen-Zhu et al., 2019). In this work, inspired by the empirical effectiveness of non-linear looped Transformers (Yang et al., 2023), we delve deeper into the problem of optimizing looped Transformers and theoretically study their landscape and convergence for in-context linear regression under the Gaussian data distribution setting used in (Ahn et al., 2023; Zhang et al., 2023). In particular, the main contributions of our paper are as follows:

- We obtain a precise characterization of the global minimizer of the population loss for a linear looped Transformer model, and show that it indeed implements multi-step preconditioned gradient descent with preconditioner close to the inverse of the population covariance matrix, as intuitvely expected.
- Despite the non-convexity of the loss landscape, we prove the convergence of the gradient flow for in-context linear regression with looped Transformer. To our knowledge, ours is the first such convergence result for a network beyond one-layer in this setting.
- To show this convergence, we prove that the loss satisfies a novel gradient-dominance condition, which guides the flow toward the global optimum. We expect this convergence proof to be generalizable to first-order iterative algorithms such as SGD with gradient estimate using a single random instance (De Sa et al., 2022).
- We further translate having a small sub-optimality gap, achieved by our convergence analysis, to the proximity of the parameters to the global minimizer of the loss. Furthermore we prove that the learned looped Transformer can extrapolate to out-of-distribution examples with skewed covariances, thus providing theoretical justification for the empirical findings in (Yang et al., 2023).

[1]The algorithm should converge to a desirable fixed point.

2. Related Work

In-context learning. Language models, especially at larger scale, have been shown to empirically demonstrate the intriguing ability to in-context learn various tasks on test data (Brown et al., 2020) More recently, Garg et al. (2022) formalized in-context learning ability and empirically observed that Transformers are capable of in-context learning some hypothesis classes such as linear or two layer neural networks, sometimes improving over conventional solvers. There have since been many paper studying this intriguing in-context learning phenomenon (Xie et al., 2022; Akyürek et al., 2022; Von Oswald et al., 2023; Bai et al., 2023)

Transformers in modeling iterative optimization algorithms. He et al. (2016) first observed that neural networks with residual connections are able to implicitly implement gradient descent. Von Oswald et al. (2023); Akyürek et al. (2022) use this line of reasoning for in-context learning by constructing weights for linear self-attention layers that can emulate gradient descent for various in-context learning tasks, including linear regression. Furthermore, Akyürek et al. (2022) empirically investigate various in-context learners that Transformers can learn as a function of depth and width. Also, von Oswald et al. hypothesize the ability of Transformers to (i) build an internal loss based on the specific in-context task, and (ii) optimize over that loss via an iterative procedure implemented by the Transformer weights. Panigrahi et al. (2023) generalize the results to show that Transformers can implement gradient descent over a smaller Transformer. Recently, Fu et al. (2023) empirically observe that Transformers can learn to emulate higher order algorithms such as Newton's method that converge faster than gradient descent.

Transformers in reasoning and computation. Indeed the in-context capabilities of Transformers in doing reasoning at test time and emulating an input-specific algorithm as a computer bear deep similarities (Dasgupta et al., 2022; Chung et al., 2022; Lewkowycz et al., 2022). Years before

the advent of Transformers, (Siegelmann & Sontag, 1994) study the Turing completeness of recurrent neural networks. To show the computational power of Transformer as a programmable device, (Pérez et al., 2019; 2021; Wei et al., 2022a) demonstrate that Transformers can simulate Turing machines. Furthermore, Lindner et al. (2023) propose using Transformer as programmable units and construct a compiler for the domain specific programming language called RASP. Pérez et al. (2019) further find a more efficient implementation of a programming language that is also Turing complete using looped Transformers, without scaling with the number of lines of code. More recently Giannou et al. (2023) used looped models to simulate a single-instruction program. Yang et al. (2023) show that looped Transformers can in-context learn data-fitting problems such as linear regression or decision trees as well as normal Transformers but with much fewer parameters.

3. Preliminaries

3.1. In-context Learning (ICL)

One of the surprising emergent abilities of large language models is their ability to adapt to specific learning tasks without requiring any additional fine tuning. Here we restate the formalism of in-context learning introduced by Garg et al. (2022). Suppose for a class of functions $\mathcal{F}$ and input domain $\mathcal{X}$, we sample an in-context learning instance $\mathcal{I} = (\{x_i\}_{i=1}^n, \{y_i\}_{i=1}^n, x_q)$ is then generated by sampling $x_i \sim \mathcal{D}_{\mathcal{X}}$ and $f \sim \mathcal{D}_{\mathcal{F}}$ independently, then calculating $\forall i \in \{1, 2, \ldots, n\}, y_i = f(x_i)$. An in-context learner M_θ parameterized by θ is then a mapping from the instance $\mathcal{I}$ to a prediction for the label of the query point $f(x_q)$. The population loss of M_θ is then defined as

$$\mathcal{L}(M_\theta) = \mathbb{E}_{\mathcal{D}_f, \mathcal{D}_{\mathcal{X}}}\left[\left(M_\theta(\mathcal{I}) - f(x_q)\right)^2\right] \tag{1}$$

3.2. Linear regression ICL setup

In this work, we consider linear regression in-context learning; namely, we assume sampling a linear regression instance is given by $\mathcal{I} = \left(\left\{x_i, y_i\right\}_{i=1}^n, x_q\right)$ where for $w^* \sim \mathcal{N}(0, {\Sigma^*_{d\times d}}^{-1})$, $x_i \sim \mathcal{N}(0, \Sigma^*_{d\times d})$ we have $y_i = f_{w^*}(x_i) = {w^*}^\top x_i$ for all $i \in [n]$. The goal is to predict the label of x_q, i.e. ${w^*}^\top x_q$. Define the data matrix $X \in \mathbb{R}^{d\times n}$, whose columns are the data points $\{x_i\}_{i=1}^n$:

$$X = [x_1, \ldots, x_n]$$

We further assume $n > d$, i.e. the number of samples is larger than the dimension. This combined with the fact that $\mathcal{I}$ is realizable implies that we can recover w^* from $\left\{x_i, y_i\right\}_{i=1}^n$ by the well-known pseudo-inverse formula:

$$w^* = (XX^\top)^{-1}Xy.$$

While a reasonable option for the context-learner $M_\theta(\mathcal{I})$ is to implement $(XX^\top)^{-1}Xy$, matrix inversion is arguably an operation that can be costly for Transformers to implement. On the other hand, it is known that linear regression can also be solved by first order algorithms that move along the negative gradient direction of the loss

$$\ell_2^2(w) = \|X^\top w^* - y\|^2.$$

Using a standard analysis for smooth convex optimization, since the Hessian of the loss $\|X^\top w^* - y\|^2$ is $XX^\top$ with condition number κ, gradient descent with step size $\frac{1}{\kappa}$ converges in $O(\kappa)$ iterations. This means that we need $O(\kappa)$ many layers in the Transformer to solve linear regression. Particularly, Von Oswald et al. (2023) show a simple weighting strategy for the key, query, and value matrices of a linear self-attention model so that it implements gradient descent, which we introduce in the next section.

3.3. Linear Self-attention layer

Here we define a single attention layer that forms the basis of the linear Transformer model we consider. Define the matrix $Z^{(0)}$, which we use as the input prompt to the Transformer, by combining the data matrix X, their labels y, and the query vector x_q as

$$Z^{(0)} = \begin{bmatrix} X & x_q \\ y^\top & 0 \end{bmatrix}$$

Following (Ahn et al., 2023; Schlag et al., 2021; Von Oswald et al., 2023), we consider the linear self-attention model $\text{Attn}^{lin}(Z; W_{k,q,v})$ defined as

$$\text{Attn}^{lin}(Z; W_{k,q,v}) \coloneqq W_v Z M (Z^\top W_k^\top W_q Z),$$
$$M \coloneqq \begin{bmatrix} I_{n\times n} & 0 \\ 0 & 0 \end{bmatrix} \in \mathbb{R}^{(n+1)\times(n+1)},$$

where W_k, W_q, W_v are the key, query, and value matrices, respectively and the index $k \times r$ below a matrix determines its dimensions. Furthermore, similar to (Ahn et al., 2023), we use mask matrix M in order to avoid the tokens corresponding to (x_i, y_i) to attend the query vector x_q, and combine product of the key and query matrices into $Q = W_k^\top W_q$ to obtain the following parameterization for the attention layer (we denote W_v by P):

$$\text{Attn}^{lin}(Z; Q, P) \coloneqq PZM(Z^\top QZ).$$

While linear attention is a simplification to the standard softmax attention used in practice, recent work (Ahn et al., 2024) shows that linear attention can provide valuable insights into dynamics of non-linear attention as well.

3.4. Linear looped Transformer

The linear looped transformer $\mathrm{TF}_L(Z^{(0)}; Q, P)$ can be defined by simply chaining L linear self-attention layers with shared parameters Q and P. In particular, we define

$$Z^{(t)} := Z^{(t-1)} - \frac{1}{n}\mathrm{Attn}^{lin}\left(Z; Q, P\right). \tag{2}$$

for all $t \in [L]$. Then, the output of an L layer looped transformer $\mathrm{TF}_L(Z^{(0)}; Q, P)$ just uses the $(d+1)\times(n+1)$ entry of matrix $Z^{(L)}$ i.e.,

$$\mathrm{TF}_L(Z^{(0)}; Q, P) = -Z^{(L)}{}_{(d+1),(n+1)}. \tag{3}$$

We note that the minus sign in the final output of the Transformer is only for simplicity of our expositions later on. For technical reasons, throughout the paper we assume n is large enough so that we have $\frac{8Ld^2}{\sqrt{n}} \leq \frac{1}{2^{2L}}$.

Can looped Transformer implement multi-step gradient descent? We first examine the expressivity of looped Transformer. The key idea is to leverage the existing result of one step preconditioned gradient descent from (Ahn et al., 2023) and use the loop structure of looped Transformer to show that it can implement multi-step preconditioned gradient descent. For completeness, we first restate the observation of (Ahn et al., 2023) that linear attention can implement a step of preconditioned gradient descent with arbitrary preconditioner.

Corollary 3.1 (Lemma 1 from Ahn et al. (2023))**.** *There is a choice of Q, P such that Linear looped Transformer architecture as defined in Equations (2) and (3) implements preconditioned gradient descent with arbitrary preconditioner A.*

Proof. For this, it suffices to pick

$$Q := \begin{bmatrix} A_{d\times d} & 0 \\ 0 & 0 \end{bmatrix}, P := \begin{bmatrix} 0_{d\times d} & 0 \\ 0 & 1 \end{bmatrix}, \tag{4}$$

then for the matrix $\begin{bmatrix} X & x_q \end{bmatrix} \in \mathbb{R}^{d\times(n+1)}$

$$\left[\mathrm{Attn}^{lin}\left(Z; Q, P\right)\right]_{(d+1),} = y^\top X^\top A \begin{bmatrix} X & x_q \end{bmatrix} \tag{5}$$

$$= -(0 - \frac{1}{n}A\nabla_w \ell_2^2(0))^\top X, \tag{6}$$

$$\left[\mathrm{Attn}^{lin}\left(Z; Q, P\right)\right]_{1:d,} = 0_{d\times n}, \tag{7}$$

where index $(k:r,)$ denotes the restriction of the matrix to its rows between k and r, and we used the fact that $\nabla_w \ell_2^2(0) = Xy$. But if we update w with the gradient of $\ell_2^2(w)$ preconditioned by A and step size $\frac{1}{n}$ and assuming $w_0 = 0$, then

$$w_1 = w_0 - \frac{1}{n}A\nabla_w \ell_2^2(w_0) = 0 - \frac{1}{n}A\nabla_w \ell_2^2(0).$$

Plugging this into Equation (6):

$$\left[\mathrm{Attn}^{lin}\left(Z; Q, P\right)\right]_{(d+1),1:n} = -w_1^\top X$$
$$\left[\mathrm{Attn}^{lin}\left(Z; Q, P\right)\right]_{(d+1),n+1} = -w_1^\top x_q.$$

Further, by using Equation (6) in Equation (2) we get

$$\left[Z^{(1)}\right]_{(d+1),1:n} = y^\top - w_1^\top X,$$
$$\left[Z^{(1)}\right]_{(d+1),n+1} = -w_1^\top x_q, \left[Z^{(1)}\right]_{1:d,} = X. \tag{8}$$

It is easy to see that Equations (8) hold for all $Z^{(t)}$ with w_1 substituted by corresponding w_t, thus, allowing implementation of multi-step gradient descent. □

3.5. Loss function on the weights

In previous section, while we observed that looped Transformer can implement preconditioned gradient descent, the choice of the preconditioner and its learnability by optimizing a loss function (e.g. squared error loss) still remain unclear. Following (Ahn et al., 2023; Zhang et al., 2023), we search for the best setting of matrices P, Q, where $Q := \begin{bmatrix} A_{d\times d} & 0 \\ 0 & 0 \end{bmatrix}$, i.e. only the top left $d \times d$ block can be non-zero, and $P := \begin{bmatrix} 0_{d\times d} & 0 \\ u^\top & 1 \end{bmatrix}$ for parameter vector $u \in \mathbb{R}^d$.

The population squared loss as a function of A and u is

$$L(A, u) = \mathbb{E}_{w^*, X}\left[(\mathrm{TF}_L(Z_0; Q, P) - y_q)^2\right].$$

We define a parameter $\delta := \left(\frac{8Ld}{\sqrt{n}}\right)^{1/(2L)}$ which governs the accuracy of our estimates, which goes to zero as $n \to \infty$.

We are now ready to present the main theoretical results. An important point to note is that, although the model being considered uses linear attention and the problem instance is linear regression, the resultant model is far from linear in its inputs and the corresponding optimization problem can be far from convex. This makes the analysis of multi-layer (multi-loop) models tricky. In the next section, we present theoretical convergence by getting around these issues.

4. Theoretical results for convergence

In this section, we show theoretical results for the *convergence* to a preconditioned gradient descent solution with a good preconditioner. Before presenting the main results, it is instructive to discuss the choice of the preconditioner A since it determines speed of convergence of w_i to the solution of the regression. Note that the exact solution of an over-determined linear regression instance (X, y) is

Table 1. Summary of main theoretical results and key assumptions

Results	Initialization	Basic Description
Theorem 4.1	Arbitrary	For the global minimizer of the loss (A^{opt}, u^{opt}), $u^{opt} = 0$ and the preconditioner part A^{opt} is close to Σ^{*-1}.
Theorem 4.2	$u = 0, \Sigma^* = I$	The gradient flow of the loss converge to a loss value with small suboptimality gap, in the proximity of the global minimizer in the parameter space
Theorem 4.3	$u = 0, \Sigma^* = I$	The loss satisfies a gradient dominance with power $\frac{2L-1}{L}$, given that the suboptimality gap is not too small
Theorem 5.4	$u = 0$	Small suboptimality gap implies closeness in the parameter space (In spectral distance).
Theorem 4.4	$u = 0$	Instance-dependent out of distribution generalization for the minimizer of the population loss.

$w = (XX^\top)^{-1}Xy$. This can be obtained only after one step of preconditioned gradient descent starting from the origin and using inverse of the data covariance matrix preconditioner

$$\Sigma = \frac{1}{n}\sum_{i=1}^{n} x_i x_i^\top.$$

In general, it may not be possible to pick the weights of the Transformer to ensure such a preconditioner for all possible regression instances as every instance (X, y) has its own data covariance matrix $\frac{1}{n}XX^\top$. But since x_i's are sampled i.i.d from $\mathcal{N}(0, \Sigma^*)$, it is known that the inverse of the data covariance matrix concentrates around the inverse of the population covariance Σ^*. Thus, a reasonable choice of A is the inverse of the population covariance matrix Σ^{*-1}.

In fact, Ahn et al. (2023) show that the global minimum of *one-layer* linear self-attention model under Gaussian data is the inverse of the population covariance matrix plus some small regularization term. However, the characterization of global minimizer(s) of the population loss for the multilayer case is largely missing. Specifically, *is there a global optimum for solving regression with Transformers that is close to gradient descent with preconditioner Σ^{*-1}?* In this work, we solve this open problem for looped Transformers; given that data is sampled iid from $\mathcal{N}(0, \Sigma^*)$, we show that the optimal looped Transformer under constraints stated in Section 3.5, A^{opt} will be close to Σ^{*-1}.

4.1. Global optimizer

In this section, we state our main results. First, we give a tight estimate on the set of global minimizers of the population loss, under the Gaussian assumption, for the looped Transformer model with arbitrary number of loops L.

Theorem 4.1 (Characterization of the optimal solution)**.** *Suppose $\{A^{opt}, u^{opt}\}$ are a global minimizer for $\mathcal{L}(A, u)$. Then, under condition $\frac{8Ld^2}{\sqrt{n}} \leq \frac{1}{2^{2L}}$,*

1. $\mathcal{L}(A^{opt}, u^{opt}) \leq \frac{8Ld^2 2^{2L}}{\sqrt{n}}$
2. $(1-c)\Sigma^{*-1} \preccurlyeq A^{opt} \preccurlyeq (1+c)\Sigma^{*-1}, c = 8\delta d^{1/(2L)}$ *and $u^{opt} = 0$, where recall $\delta := \left(\frac{8Ld}{\sqrt{n}}\right)^{1/(2L)}$.*

Remark. From Theorem 4.1, we first observe that the parameter u has no effect in obtaining a better regression solver and has to be set to zero in the global minimizer. This result was not known in the previous work (Ahn et al., 2023). A value of $u^{\text{opt}} = 0$ implies that the optimal looped Transformer exactly implements L steps of preconditioned gradient descent, with preconditioner A^{opt}.

Secondly, as discussed in Section 4, the choice of preconditioner plays an important role in how fast gradient descent converges to the solution of linear regression. Intuitively the inverse of the population covariance seems like a reasonable choice for a single fixed preconditioner, since it is close to the inverse of the data covariance for all linear regression instances. The above result shows that the global optimum is indeed very close to the inverse of the population covariance.

Precisely how close the optimum is to the population covariance depends on the parameter $\delta = \frac{4kd}{\sqrt{n}}$, which goes to zero as the number of examples in each prompt goes to infinity. In general, we do not expect the global minimizer to be exactly equal to Σ^{*-1}. Indeed for the case of one layer Transformer, which is equivalent to a loop-transformer with looping parameter $L = 1$, the global minimizer found in Ahn et al. (2023) is not exactly the inverse of the covariance matrix, but close to it. Even in their case, the distance goes to zero as $n \to \infty$. This shows that our estimate in Theorem 4.1 is essentially the best that one can hope for.

4.2. Convergence results

Next, we state our second result, which concerns the convergence of the gradient flow of the loss to the proximity of the global minimizer.

Theorem 4.2 (Convergence of the gradient flow)**.** *Consider*

the gradient flow with respect to the loss $L(A, 0)$ for $\Sigma^* = I$:

$$\frac{d}{dt}A(t) = -\nabla_A L(A(t), 0).$$

Then, for any $\xi \geq \frac{16d^{3/2}Ld4^L}{\sqrt{n}}$, after time $t \geq \left(\frac{1}{\xi}\right)^{(L-1)/L} \frac{4Ld^{(L-1)/(2L-1)}}{(L-1)^2}$ if $L > 1$ and $t \geq 8\ln(\frac{L(A_0,0)}{\xi})$ if $L = 1$, we have

1. $\mathcal{L}(A(t)) \leq \xi$,
2. $(1 - 8(1 + 4d^{1/(2L)})\xi^{1/(2L)})A^{opt} \preccurlyeq A \preccurlyeq (1 + 8(1 + 4d^{1/(2L)})\xi^{1/(2L)})A^{opt}$

Note that the landscape of the loss with respect to A is highly non-convex, hence Theorem 4.2 does not follow from the typical convex analysis of gradient flows. The key in obtaining this result is that we show a novel gradient dominance condition for the loss with power $(2L-1)/L$, which we state next.

Theorem 4.3 (Gradient dominance)**.** *Given $\Sigma^* = I$, for any A that $L(A, 0) \geq \frac{16d^{3/2}Ld4^L}{\sqrt{n}}$, we have the following gradient dominance condition:*

$$\|\nabla_A L(A, 0)\|^2 \geq \frac{L - \frac{1}{2}}{4d^{(L-1)/(2L-1)}} L(A, 0)^{(2L-1)/L}.$$

Remark. Theorem 4.3 illustrates that the squared norm of the gradient is at least proportional to the power $(2L-1)/L$ of the value of the loss. On the other hand, it is easy to see that the speed of change of the value of the loss on the gradient flow, namely $\frac{d}{dt}\mathcal{L}(A(t), 0)$, is equal to the squared norm of the gradient. But when the value of the loss is large, then the size of the gradients increase accordingly due to gradient dominance, therefore the convergence is faster when the loss is high. This trend is evident in the rigorous rate that we obtain on the convergence of the gradient flow in Theorem 4.1.

4.3. Out of distribution generalization

In the result below, we show that a looped Transformer learned on one in-context distribution can generalize to other problem instances with different covariance, owing to the fact that it has learned a good iterative algorithm.

Theorem 4.4. *Let A^{opt}, u^{opt} be the global minimizers of the poplulation loss for looped Transformer with depth L when the in-context input $\{x_i\}_{i=1}^n$ are sampled from $\mathcal{N}(0, \Sigma^*)$ and w^* is sampled from $\mathcal{N}(0, {\Sigma^*}^{-1})$. Suppose we are given an arbitrary linear regression instance $\mathcal{I}^{out} = \left\{x_i^{out}, y_i^{out}\right\}_{i=1}^n, w^{out,*}$ with input matrix $X^{out} = [x_1^{out}, \ldots, x_n^{out}]$, query vector x_q^{out}, and label $y_q^{out} = {w^{out,*}}^\top x_q^{out}$. Then, if for parameter $0 < \xi < 1$, the input covariance matrix $\Sigma^{out} = X^{out}{X^{out}}^\top$ of the out of distribution instance satisfies*

$$(1 - \xi)\Sigma^* \preccurlyeq \Sigma^{out} \preccurlyeq (1 + \xi)\Sigma^*, \tag{9}$$

we have the following instance-dependent bound on the out of distribution loss:

$$\begin{aligned}&(TF_L(Z_0^{out}; Q, P) - y_q^{out})^2 \\ &\leq (1 + 16\delta d^{1/(2L)})^2(16\delta d^{1/(2L)} + \xi)^{2L} \\ &\times \left\|x_q^{out}\right\|_{\Sigma^*}^2 \left\|w^{out,*}\right\|_{\Sigma^{*-1}}^2.\end{aligned}$$

5. Proof Ideas

In this section we present the high level intuitions and key steps in the proof of the results. The proof is structured as follows:

- First we obtain closed form formula for the loss function in Lemma 5.1 in terms of the parameter A and covariance Σ^*. The loss depends on how close $A^{1/2}\Sigma A^{1/2}$ is to identity for a randomly sampled Σ. Using the estimates in Lemma 5.3, we obtain an estimate on the loss based on the eigenvalues of the matrix $A^{1/2}\Sigma^* A^{1/2}$. Importantly, the result of Lemma 5.3 is based on estimating the higher moments of the Wishart matrix with arbitrary covariance, shown in Lemma 5.2 Using our estimate of the loss in Lemma 5.3, we obtain a precise characterization of the global optimum.
- We further use Lemma 5.3 to drive an estimate on the magnitude of the gradient based on the same eigenvalues, those of $A^{1/2}\Sigma^* A^{1/2}$. Comparing this with our estimate for the loss from Lemma 5.3, we obtain the gradient dominance condition in Theorem 4.3.
- We use the gradient dominance condition to estimate the speed of convergence of the gradient flow to the proximity of the global minimizer in Theorem 4.2.

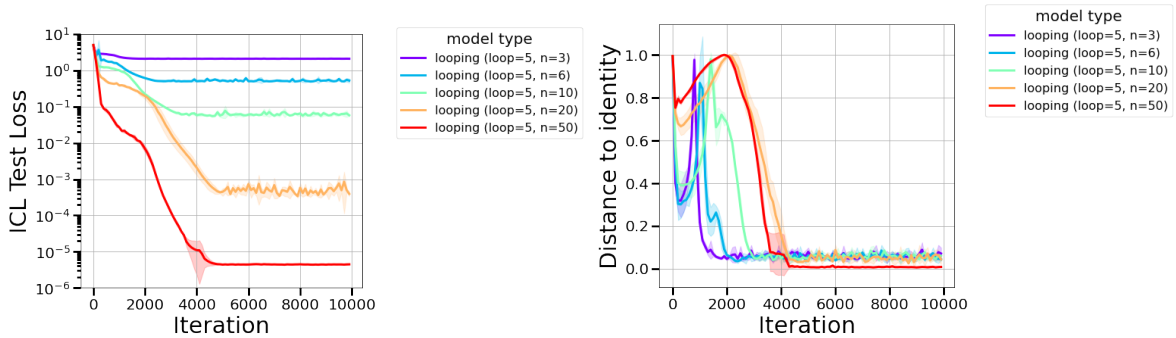
The starting point of the proof is that we can write the loss in a matrix power format based on A when u is set to zero:

Lemma 5.1. *Given $u = 0$, the loss for looped Transformer is as follows:*

$$\mathcal{L}(A, 0) = \mathbb{E}_X\left[tr((I - A^{1/2}\Sigma A^{1/2})^{2L})\right],$$

where $\Sigma = \frac{1}{n}\sum_{i=1}^n x_i x_i^\top$.

To be able to estimate the global minimizers of this loss, first we need to estimate its value. In particular, we hope to relate the value of the loss to the eigenvalues of



(a) Loss curve

(b) Distance to identity

Figure 1. Measuring the effect of number of samples n for looped models trained on inputs of dimension $d = 5$. Theorem 4.1 shows that the global optima for looped models is $A = I$ (since $\Sigma^* = I$ here) for large enough n. Here we verify that A converges to something very close to I even for smaller values of n and even $n < d$.

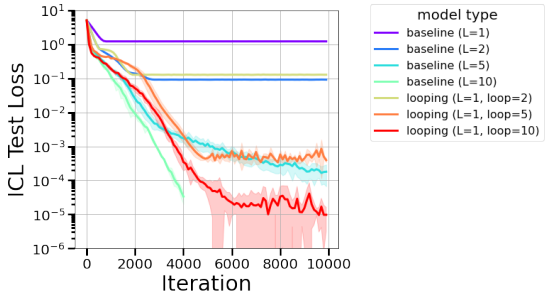


Figure 2. Loss trajectory as training proceeds for looped models and baseline multilayer models. Interestingly a 1-layer model looped L times performs similarly to an L layer multilayer model. Furthermore, increasing the number of loops leads to lower loss.

A. Note that if the data covariance matrix Σ was equal to the population covariance matrix Σ^*, then loss would turn into $\mathrm{tr}((I - A^{1/2}\Sigma^* A^{1/2})^{2L})$, whose global minimum is $A = \Sigma^{*-1}$. However, we can still hope to approximate the value of $\mathbb{E}_X\left[\mathrm{tr}((I - A^{1/2}\Sigma A^{1/2})^{2L})\right]$ with $\mathrm{tr}((I - A^{1/2}\Sigma^* A^{1/2})^{2L})$ given that we have a control on the expectation of the powers of the form $\mathbb{E}_X\left[(A^{1/2}\Sigma A^{1/2})^k\right]$ for $1 \le k \le 2L$. While there are some work on obtaining formulas for the moments of the Wishart matrix (note that ΣA is a Wishart matrix), these formulas (Bishop et al., 2018) are in the form of large summations and do not directly provide closed-form estimates in the general case. In general, the moment of the product of n Gaussian scalar variables can be written as a sum over various allocations of the variables into pairs, then multiplying the covariances of the pairs, due to Isserlis' Theorem. However, this gives a formula in terms of a large summation. Here, we propose a simple combinatorial argument in Lemma 5.2 which relates the moments of the Wishart matrix to the cycle structure of certain graphs related to the pairings of the Gaussian vectors, while using Isserlis' theorem. In particular, we show the following Lemma which relates the eigenvalues of the moments of the data covariance matrix and the covariance matrix itself:

Lemma 5.2 (Moment controls)**.** *Suppose* $\forall i \in [n], \tilde{x}_i \sim \mathcal{N}(0, \tilde{\Sigma})$. *Consider the eigen-decomposition* $\tilde{\Sigma} = \sum_{i=1}^d \lambda_i u_i u_i^\top$ *with eigenvalues* $\lambda_1 \ge \lambda_2 \ge \cdots \ge \lambda_d$. *Then, for all* $1 \le k \le 2L$, $\mathbb{E}\left[(\frac{1}{n}\sum_{i=1}^n \tilde{x}_i \tilde{x}_i^\top)^k\right]$ *can be written as*

$$\mathbb{E}\left[(\frac{1}{n}\sum_{i=1}^n \tilde{x}_i \tilde{x}_i^\top)^k\right] = \sum_{j=1}^d \alpha_{n,d,k}^{(j)} u_j u_j^\top,$$

where for all $1 \le j \le d$*:*

$$\lambda_j^k - \delta^k \lambda_1^k \le \alpha_{n,d,k}^{(j)} \le \lambda_j^k + \delta^k \lambda_1^k.$$

Next, we translate this Lemma to a control over the eigenvalues of $A^{1/2}\mathbb{E}\left[(I - \Sigma A)^k\right]A^{-1/2}$ with respect to that of $A^{1/2}\Sigma^* A^{1/2}$:

Lemma 5.3 (Eigenvalue approximation)**.** *Given the eigen-decomposition*

$$A^{1/2}\Sigma^* A^{1/2} = \sum_{i=1}^n \lambda_i u_i u_i^\top,$$

then for all $k \le 2L$*, the matrix* $\mathbb{E}\left[(I - A^{1/2}\Sigma A^{1/2})^k\right]$ *can be written as*

$$\mathbb{E}\left[(I - A^{1/2}\Sigma A^{1/2})^k\right] = \sum_{i=1}^n \beta_i^{(k)} u_i u_i^\top,$$

where

$$(1 - \lambda_i)^k - \delta^k(\lambda_1 + 1)^k \le \beta_i^{(k)} \le (1 - \lambda_i)^k + \delta^k(\lambda_1 + 1)^k.$$

We use Lemma 5.3 to argue that the matrix $(I - A^{1/2}\Sigma A^{1/2})^{2L}$ for data covariance matrix $\Sigma = \frac{1}{n}\sum_{i=1}^{n} x_i x_i^\top$ roughly behaves like $(I - A^{1/2}\Sigma^* A^{1/2})^{2L}$, plus some noise on each eigenvalue.

The rest of the proof in a high level goes as follows: Given that the value of the loss has certain amount of sub-optimality gap, we deduce, using Lemma 5.3, a lower bound on the distance of the eigenvalues of $A^{1/2}\Sigma A^{1/2}$ from one in $2L$-norm. We then again apply Lemma 5.3, this time for power $2L-1$, which is relevant from the algebraic form of the gradient $\nabla_A L(A, 0)$, to deduce a lower bound for norm of the gradient based on the distance of the eigenvalues of $A^{1/2}\Sigma A^{1/2}$ to one in the $4L-2$-norm. Finally by relating these two results using Holder's inequality, we obtain the gradient dominance for values of sub-optimality that are not too small.

Note that as in Lemma 5.3, the magnitude of the noise on all the eigenvalues is controlled by the largest eigenvalue, hence the noise is multiplicative only for the largest eigenvalue. This introduces additional difficulty in arguing about the distance of eigenvalues of A from one given a certain suboptimality gap. Next, using the gradient dominance condition, we estimate the gradient flow ODE and upper bound the value of the loss at a positive time $t > 0$ in Theorem 4.2. To finish the proof of Theorem 4.2, we need to translate a small suboptimality gap into closeness to global optimum, which we prove the following Theorem:

Theorem 5.4 (Small loss implies close to optimal)**.** *For $\epsilon > 4\delta$, if $L(A, 0) \leq \epsilon^{2L}/2$, then for $c = (4 + 16d^{1/(2L)})$*

$$(1 - c\epsilon)A^{opt} \preccurlyeq A \preccurlyeq (1 + c\epsilon)A^{opt}.$$

Note that δ goes to zero as $n \to \infty$, so A can get arbitrarily close to A^{opt} given large enough n.

6. Experiments

In this section we run experiments on in-context learning linear regression to validate the theoretical results and to go beyond them. In particular, we test if looped models can indeed be trained to convergence, as the theory suggests, and whether the learned solution is close to the predicted global minima. Furthermore, we investigate the effect of various factors such number of loops, number of in-context samples and depth of the model (in the multi-layer case). We use the codebase and experimental setup from (Ahn et al., 2023) for all our linear regression experiments. In particular we work with $d = 10$ dimensional inputs and train with L attention layer models for multilayer training and 1 layer attention model looped L times. Inputs and labels are sampled exactly based on the setup from Section 3.2, using covariance $\Sigma^* = I$.

6.1. Effect of loops

We first test whether training with looped model converges to a low loss, and how small the loss can be made with more loops. In Figure 2, we see that looped models indeed converge to very small loss very quickly, and higher loops leads to lower loss as expected. Interestingly, we find that a 1-layer model looped L times roughly has very loss to an L-layer non-looped model.

6.2. Effect of in-context samples

Theorem 4.2 shows convergence of the gradient flow for looped models when the number of in-context samples, n, is large compared to the dimension d. In these experiments we test the convergence of loss and iterate for smaller values of n, when n is closer to, or even smaller than d. In Figure 1 we observe that the loss converges for all values of $n > 1$ and the iterates also converge to a value very close identity. Theoretically proving this result remains an open question.

6.3. Out-of-distribution evaluation

While the looped model was trained with linear regression instances with identity covariance, we evaluate the trained looped model on out-of-distribution (OOD) data with a different covariance $\neq I$. The in-distribution covariance is $\Sigma^\star = I$ while the out of distribution covariance is $\Sigma = U\Sigma^2 U^T$ where U is a randomly sampled orthogonal matrix, and $\Sigma = [1, 1, 1/2, 1/4, 1]$. The input and classifier are sampled from the distribution described in Section 3.2.

Theorem 4.4 predicts that the model trained on identity covariance should also generalize to other covariances, because it simulates multi-step preconditioned gradient descent that works for all problems instances. In Figure 3, we find that the learned looped model achieves small loss for OOD data, although the scale of the loss is higher than in-distribution (ID) data. Interestingly, for looped models trained with just 2 (or 5) loops, evaluating them with arbitrarily large number of loops during test time continues to decrease the loss even further for ID and OOD data. This suggests that the trained looped models are indeed learning a good iterative algorithm.

7. Conclusion

This work provides the first convergence result showing that attention based models can *learn* to simulate multi-step gradient descent for in-context learning. The result not only demonstrates that Transformers can learn interpretable multi-step iterative algorithms (gradient descent in this case), but also highlights the importance of looped models in understanding such phenomena. There are several open questions in this space including understanding the

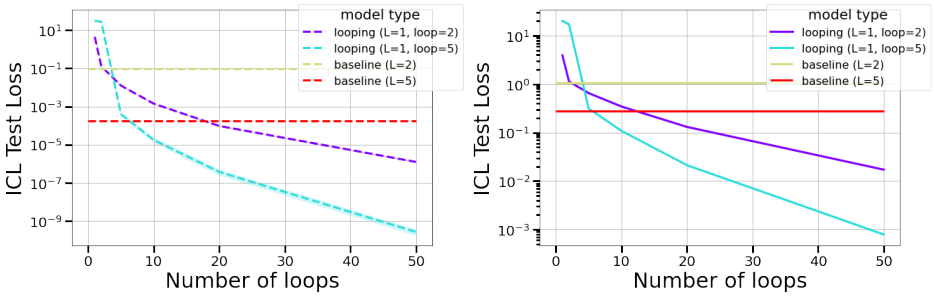


Figure 3. In-context linear regression loss on in-distribution (left) and out-of-distribution (right) data which is sampled using a different covariance $\Sigma \neq I$. For looped models trained with just few loops (2, or 5), evaluating with more loops keeps improving the loss in both cases, suggesting that it learned the correct iterative algorithm.

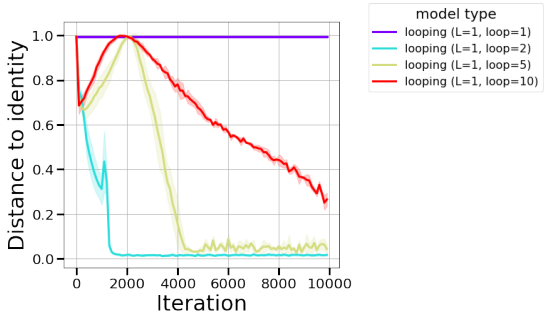


Figure 4. The iterate A converges to identity for all number of loops. The converge is slower for large number of loops which is also observed by our rate of convergence in Theorem 4.2. Interestingly just training with 1 loop does not converge in this setup.

landscape of the loss, convergence of training without weight sharing across layers, and handling of non-linearity in the attention layers. It is also interesting to understand the empirical phenomenon that looping the trained models beyond the number of loops used in training can continue to improve the test loss. One way to show this is by obtaining a tighter upper bound on the optimal loss value.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

Acknowledgement

This research was supported in part by Office of Naval Research grant N00014-20-1-2023 (MURI ML-SCOPE), NSF AI Institute TILOS (NSF CCF-2112665), NSF award 2134108, and the Alexander von Humboldt Foundation.

References

Ahn, K., Cheng, X., Daneshmand, H., and Sra, S. Transformers learn to implement preconditioned gradient descent for in-context learning. *arXiv preprint arXiv:2306.00297*, 2023.

Ahn, K., Cheng, X., Song, M., Yun, C., Jadbabaie, A., and Sra, S. Linear attention is (maybe) all you need (to understand transformer optimization). In *The Twelfth International Conference on Learning Representations*, 2024.

Akyürek, E., Schuurmans, D., Andreas, J., Ma, T., and Zhou, D. What learning algorithm is in-context learning? investigations with linear models. *arXiv preprint arXiv:2211.15661*, 2022.

Allen-Zhu, Z., Li, Y., and Liang, Y. Learning and generalization in overparameterized neural networks, going beyond two layers. *Advances in neural information processing systems*, 32, 2019.

Bai, Y., Chen, F., Wang, H., Xiong, C., and Mei, S. Transformers as statisticians: Provable in-context learning with in-context algorithm selection. *arXiv preprint arXiv:2306.04637*, 2023.

Bishop, A. N., Del Moral, P., Niclas, A., et al. An introduction to wishart matrix moments. *Foundations and Trends® in Machine Learning*, 11(2):97–218, 2018.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. In *Neural Information Processing Systems (NeurIPS)*, 2020.

Buhai, R.-D., Halpern, Y., Kim, Y., Risteski, A., and Sontag, D. Empirical study of the benefits of overparameterization in learning latent variable models. In *International Conference on Machine Learning*, pp. 1211–1219. PMLR, 2020.

Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, E., Wang, X., Dehghani, M., and Brahma, Siddhartha, e. a. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*, 2022.

Dasgupta, I., Lampinen, A. K., Chan, S. C., Creswell, A., Kumaran, D., McClelland, J. L., and Hill, F. Language models show human-like content effects on reasoning. *arXiv preprint arXiv:2207.07051*, 2022.

De Sa, C. M., Kale, S., Lee, J. D., Sekhari, A., and Sridharan, K. From gradient flow on population loss to learning with stochastic gradient descent. *Advances in Neural Information Processing Systems*, 35:30963–30976, 2022.

Fu, D., Chen, T.-Q., Jia, R., and Sharan, V. Transformers learn higher-order optimization methods for in-context learning: A study with linear models. *arXiv preprint arXiv:2310.17086*, 2023.

Garg, S., Tsipras, D., Liang, P. S., and Valiant, G. What can transformers learn in-context? a case study of simple function classes. *Advances in Neural Information Processing Systems*, 35:30583–30598, 2022.

Giannou, A., Rajput, S., Sohn, J.-y., Lee, K., Lee, J. D., and Papailiopoulos, D. Looped transformers as programmable computers. *arXiv preprint arXiv:2301.13196*, 2023.

He, K., Zhang, X., Ren, S., and Sun, J. Residual connections encourage iterative inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Lewkowycz, A., Andreassen, A., Dohan, D., Dyer, E., Michalewski, H., Ramasesh, V., Slone, A., Anil, C., Schlag, I., and Gutman-Solo, Theo, e. a. Solving quantitative reasoning problems with language models. *arXiv preprint arXiv:2206.14858*, 2022.

Lindner, D., Kramar, J., Farquhar, S., Rahtz, M., McGrath, T., and Mikulik, V. Tracr: Compiled transformers as a laboratory for interpretability. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=tbbId8u7nP.

Panigrahi, A., Malladi, S., Xia, M., and Arora, S. Trainable transformer in transformer. *arXiv preprint arXiv:2307.01189*, 2023.

Pérez, J., Marinković, J., and Barceló, P. On the turing completeness of modern neural network architectures. *arXiv preprint arXiv:1901.03429*, 2019.

Pérez, J., Barceló, P., and Marinkovic, J. Attention is turing-complete. *Journal of Machine Learning Research*, 22 (75):1–35, 2021.

Schlag, I., Irie, K., and Schmidhuber, J. Linear transformers are secretly fast weight programmers. In *International Conference on Machine Learning*, 2021.

Siegelmann, H. T. and Sontag, E. D. On the computational power of neural nets. In *Proceedings of Workshop on Computational Learning Theory*, 1994.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Łukasz Kaise, and Polosukhin, I. Attention is all you need. In *Neural Information Processing System*, 2017.

von Oswald, J., Niklasson, E., Schlegel, M., et al. Uncovering mesa-optimization algorithms in transformers, sep 2023. *URL http://arxiv. org/abs/2309.05858.→ p*, 9.

Von Oswald, J., Niklasson, E., Randazzo, E., Sacramento, J., Mordvintsev, A., Zhmoginov, A., and Vladymyrov, M. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, pp. 35151–35174. PMLR, 2023.

Wei, C., Chen, Y., and Ma, T. Statistically meaningful approximation: a case study on approximating turing machines with transformers. In *Advances on Neural Information Processing Systems (NeurIPS)*, 2022a.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 2022b.

Xie, S. M., Raghunathan, A., Liang, P., and Ma, T. An explanation of in-context learning as implicit bayesian inference. In *International Conference on Learning Representations*, 2022.

Yang, L., Lee, K., Nowak, R., and Papailiopoulos, D. Looped transformers are better at learning learning algorithms. *arXiv preprint arXiv:2311.12424*, 2023.

Zhang, R., Frei, S., and Bartlett, P. L. Trained transformers learn linear models in-context. *arXiv preprint arXiv:2306.09927*, 2023.

A. Gradient Dominance and Convergence of SGD in multilayer Transformers

A.1. A formula for the loss in the multilayer case

Theorem A.1. *Consider the linear attention layer with matrices P, Q set as in Equation (10) but with different parameters for different layers (i.e. without weight sharing). Namely, suppose for the layer t attention, we set*

$$Q^{(t)} := \begin{bmatrix} A^{(t)}_{d\times d} & 0 \\ 0 & 0 \end{bmatrix}, P^{(t)} := \begin{bmatrix} 0_{d\times d} & 0 \\ {u^{(t)}}^\top & 1 \end{bmatrix}. \tag{10}$$

Now defining

$$\left[Z^{(t)}\right]_{(d+1),1:n} = y^{(t)},$$
$$\left[Z^{(t)}\right]_{(d+1),(n+1)} = -y_q^{(t)},$$

we have the following recursions:

$${y^{(t)}}^\top = {w^*}^\top \prod_{i=0}^{t-1}(I-\Sigma A^{(i)})X + \sum_{i=0}^{t-1} {u^{(i)}}^\top \Sigma A^{(i)} \prod_{j=i+1}^{t-1}(I-\Sigma A^{(j)})X, \tag{11}$$

$${y_q^{(t)}}^\top = {y_q}^\top - {w^*}^\top \prod_{i=0}^{t-1}(I-\Sigma A^{(i)})x_q - \sum_{i=0}^{t-1} {u^{(i)}}^\top \Sigma A^{(i)} \prod_{j=i+1}^{t-1}(I-\Sigma A^{(j)})x_q, \tag{12}$$

with the convention that $\prod_0^{-1} = 1$ and $\sum_0^{-1} = 0$.

Proof. We show this by induction on t. For $t = 0$, note that ${w^*}^\top X = y^{(0)}$ and $y_q^{(0)} = 0 = y_q - {w^*}^\top x_q$. For the step of induction, suppose we have Equations (11) and (12) for $t-1$. Then, from the update rule

$$Z^{(t)} = Z^{(t-1)} - \frac{1}{n} P^{(t)} M Z^{(t)} A^{(t)} {Z^{(t)}}^\top,$$

we get

$$\begin{aligned}
{y^{(t+1)}}^\top &= {y^{(t)}}^\top - \frac{1}{n}{y^{(t)}}^\top X^\top A^{(t)} X \\
&= {w^*}^\top \prod_{i=0}^{t-1}(I-\Sigma A^{(i)})X - {w^*}^\top \prod_{i=0}^{t-1}(I-\Sigma A^{(i)})(XX^\top)A^{(i)}X \\
&+ \sum_{i=0}^{t-1} {u^{(i)}}^\top \Sigma A^{(i)} \prod_{j=i+1}^{t-1}(I-\Sigma A^{(j)})X - \sum_{i=0}^{t-1} {u^{(i)}}^\top \Sigma A^{(i)} \prod_{j=i+1}^{t-1}(I-\Sigma A^{(j)})(XX^\top)A^{(t)}X \\
&= {w^*}^\top \prod_{i=0}^{t}(I-\Sigma A^{(t)})X + \sum_{i=0}^{t-1} {u^{(i)}}^\top \Sigma A^{(i)} \prod_{j=i+1}^{t-1}(I-\Sigma A^{(j)})X,
\end{aligned}$$

where we used the fact that $XX^\top = \Sigma$. Moreover

$$\begin{aligned}
y_q^{(t+1)} &= y_q^{(t)} - \frac{1}{n}{y^{(t)}}^\top X^\top Q x_q \\
&= y_q - {w^*}^\top \prod_{i=0}^{t-1}(I-\Sigma A^{(i)})x_q - \sum_{i=0}^{t-1} {u^{(i)}}^\top \Sigma A^{(i)} \prod_{j=i+1}^{t-1}(I-\Sigma A^{(j)})x_q \\
&- {w^*}^\top \prod_{i=0}^{t-1}(I-\Sigma A^{(i)})(XX^\top)A^{(i)}x_q + \sum_{i=0}^{t-1} {u^{(i)}}^\top \Sigma A^{(i)} \prod_{j=i+1}^{t-1}(I-\Sigma A^{(j)})(XX^\top)A^{(j)}x_q \\
&= y_q - {w^*}^\top \prod_{i=0}^{t}(I-\Sigma A^{(i)})x_q - \sum_{i=0}^{t-1} {u^{(i)}}^\top \Sigma A^{(i)} \prod_{j=i+1}^{t}(I-\Sigma A^{(j)})x_q,
\end{aligned}$$

□

which completes the step of induction.

Lemma A.2 (moments of the Gaussian covariance)**.** *Given $n \geq 4k^2d^2$, for iid normal random vectors $x_1, \dots, x_n \sim \mathcal{N}(0, I)$ and data covariance $\Sigma = \frac{1}{n}\sum_{i=1}^n x_i {x_i}^\top$ we have*

$$\mathbb{E}\Big[(\frac{1}{n}\sum_{i=1}^n x_i {x_i}^\top)^k\Big] = \alpha_{n,k,d} I,$$

for

$$1 \leq \alpha_{n,d,k} \leq 1 + \frac{4kd}{\sqrt{n}}.$$

Proof. Note that for $2k$ (correlated) normal variables $u_1, \dots, u_{2k}$, from Isserlis' theorem we have

$$\mathbb{E}\Big[u_1 \dots u_{2k}\Big] = \sum_{p \in \mathcal{P}(2k)} \mathbb{E}\Big[u_{p_1^1} u_{p_2^1}\Big]\mathbb{E}\Big[u_{p_1^2} u_{p_2^2}\Big] \dots \mathbb{E}\Big[u_{p_1^k} u_{p_2^k}\Big],$$

where $\mathcal{P}(2k)$ is the set of allocations of $\{1, 2, \dots, 2k\}$ into unordered pairs $\Big((p_1^1, p_2^1), p^2 = (p_1^2, p_2^2), \dots, (p_1^k, p_2^k)\Big)$. For an array of random matrices $\Big(M^{(1)}, \dots, M^{(2k)}\Big)$ and allocation $p \in \mathcal{P}(2k)$, let $M(p) = (M(p)^{(1)}, \dots, M(p)^{(2k)})$ be the set of random matrices where for each pair $(p_1^i, p_2^i) \in p$, $M(p)_{p_1^i}$ and $M(p)_{p_2^i}$ have the same joint distribution as $M_{p_1^i}$ and $M_{p_2^i}$, while for $i \neq j$, $(M(p)_{p_1^i}, M(p)_{p_2^i})$ and $(M(p)_{p_1^j}, M(p)_{p_2^j})$ are indepdenent from each other. Equipped with this notation, we now apply Isserlis' theorem to each summand of the product of $2k$ matrices $M_1, \dots, M_{2k}$, which provides us with a similar expansion of the expectation of the product of matrices:

$$\mathbb{E}\Big[M^{(1)} \dots M^{(2k)}\Big] = \sum_{p \in \mathcal{P}(2k)} \mathbb{E}\Big[M(p)^{(1)} M(p)^{(2)} \dots M(p)^{(2k)}\Big]. \tag{13}$$

Now using Equation (13)

$$\mathbb{E}\Big[(\sum_{i=1}^n x_i x_i^\top)^k\Big] = \sum_{p \in \mathcal{P}(k)} \sum_{(\mathbf{i}_1, \dots, \mathbf{i}_k) \in [n]^k} \mathbb{E}\Big[x_{\mathbf{i}_1}^{(1)}(p) x_{\mathbf{i}_1}^{(1)}(p)^\top x_{\mathbf{i}_3}^{(2)}(p) x_{\mathbf{i}_4}^{(2)}(p)^\top \dots x_{\mathbf{i}_k}^{(k)}(p) x_{\mathbf{i}_k}^{(k)}(p)^\top\Big] \tag{14}$$

where $x^{(i)}(p)$ for an allocation $p \in \mathcal{P}(2k)$ is defined similarly to $M^{(i)}(p)$ above. Now consider the graph $\mathcal{G}_p$ with vertices $\{1, \dots, k\}$ where we put an edge between j and k if one of the indices $(\mathbf{i}_{2j-1}, \mathbf{i}_{2j})$ is paired with $(\mathbf{i}_{2k-1}, \mathbf{i}_{2k})$ according to p. A key idea that we use here is considering the cycle structure of $\mathcal{G}_p$. It is clear that each vertex has degree exactly two, hence it is decomposed into a number of cycles.

note that for the multi-indices $(\mathbf{i}_1, \dots, \mathbf{i}_{2k})$ in the sum (14) and a pair (p_1^j, p_2^j), if $\mathbf{i}_{p_1^j}$ and $\mathbf{i}_{p_2^j}$ are different, then the corresponding $x_{p_1^j}$ and $x_{p_2^j}$ are independent. Hence, the expectation is zero:

$$\mathbb{E}\Big[x_{\mathbf{i}_1}^{(1)}(p) x_{\mathbf{i}_1}^{(1)}(p)^\top x_{\mathbf{i}_2}^{(2)}(p) x_{\mathbf{i}_2}^{(2)}(p)^\top \dots x_{\mathbf{i}_k}^{(k)}(p) x_{\mathbf{i}_k}^{(k)}(p)^\top\Big] = 0.$$

therefore, for each pair $p^i = (p_1^i, p_2^i)$, $\mathbf{i}_{p_1^i} = \mathbf{i}_{p_2^i}$. This means that if there is an edge between j_1 and j_2 in $\mathcal{G}_p$, then $\mathbf{i}_{j_1} = \mathbf{i}_{j_2}$. Therefore, for a cycle $C = (j_1, \dots, j_r)$ in $\mathcal{G}_p$ we have $\mathbf{i}_{j_1} = \mathbf{i}_{j_2} = \dots = \mathbf{i}_{j_r}$. Note that we have exactly n choices for the value of these indices.

Hence, given a pairing p, the number of different ways of picking the multi-index $\mathbf{i}$ such that the expectation of its corresponding term is not zero is exactly

$$n^{C(\mathcal{G}_p)},$$

where $C(\mathcal{G}_p)$ is the number of cycles in $\mathcal{G}$. On the other hand, all of the non-zero terms have equal expectation. Therefore, we can write the expectation in Equation (14) as

$$\mathbb{E}\Big[(\sum_{i=1}^n x_i x_i^\top)^k\Big] = \sum_{p \in \mathcal{P}(k)} n^{C(\mathcal{G}_p)} \mathbb{E}\Big[x_1^{(1)}(p) x_1^{(1)}(p)^\top x_1^{(2)}(p) x_1^{(2)}(p)^\top \dots x_1^{(k)}(p) x_1^{(k)}(p)^\top\Big]. \tag{15}$$

Next, we make the observation that for taking expectation with respect to a pair, we can substitute both of the vectors in that pair by one of vectors in the standard basis, and sum the results. More rigorously, for a fixed matrix A, vector v, and scalar α we have

$$\begin{aligned}
\mathbb{E}\Big[x_1^\top A x_1\Big] &= \text{tr}(A) = \sum_i e_i^\top A e_i, \\
\mathbb{E}\Big[x_1 v^\top x_1\Big] &= v = \sum_i e_i v^\top e_i = v, \\
\mathbb{E}\Big[x_1^\top v x_1^\top\Big] &= v^\top = \sum_i e_i^\top v e_i^\top, \\
\mathbb{E}\Big[x_1 \alpha x_1^\top\Big] &= \alpha I = \sum_i e_i \alpha e_i^\top.
\end{aligned} \tag{16}$$

We can use this observation to unroll the expectation in Equation (13) as a sum. For example if j_1, j_2 are paired according to p, then

$$\begin{aligned}
&\mathbb{E}\Big[x_1^{(1)}(p)x_1^{(1)}(p)^\top x_1^{(2)}(p)x_1^{(2)}(p)^\top \dots x_1^{(k)}(p)x_1^{(k)}(p)^\top\Big] \\
&= \mathbb{E}\Big[x_1^{(1)}(p)\dots x_1^{(j_1)}(p)\Big(x_1^{(j_1)}(p)^\top x_1^{(j_1+1)}(p)\dots x_1^{(j_2-1)}(p)^\top\Big)x_1^{(j_2)}(p)\dots x_1^{(k)}(p)^\top\Big]. \\
&= \sum_{i=1}^n \mathbb{E}\Big[x_1^{(1)}(p)\dots e_i\Big(x_1^{(j_1)}(p)^\top x_1^{(j_1+2)}(p)\dots x_1^{(j_2-1)}(p)^\top\Big)e_i\dots x_1^{(k)}(p)^\top\Big].
\end{aligned}$$

Unrolling the expectation using Equations (16), we get

$$\mathbb{E}\Big[x_1^{(1)}(p)x_1^{(1)}(p)^\top x_1^{(2)}(p)x_1^{(2)}(p)^\top \dots x_1^{(k)}(p)x_1^{(k)}(p)^\top\Big] = \sum_{(\mathbf{i}_1,\dots,\mathbf{i}_{2k})\in[n]^{2k},\ \forall(j_1,j_2)\in p, \mathbf{i}_{j_1}=\mathbf{i}_{j_2}} e_{\mathbf{i}_1}e_{\mathbf{i}_2}^\top e_{\mathbf{i}_3}e_{\mathbf{i}_4}^\top \dots e_{\mathbf{i}_{2k-1}}e_{\mathbf{i}_{2k}}^\top. \tag{17}$$

The first observation above is that for consecutive elements $e_{\mathbf{i}_j}^\top e_{\mathbf{i}_{j+1}}$ we should have $\mathbf{i}_j = \mathbf{i}_{j+1}$ otherwise the product is zero. Hence, the sum above is really on multiindices of size k. Based on this observation, we consider a graph $\mathcal{G}_p'$ corresponding to the allocation p whose nodes are the pairs $(2,3),(4,5),\dots,(2k-2,2k-1),(1,2k)$, and we connect two nodes (j_1,j_2) and (i_1,i_2) in $\mathcal{G}_p'$ if either j_1 or j_2 is paired with i_1 or i_2 according to p. Then, similar to our argument for $\mathcal{G}$, for a cycle $(j_1,j_2,\dots,j_r)$ in $\mathcal{G}_p'$ in order for the term in Equation (17) to be non-zero, we should have $\mathbf{i}_{j_1} = \mathbf{i}_{j_2} = \dots = \mathbf{i}_{j_r}$. Therefore, the total number of choices for the multi-index $\mathbf{i}$ in Equation (17) is $n^{C(\mathcal{G}_p')}$, in which case the term $e_{\mathbf{i}_1}e_{\mathbf{i}_2}^\top e_{\mathbf{i}_3}e_{\mathbf{i}_4}^\top \dots e_{\mathbf{i}_{2k-1}}e_{\mathbf{i}_{2k}}^\top$ is equal to I. Therefore

$$\mathbb{E}\Big[x_1^{(1)}(p)x_1^{(1)}(p)^\top x_1^{(2)}(p)x_1^{(2)}(p)^\top \dots x_1^{(k)}(p)x_1^{(k)}(p)^\top\Big] = n^{C(\mathcal{G}_p')}I. \tag{18}$$

Combining Equations (18) and (15):

$$\mathbb{E}\Big[(\sum_{i=1}^n x_i x_i^\top)^k\Big] = \sum_{p\in\mathcal{P}(k)} n^{C(\mathcal{G}_p)} d^{C(\mathcal{G}_p')-1} I.$$

Now we need to estimate the number of cycles in the two graphs $\mathcal{G}_p$ and $\mathcal{G}_p'$ and how they interact with the choice of the allocation p. Note that each pair in the allocation p translates into an edge in $\mathcal{G}_p$ and $\mathcal{G}_p'$ and can be a self-loop (from a node to itself). Another point is that from the definition of $\mathcal{G}_p$ and $\mathcal{G}_p'$, every node has degree exactly two. The key idea that we use here is that the total number of loops in the two graphs is bounded by $k+1$. The reason is that each pair in p can be a self-loop in at most one of the graphs (this is true from the definition of the graphs), and each self-loop in one of the graph reduces the number of possible cycles in the other graph; suppose the number of self-loops in $\mathcal{G}_p$ is r. Then the rest of the $k-r$ nodes can at most divide into cycles of length two. This means $C(\mathcal{G}_p) \le r + \lfloor \frac{k-r}{2} \rfloor$. On the other hand, note that each self-loop in $\mathcal{G}_p$ is created by a pair $(2i-1,2i)$ in p, which is an edge between two consecutive nodes $((2i-2,2i-1),(2i,2i+1))$ in $\mathcal{G}_p'$. Such an edge reduces the number of connected components of G_p' by one. But as G_p'

mentioned, G'_p decomposes into a number of loops, hence the number of its connected components is equal to the number of its loops. Therefore, the number of loops in G'_p is at most $k-r$, i.e. $C(\mathcal{G}'_p) \leq k-r+1$.

Next, we upper bound the number of p's for which $\mathcal{G}_p$ has at least r self loops: We have at most $\binom{k}{r}$ number of choices for the self-loop nodes. Then, we are left with $k-r$ nodes, including $2(k-r)$ pairs of indices (according to the definition of $\mathcal{G}_p$). This means there are at most C_{k-r} choices for the rest of the graph, where C_{k-r} is the number of different ways that we can allocate $\{1, 2, \ldots, 2(k-r)\}$ into $(k-r)$ pairs. It is easy to see that $C_{k-r} = (2k-2r)!!$. Therefore, there are at most $\binom{k}{r}(2k-2r)!!$ choices of p which results in $\mathcal{G}_p$ with at least r self-loops. Putting everything together

$$
\begin{aligned}
n^k \alpha_{n,d,k} &\leq \sum_{p \in \mathcal{P}(k)} n^{r+\lfloor \frac{k-r}{2} \rfloor} d^{k-r} \\
&\leq \sum_{r=0}^{k} \binom{k}{r} (2k-2r)!! n^{\frac{k+r}{2}} d^s \\
&= \sum_{s=0}^{k} \binom{k}{s} (2s)!! n^{k-\frac{s}{2}} d^s \\
&\leq \sum_{s=0}^{k} \frac{k^s}{\sqrt{2\pi s}(s/e)^s} \sqrt{2s} (\frac{2s}{e})^s n^k (\frac{d}{\sqrt{n}})^s \\
&\leq \sum_{s=0}^{k} (2k)^s n^k (\frac{d}{\sqrt{n}})^s. \qquad (19)
\end{aligned}
$$

Now from the assumption $n \geq 4k^2 d^2$, we get

$$
n^k \alpha_{n,d,k} \leq n^k (1 + \frac{4kd}{\sqrt{n}}).
$$

The proof of the upper bound on $\alpha_{n,d,k}$ is complete. The lower bound simply follows from Jensen inequality. □

Lemma A.3. *The loss can be written as*

$$
L(A,u) = \mathbb{E}_X \Big[\Big\| \Sigma^{-1/2} \prod_{i=0}^{L-1} (I - \Sigma A^{(i)}) \Sigma^{1/2} \Big\|^2 \Big] + \mathbb{E}_X \Big[\Big\| \sum_{i=0}^{L-1} d_i^\top \Sigma A^{(i)} \prod_{j=i+1}^{L-1} (I - \Sigma A^{(j)}) \Sigma^{1/2} \Big\|^2 \Big].
$$

Proof. Note that

$$
\mathbb{E}\Big[(y_q - y_q^{(L)})^2\Big] = \mathbb{E}_{w^*, X, x_q} \Big[\Big(w^{*\top} \prod_{i=0}^{L-1} (I - \Sigma A^{(i)}) x_q + \sum_{i=0}^{L-1} d_i^\top \Sigma A^{(i)} \prod_{j=i+1}^{L-1} (I - \Sigma A^{(j)}) x_q \Big)^2 \Big].
$$

Now note that w^* is independent of $\sum_{i=0}^{L-1} d_i^\top \Sigma A^{(i)} \prod_{j=i+1}^{L-1} (I - \Sigma A^{(j)}) x_q$. Therefore, taking expectation with respect to w^*:

$$
\mathbb{E}\Big[(y_q - y_q^{(L)})^2\Big] = \mathbb{E}_{w^*, x_q, X} \Big[\Big(w^{*\top} \prod_{i=0}^{L-1} (I - \Sigma A^{(i)}) x_q \Big)^2 \Big] + \mathbb{E}_{x_q, X} \Big[\Big(\sum_{i=0}^{t-1} d_i^\top \Sigma A^{(i)} \prod_{j=i+1}^{L-1} (I - \Sigma A^{(j)}) x_q \Big)^2 \Big].
$$

Taking expectation with respect to x_q and w^*:

$$
\mathbb{E}\Big[(y_q - y_q^{(L)})^2\Big] = \mathbb{E}_X \Big[\Big\| \Sigma^{-1/2} \prod_{i=0}^{L-1} (I - \Sigma A^{(i)}) \Sigma^{1/2} \Big\|_F^2 \Big] + \mathbb{E}_X \Big[\Big\| \sum_{i=0}^{L-1} d_i^\top \Sigma A^{(i)} \prod_{j=i+1}^{L-1} (I - \Sigma A^{(j)}) \Sigma^{1/2} \Big\|^2 \Big]
$$

□

Corollary A.4. *The loss for loop Transformer is*

$$L(A,u) = \mathbb{E}_X\Big[tr((I - A^{1/2}\Sigma A^{1/2})^{2L})\Big] + \mathbb{E}_X\Big[\Big\|\sum_{i=0}^{L-1} u^\top \Sigma A(I-\Sigma A)^{L-1-i}\Sigma^{1/2}\Big\|^2\Big].$$

Proof. Just note that

$$\begin{aligned}
L(A,u) &= \mathbb{E}_X\Big[\text{tr}((I - \Sigma^{1/2}A\Sigma^{1/2})^{2L})\Big] + \mathbb{E}_X\Big[\Big\|\sum_{i=0}^{L-1} u^\top \Sigma A(I-\Sigma A)^{L-1-i}\Sigma^{1/2}\Big\|^2\Big] \\
&= \mathbb{E}_X\Big[\text{tr}((I - \Sigma^{1/2}A^{1/2}A^{1/2}\Sigma^{1/2})^{2L})\Big] + \mathbb{E}_X\Big[\Big\|\sum_{i=0}^{L-1} u^\top \Sigma A(I-\Sigma A)^{L-1-i}\Sigma^{1/2}\Big\|^2\Big] \\
&= \mathbb{E}_X\Big[\text{tr}((I - A^{1/2}\Sigma A^{1/2})^{2L})\Big] + \mathbb{E}_X\Big[\Big\|\sum_{i=0}^{L-1} u^\top \Sigma A(I-\Sigma A)^{L-1-i}\Sigma^{1/2}\Big\|^2\Big].
\end{aligned}$$

□

Lemma A.5 (Restatement of Lemma 5.2)**.** *Suppose* $\forall i \in [n], \tilde{x}_i \sim \mathcal{N}(0, \Sigma^*)$. *Consider the eigen-decomposition* $\Sigma^* = \sum_{i=1}^d \lambda_i u_i u_i^\top$ *with eigenvalues* $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d$. *Then,* $\mathbb{E}\Big[(\frac{1}{n}\sum_{i=1}^n \tilde{x}_i\tilde{x}_i^\top)^k\Big]$ *can be written as*

$$\mathbb{E}\Big[(\frac{1}{n}\sum_{i=1}^n \tilde{x}_i\tilde{x}_i^\top)^k\Big] = \sum_{j=1}^d \alpha_{n,d,k}^{(j)} u_j u_j^\top,$$

where for all $1 \leq j \leq d$*:*

$$\lambda_j^k - \delta\lambda_1^k \leq \alpha_{n,d,k}^{(j)} \leq \lambda_j^k + \delta\lambda_1^k.$$

Proof. In the proof of Lemma A.2, we used Equations (16) to simplify the loss and write it as a sum over the normal basis vectors e_i. It is easy to see that we have the equivalence of Equations (16) for $\tilde{x}_i \sim \mathcal{N}(0, \Sigma^*)$ when e_i's are replaced by $v_i = \sqrt{\lambda_i}u_i$.

$$\mathbb{E}\Big[(\sum_{i=1}^n x_i x_i^\top)^k\Big] = \sum_{p\in\mathcal{P}(k)} n^{C(\mathcal{G}_p)} \sum_{\mathbf{i}\in[d]^{C(\mathcal{G}'_p)}} \Big(\prod_{c\in C(\mathcal{G}'_p),(1,2k)\notin c} \lambda_{\mathbf{i}_c}^{|c|}\Big)\lambda_{\mathbf{i}_{c^*}}^{|c^*|} v_{\mathbf{i}_{c^*}} v_{\mathbf{i}_{c^*}}^\top,$$

where c^* is the loop in $\mathcal{G}'_p$ that includes the node consisting of the first and the last indices , i.e. $(1, 2k)$. But pushing the second sum to the product:

$$\begin{aligned}
\mathbb{E}\Big[(\sum_{i=1}^n x_i x_i^\top)^k\Big] &= \sum_{p\in\mathcal{P}(k)} n^{C(\mathcal{G}_p)}\Big(\prod_{c\in C(\mathcal{G}'_p),(1,2k)\notin c}\Big(\sum_{i=1}^d \lambda_i^{|c|}\Big)\Big)\Big(\sum_{i=1}^d \lambda_i^{|c^*|-1} v_{\mathbf{i}_{c^*}} v_{\mathbf{i}_{c^*}}^\top\Big) \\
&= \sum_{p\in\mathcal{P}(k)} n^{C(\mathcal{G}_p)}\Big(\prod_{c\in C(\mathcal{G}'_p),(1,2k)\notin c}\Big(\sum_{i=1}^d \lambda_i^{|c|}\Big)\Big)\Big(\Sigma^{*|c^*|}\Big).
\end{aligned}$$

Now if we upper bound all the eigenvalues λ_i in the sum $\Big(\sum_{i=1}^d \lambda_i^{|c|}\Big)$ above, we have similar to Equation (19) in the proof

of Lemma A.2.

$$\begin{aligned} n^k \alpha_{n,d,k}^{(j)} &= u_j^\top \mathbb{E}\Big[(\sum_{i=1}^n x_i x_i^\top)^k\Big] u_j \\ &\leq \sum_{p\in\mathcal{P}(k)} n^{C(\mathcal{G}_p)} \Big(\prod_{c\in C(\mathcal{G}_p'),(1,2k)\notin c} d\lambda_1^{|c|}\Big)\lambda_j^{|c^*|} \\ &\leq \sum_{p\in\mathcal{P}(k)} n^{C(\mathcal{G}_p)} d^{C(\mathcal{G}_p')-1}\lambda_j^k \\ &\leq n^k\lambda_j^k + n^k\lambda_1^k \frac{4kd}{\sqrt{n}}, \end{aligned}$$

and similarly

$$n^k \alpha_{n,d,k}^{(j)} \geq n^k \lambda_j^k (1 - \frac{4kd}{\sqrt{n}}).$$

□

Lemma A.6 (Changing the covariance matrix)**.** *We can write the first part of the loss* $\mathbb{E}_X\Big[tr((I - A^{1/2}\Sigma A^{1/2})^{2L})\Big]$ *as*

$$\mathbb{E}_X\Big[tr((I - A^{1/2}\Sigma A^{1/2})^{2L})\Big] = tr(\mathbb{E}_X\Big[(I-\tilde{\Sigma})^{2L}\Big]),$$

for

$$\tilde{\Sigma} = \frac{1}{n}\sum_{i=1}^n \tilde{x}_i\tilde{x}_i^\top,$$

where $\forall i, \tilde{x}_i \sim \mathcal{N}(0, A^{1/2}\Sigma^* A^{1/2})$.

Proof. Defining $\tilde{x}_i = A^{1/2}x_i$, then $\tilde{x}_i \sim \mathcal{N}(0, A^{1/2}\Sigma^* A^{1/2})$, and

$$\mathbb{E}_X\Big[(I - A^{1/2}\Sigma A^{1/2})^{2L}\Big] = \mathbb{E}_X\Big[(I-\tilde{\Sigma})^{2L}\Big],$$

for

$$\tilde{\Sigma} = \frac{1}{n}\sum_{i=1}^n \tilde{x}_i\tilde{x}_i^\top.$$

This finishes the proof. □

Lemma A.7 (Restatement of Lemma 5.3)**.** *Consider the following eigendecomposition:*

$$\mathbb{E}\Big[(A^{1/2}\Sigma A^{1/2} - I)^k\Big] = \sum_{i=1}^d \beta_i^{(k)} u_i u_i^\top.$$

Then for

$$\delta^k = \frac{4kd}{\sqrt{n}}$$

we have

$$(\lambda_i - 1)^k - \delta^k(\lambda_1+1)^k \leq \beta_i^{(k)} \leq (\lambda_i - 1)^k + \delta^k(\lambda_1+1)^k,$$

where λ_i *is the* i*th eigenvalue of* $A^{1/2}\Sigma^* A^{1/2}$*, where recall* Σ^* *is the covariance matrix of* x_i*'s.*

Proof. The idea is to open up the power of matrix and estimate each of the terms separately:

$$\mathbb{E}\Big[A^{-1/2}(A^{1/2}\Sigma A^{1/2}-I)^k A^{1/2}\Big]=\sum_{i=1}^{k}(-1)^i\binom{k}{i}\mathbb{E}\Big[A^{-1/2}(A^{1/2}\Sigma A^{1/2})^i A^{1/2}\Big]. \tag{20}$$

The proof Directly follows from Lemmas 5.2 and A.6. □

Next, we prove some properties for the optimal solution of the loss. Theorem A.8 is another version of Theorem 4.1.

Theorem A.8 (Characterization of the optimal solution)**.** *Suppose $\{A^{opt}, d^{opt}\}$ is a global minimizer for the $L(A,d)$. Then, under the condition $\delta d^{1/(2L)} < \frac{1}{2}$,*

1.

$$L(A^{opt}, d^{opt}) \le d(2\delta)^{2L}.$$

2.

$$\|A^{opt\,1/2}\Sigma^* A^{opt\,1/2} - I\| \le 4\delta d^{1/(2L)}, d^{opt} = 0. \tag{21}$$

3.

$$(1-8\delta d^{1/(2L)})\Sigma^{*-1} \preccurlyeq A^{opt} \preccurlyeq (1+8\delta d^{1/(2L)})\Sigma^{*-1}. \tag{22}$$

Proof. Recall the form of the loss from Corollary A.4. First, note that the second term, $\mathbb{E}_X\Big[\Big\|\sum_{i=0}^{L-1} u^\top \Sigma A(I-\Sigma A)^{L-1-i}\Big\|^2\Big]$ is always positive when $u \neq 0$ and is zero if $u=0$, where for arbitrary vector v we define $\|v^\top\| = \|v\|$. Therefore $u^{\text{opt}} = 0$. Next, we show the upper bound on the optimal loss:

$$L(A^{\text{opt}}, d^{\text{opt}}) = L(A^{\text{opt}}, 0) \le L(\Sigma^{*-1}, 0).$$

Recall $\beta_i^{(2L)}$ is the ith eigenvalue of $\mathbb{E}\Big[A^{-1/2}(A^{1/2}\Sigma A^{1/2}-I)^k A^{1/2}\Big]$. Note that from Lemma 5.3

$$-(2\delta)^{2L} \le \beta_i^{(2L)} \le (2\delta)^{2L}.$$

On the other hand, $L(A,u) = \sum_{i=1}^{d}\beta_i^{(2L)}$, which means $L(I,0) \le d(2\delta)^{2L}$. To show Equation (21), suppose $|\lambda_1 - 1| \ge 4\delta d^{1/(2L)}$. Then, using Lemma 5.3

$$\beta_i^{(k)} \ge (\lambda_i - 1)^k - \delta^k(\lambda_1+1)^k \ge (4\delta d^{1/(2L)})^{2L} - \delta^{2L}(1+4\delta d^{1/(2L)})^{2L} > d(2\delta)^{2L}.$$

where we used the inequality $\delta d^{1/(2L)} < \frac{1}{2}$. Finally this implies

$$\|A^{\text{opt}\,1/2}\Sigma^* A^{\text{opt}\,1/2} - I\| \le 4\delta d^{1/(2L)}, d^{\text{opt}} = 0, \tag{23}$$

which means (using $4\delta d^{1/(2L)} \le \frac{1}{2}$)

$$(1-8\delta d^{1/(2L)})\Sigma^{*-1} \preccurlyeq A^{\text{opt}} \preccurlyeq (1+8\delta d^{1/(2L)})\Sigma^{*-1}.$$

□

Lemma A.9 (Small loss implies close to optimal)**.** *For $\epsilon > 3\delta$ and parameter A suppose we have $L(A,0) \le \epsilon^{2L} - \delta^{2L}(\epsilon+2)^{2L}$. Then*

$$(1-(4\epsilon+16\delta d^{1/(2L)}))A^{opt} \preccurlyeq A \preccurlyeq (1+(4\epsilon+16\delta d^{1/(2L)}))A^{opt}.$$

Proof. Consider the eigen-decomposition

$$A^{1/2}\Sigma^* A^{1/2} = \sum_{i=1}^{d} \lambda_i u_i u_i^\top,$$

$$\mathbb{E}\Big[(I - A^{1/2}\Sigma A^{1/2})^{2L}\Big] = \sum_{i=1}^{d} \beta_i^{(2L)} u_i u_i^\top.$$

Then, using Lemma 5.1, we have

$$L(A, 0) = \sum_{i=1}^{d} \beta_i^{(2L)}.$$

First note that we should have $|\lambda_1 - 1| \leq \epsilon$. This is because from Lemma 5.3 we get

$$\epsilon^{2L} - \delta^{2L}(\epsilon + 2)^{2L} \geq L(A, 0) \geq \beta_1^{(2L)} \geq (\lambda_1 - 1)^{2L} - \delta^{2L}(\lambda_1 + 1)^{2L},$$

which implies

$$|\lambda_1 - 1| \leq \epsilon.$$

Then again using Lemma 5.3 this time for λ_i we should also have

$$\begin{aligned}\epsilon^{2L} - \delta^{2L}(\epsilon + 2)^{2L} \geq L(A, 0) \geq \beta_i^{(2L)} &\geq (\lambda_i - 1)^{2L} - \delta^{2L}(\lambda_1 + 1)^{2L} \\ &\geq (\lambda_i - 1)^{2L} - \delta^{2L}(\epsilon + 2)^{2L},\end{aligned}$$

which implies $|\lambda_i - 1| \leq \epsilon$. The last inequality follows from the fact that the function $f(a) = a^{2L} - \delta^{2L}(a + 2)^{2L}$ is increasing for $a \geq \epsilon$ based on the assumption $\epsilon \geq 3\delta$ (just by checking the derivative is positive). Hence, overall we showed

$$\|A^{1/2}\Sigma^* A^{1/2} - I\| \leq \epsilon,$$

which means

$$(1 - 2\epsilon)\Sigma^{*-1} \preccurlyeq A \preccurlyeq (1 + 2\epsilon)\Sigma^{*-1}.$$

Combining this with Equation (22) from Lemma 4.1,

$$(1 - (4\epsilon + 16\delta d^{1/(2L)}))A^{opt} \preccurlyeq A \preccurlyeq (1 + (4\epsilon + 16\delta d^{1/(2L)}))A^{opt}.$$

□

Theorem A.10 (Restatement of Theorem 5.4)**.** *For $\epsilon > 4\delta$, if $L(A, 0) \leq \epsilon^{2L}/2$, then*

$$(1 - (4 + 16d^{1/(2L)})\epsilon)A^{opt} \preccurlyeq A \preccurlyeq (1 + (4 + 16d^{1/(2L)})\epsilon)A^{opt}.$$

Proof. Directly from Lemma A.9. □

Lemma A.11. *Suppose $L(A) \geq \epsilon^{2L} + d\delta^{2L}(\epsilon + 2)^{2L}$. Then*

$$\sum_{i=1}^{d} (\lambda_i - 1)^{2L} \geq \epsilon^{2L}.$$

Proof. Suppose the claim is not true, i.e. $\sum_{i=1}^{d} (\lambda_i - 1)^{2L} < \epsilon^{2L}$. Then, $(\lambda_1 - 1)^{2L} \leq \epsilon^{2L}$ or $|\lambda_1 - 1| \leq \epsilon$. Then using Lemma 5.3:

$$L(A) \leq \sum_{i=1}^{d} \beta_i^{(2L)} \leq \sum_{i=1}^{d} (\lambda_i - 1)^{2L} + d\delta^{2L}(2 + \epsilon)^{2L} < \epsilon^{2L} + d\delta^{2L}(2 + \epsilon)^{2L},$$

which is a contradiction. Hence, the proof is complete. □

Lemma A.12 (Estimate on the gradient)**.** *Given $\Sigma^* = I$, the gradient of the loss can be estimated as*

$$(2L)\sum_{j=1}^{d}\big((\lambda_j-1)^{2L-1}-\delta^{2L}(\lambda_1+1)^{2L-1}\big)u_ju_j^\top \preccurlyeq \nabla_A L(A,0) \preccurlyeq (2L)\sum_{j=1}^{d}\big((\lambda_j-1)^{2L-1}+\delta^{2L}(\lambda_1+1)^{2L-1}\big)u_ju_j^\top.$$

Proof. Note that the loss can be written as

$$L(A,0)=\mathbb{E}_X\Big[\mathrm{tr}(I-\Sigma^{1/2}A\Sigma^{1/2})^{2L}\Big]=\mathbb{E}_X\Big[\mathrm{tr}(I-\Sigma A)^{2L}\Big].$$

Taking derivative

$$\nabla_A L(A,0)=-\sum_{i=0}^{2L-1}\mathbb{E}_X\Big[(I-\Sigma A)^i\Sigma(I-\Sigma A)^{2L-1-i}\Big]. \tag{24}$$

But note that

$$\mathbb{E}_X\Big[(I-\Sigma A)^i\Sigma(I-\Sigma A)^{2L-1-i}\Big]=\mathbb{E}_X\Big[A^{-1/2}(I-A^{1/2}\Sigma A^{1/2})^i(A^{1/2}\Sigma A^{1/2})A^{-1}(I-A^{1/2}\Sigma A^{1/2})^{2L-1-i}A^{1/2}\Big]$$

Expanding this term, we get terms of the form $\mathbb{E}_X\Big[A^{-1/2}(A^{1/2}\Sigma A^{1/2})^{\ell_1}A^{-1}(A^{1/2}\Sigma A^{1/2})^{\ell_2}A^{1/2}\Big]$ for $\ell_1\geq 1$:

$$\mathbb{E}_X\Big[(I-\Sigma A)^i\Sigma(I-\Sigma A)^{2L-1-i}\Big] \tag{25}$$

$$=\sum_{0\leq\ell_1\leq i}\sum_{0\leq\ell_2\leq 2L-1-i}\binom{2L-1-i}{\ell_1}\binom{i}{\ell_2}(-1)^{\ell_1+\ell_2}\mathbb{E}_X\Big[A^{-1/2}(A^{1/2}\Sigma A^{1/2})^{\ell_1+1}A^{-1}(A^{1/2}\Sigma A^{1/2})^{\ell_2}A^{1/2}\Big] \tag{26}$$

Now similar to the proof of Lemma 5.2, we calculate the expectation of each term of this form. The subtle point here is that even though there is the matrix A^{-1} in between the random matrices, it has shared eigenvalues as the covariance matrix of the gaussians, hence the computation goes through similarly expect that for the cycle $\tilde{c}$ in $\mathcal{G}_p'$ which includes the vertex $(2\ell_1, 2\ell_1+1)$ generates a $\lambda_i^{|\tilde{c}|-1}$ instead of $\lambda_i^{|\tilde{c}|}$. More rigorously, for $A=\sum\lambda_iu_iu_i^\top$ We can argue that

$$\mathbb{E}_X\Big[A^{-1/2}(A^{1/2}\Sigma A^{1/2})^{\ell_1+1}A^{-1}(A^{1/2}\Sigma A^{1/2})^{\ell_2}A^{1/2}\Big]=\sum_i\gamma_ju_ju_j^\top,$$

where for $k=\ell_1+\ell_2$, similar to the proof of Lemma A.2, for $\tilde{x}_i=A^{1/2}x_i$:

$$\begin{aligned}
n^k\gamma_j &= u_j^\top\mathbb{E}\Big[(\sum_{i=1}^n\tilde{x}_i\tilde{x}_i^\top)^{\ell_1+1}A^{-1}(\sum_{i=1}^n\tilde{x}_i\tilde{x}_i^\top)^{\ell_2}\Big]u_j\\
&\leq \sum_{p\in\mathcal{P}(k)} n^{C(\mathcal{G}_p)}\Big(\prod_{c\in C(\mathcal{G}_p'),(1,2k)\notin c} d\lambda_1^{|c|-1\{(2\ell_1,2\ell_1+1)\in c\}}\Big)\lambda_j^{|c^*|-1\{(2\ell_1,2\ell_1+1)\in c^*\}}\\
&\leq \sum_{p\in\mathcal{P}(k)} n^{C(\mathcal{G}_p)}d^{C(\mathcal{G}_p')-1}\Big(\prod_{c\in C(\mathcal{G}_p'),(1,2k)\notin c} \lambda_1^{|c|-1\{(2\ell_1,2\ell_1+1)\in c\}}\Big)\lambda_j^{|c^*|-1\{(2\ell_1,2\ell_1+1)\in c^*\}}\\
&\leq n^k\lambda_j^{k-1}+n^k\lambda_1^{k-1}\frac{4kd}{\sqrt{n}},
\end{aligned}$$

and similarly

$$n^k\gamma_j\geq n^k\lambda_j^{k-1}-n^k\lambda_1^{k-1}\frac{4kd}{\sqrt{n}},$$

which implies

$$\lambda_j^{k-1}-\delta^k{\lambda_1}^{k-1}\leq\gamma_j\leq\lambda_j^{k-1}+\delta^k{\lambda_1}^{k-1}.$$

Plugging this into Equation (26) we get

$$\mathbb{E}_X\left[(I-\Sigma A)^i\Sigma(I-\Sigma A)^{2L-1-i}\right] = \sum_{0\le\ell_1\le i}\sum_{0\le\ell_2\le 2L-1-i}\sum_j \binom{2L-1-i}{\ell_1}\binom{i}{\ell_2}((-1)^{\ell_1+\ell_2}\lambda_j^{\ell_1+\ell_2}+\delta^{2L}\lambda_1^{\ell_1+\ell_2})u_ju_j^\top$$
$$\preccurlyeq \sum_j\left((\lambda_j-1)^{2L-1}+\delta^{2L}(\lambda_1+1)^{2L-1}\right)u_ju_j^\top,$$

and similarly

$$\mathbb{E}_X\left[(I-\Sigma A)^i\Sigma(I-\Sigma A)^{2L-1-i}\right] \succcurlyeq \sum_j\left((\lambda_j-1)^{2L-1}-\delta^{2L}(\lambda_1+1)^{2L-1}\right)u_ju_j^\top.$$

Combining this with Equation (24) concludes the result. □

Theorem A.13 (Restatement of Theorem 4.3)**.** *Suppose* $4\delta d^{3/(4L)} \le \epsilon \le 1$ *and* $4d\delta^{4L-2} \le 1$. *Then if* $L(A,0) \ge \epsilon^{2L} + d\delta^{2L}(\epsilon+2)^{2L}$, *we have*

$$\|\nabla L(A,0)\|^2 \ge \frac{L-\frac{1}{2}}{d^{(L-1)/(2L-1)}}\epsilon^{4L-2}.$$

Proof. Using Lemma 5.3, we have $\nabla_A L(A,0) = \sum_i \gamma_i u_i u_i^\top$ such that

$$\begin{aligned}\|\nabla_A L(A,0)\|^2 &\ge (2L)\sum_{i=1}^d\left((\lambda_i-1)^{2L-1}-\delta^{2L}(\lambda_1+1)^{2L-1}\right)^2\\ &\ge (2L)\sum_{i=1}^d\left(\frac{1}{2}(\lambda_i-1)^{4L-2}-\delta^{4L}(\lambda_1+1)^{4L-2}\right)\\ &\ge L\Big(\sum_{i=1}^d(\lambda_i-1)^{4L-2}\Big)-2dL\delta^{4L}(\lambda_1+1)^{4L-2}. \qquad (27)\end{aligned}$$

Now using Lemma A.11

$$d\max_i(\lambda_i-1)^{2L} \ge \sum_{i=1}^{2L}(\lambda_i-1)^{2L} \ge \epsilon^{2L},$$

or

$$\max_i|\lambda_i-1| \ge \epsilon/d^{1/(2L)}.$$

But this implies

$$d\delta^{4L}(\lambda_1+1)^{4L-2} \le d\delta^{4L}(2+\max_i|\lambda_i-1|)^{4L-2} \le \frac{1}{4}(\max_i|\lambda_i-1|)^{4L-2}.$$

The last inequality holds as the assumption on ϵ implies

$$(1+\frac{2}{\max_i|\lambda_1-1|})^{4L-2} \le (1+\frac{2d^{1/(2L)}}{\epsilon})^{4L-2} \le (\frac{3d^{1/(2L)}}{\epsilon})^{4L} \le \frac{1}{2d\delta^{4L}}.$$

Plugging this into Equation (27):

$$\|\nabla_A L(A,0)\|^2 \ge (L-\frac{1}{2})\sum_i(\lambda_i-1)^{4L-2}.$$

Using Lemma A.11 and Holder, we get

$$\begin{aligned}\sum_{i=1}^{d}(\lambda_i-1)^{4L-2} &\geq \frac{1}{d^{1/(2L)-1/(4L-2)}}\Big(\sum_{i=1}^{d}(\lambda_i-1)^{2L}\Big)^{(2L-1)/L}\\ &= \frac{1}{d^{(L-1)/(2L-1)}}\Big(\sum_{i=1}^{d}(\lambda_i-1)^{2L}\Big)^{(2L-1)/L}\\ &\geq \frac{1}{d^{(L-1)/(2L-1)}}\Big(\epsilon^{2L}\Big)^{(2L-1)/L}\\ &= \frac{1}{d^{(L-1)/(2L-1)}}\epsilon^{4L-2},\end{aligned}$$

which completes the proof. □

Theorem A.14 (Restatement of Theorem 4.3)**.** *For any A that $L(A,0)\geq 2d^{3/2}(4\delta)^{2L}$, we have the following gradient dominance condition:*

$$\|\nabla_A L(A,0)\|^2 \geq \frac{L-\frac{1}{2}}{4d^{(L-1)/(2L-1)}}L(A,0)^{(2L-1)/L}.$$

Proof. For $\epsilon > 4\delta d^{3/(4L)}$ we have $\epsilon^{2L}\geq d\delta^{2L}(\epsilon+2)^{2L}$Therefore, according to Theorem 4.3, for $\epsilon > 4\delta d^{3/(4L)}$ if we have $L(A,0)\geq 2\epsilon^{2L}$, then

$$\|\nabla L(A,0))\|^2 \geq \frac{L-\frac{1}{2}}{d^{(L-1)/(2L-1)}}\epsilon^{4L-2}. \tag{28}$$

Therefore, for any A if we have $(L(A,0)/2)^{1/(2L)}\geq 4\delta d^{3/(4L)}$, then if we define $\epsilon=(L(A,0)/2)^{1/(2L)}$, we have $L(A,0)=2\epsilon^{2L}$, which then implies (from Equation (28))

$$\|\nabla L(A,0))\|^2 \geq \frac{L-1}{d^{(L-1)/(2L-1)}}(\frac{L(A,0)}{2})^{(2L-1)/L}.$$

Therefore, we showed that if $L(A,0)\geq 2d^{3/2}(4\delta)^{2L}$, then $\|\nabla_A L(A,0)\|^2\geq \frac{L-\frac{1}{2}}{4d^{(L-1)/(2L-1)}}L(A,0)^{(2L-1)/L}$.

□

Theorem A.15 (Restatement of Theorem 4.2)**.** *Consider the gradient flow with respect to the loss $L(A,0)$):*

$$\frac{d}{dt}A(t) = -\nabla_A L(A(t),0).$$

Then, for any $\xi\geq 2d^{3/2}(4\delta)^{2L}$, after time $t\geq\left(\frac{1}{\xi}\right)^{(L-1)/L}\frac{4Ld^{(L-1)/(2L-1)}}{(L-1)^2}$ if $L>1$ and $t\geq 8\ln(\frac{L(A_0,0)}{\xi})$ if $L=1$, we have

1. *$L(A(t))\leq\xi$,*
2. *$(1-8(1+4d^{1/(2L)})\xi^{1/(2L)})A^{opt}\preccurlyeq A$*
   $\preccurlyeq(1+8(1+4d^{1/(2L)})\xi^{1/(2L)})A^{opt}$

Proof. Let $f(t)=L(A(t),0)$. Then from Theorem 4.3, if $f(t)\geq\xi\geq 2d^{3/2}(4\delta)^{2L}$, then we have

$$\begin{aligned}f'(t) &= \langle\frac{d}{dt}A(t),\nabla_A L(A,0)\rangle\\ &\quad-\langle\nabla_A L(A,0),\nabla_A L(A,0)\rangle\\ &= -\|\nabla_A L(A,0)\|^2\\ &\leq -\frac{L-\frac{1}{2}}{4d^{(L-1)/(2L-1)}}f(t)^{(2L-1)/L}.\end{aligned}$$

Therefore, if we define the ODE

$$\begin{aligned} &g(0) = f(0), \\ &\forall t \geq 0, g'(t) = -\frac{L-\frac{1}{2}}{4d^{(L-1)/(2L-1)}} g(t)^{(2L-1)/L}, \end{aligned} \tag{29}$$

then we have

$$f(t) \leq g(t), \forall t \geq 0.$$

Solving the ODE (29) for $L > 1$

$$\begin{aligned} g(t) &= (\frac{4d^{(L-1)/(2L-1)}}{L-\frac{1}{2}} \frac{L}{L-1})^{L/(L-1)} \frac{1}{(t+c)^{L/(L-1)}} \\ &= (\frac{4Ld^{(L-1)/(2L-1)}}{(L-1)(L-\frac{1}{2})})^{L/(L-1)} \frac{1}{(t+c)^{L/(L-1)}}., \end{aligned}$$

for

$$c = \frac{4Ld^{(L-1)/(2L-1)}}{(L-1)(L-\frac{1}{2})f(0)^{(L-1)/L}} = \frac{4Ld^{(L-1)/(2L-1)}}{(L-1)(L-\frac{1}{2})L(A_0,0)^{(L-1)/L}}.$$

Therefore, we get the following upper bound on f:

$$f(t) \leq (\frac{4Ld^{(L-1)/(2L-1)}}{(L-1)(L-\frac{1}{2})})^{L/(L-1)} \frac{1}{(t+c)^{L/(L-1)}} \leq (\frac{4Ld^{(L-1)/(2L-1)}}{(L-1)(L-\frac{1}{2})})^{L/(L-1)} \frac{1}{t^{L/(L-1)}}.$$

Therefore, to guarantee $f(t) \leq \xi$ we pick $t \geq \left(\frac{1}{\xi}\right)^{(L-1)/L} \frac{4Ld^{(L-1)/(2L-1)}}{(L-1)(L-\frac{1}{2})}$. Next, we handle the case $L = 1$ separately. For $L = 1$, the ODE in Equation (29) becomes

$$g'(t) = -\frac{1}{8} g(t),$$

whose solution is

$$g(t) = g(0)e^{-t/8} = f(0)e^{-t/8} = L(A_0, 0)e^{-t/8}.$$

Therefore, for $L = 1$ we get

$$f(t) \leq L(A_0, 0)e^{-t/8},$$

so in order to guarantee $f(t) \leq \xi$ we pick $t \geq 8\ln(L(A_0, 0)/\xi)$. This proves the first argument. The second argument is a consequence of applying Lemma A.9 to $L(A(t)) \leq \xi$. □

Theorem A.16 (Restatement of Theorem 4.4)**.** *Let A^{opt}, u^{opt} be the global minimizers of the poplulation loss for looped Transformer with depth L when the in-context input $\{x_i\}_{i=1}^n$ are sampled from $\mathcal{N}(0, \Sigma^*)$ and w^* is sampled from $\mathcal{N}(0, \Sigma^{*-1})$. Suppose we are given an arbitrary linear regression instance $\mathcal{I}^{out} = \left\{x_i^{out}, y_i^{out}\right\}_{i=1}^n, w^{out,*}$ with input matrix $X^{out} = [x_1^{out}, \ldots, x_n^{out}]$, query vector x_q^{out}, and label $y_q^{out} = {w^{out,*}}^\top x_q^{out}$. Then, if for parameter $0 < \xi < 1$, the input covariance matrix $\Sigma^{out} = X^{out}{X^{out}}^\top$ of the out of distribution instance satisfies*

$$(1-\xi)\Sigma^* \preccurlyeq \Sigma^{out} \preccurlyeq (1+\xi)\Sigma^*, \tag{30}$$

we have the following instance-dependent bound on the out of distribution loss:

$$\begin{aligned} &(TF_L(Z_0^{out}; Q, P) - y_q^{out})^2 \\ &\leq (1 + 16\delta d^{1/(2L)})^2 (16\delta d^{1/(2L)} + \xi)^{2L} \\ &\times \left\|x_q^{out}\right\|_{\Sigma^*}^2 \left\|w^{out,*}\right\|_{\Sigma^{*-1}}^2. \end{aligned}$$

Proof. Note that from Lemma, the global optimum A^{opt}, u^{opt} satisfies $u^{opt} = 0$ and

$$(1 - 8\delta d^{1/(2L)})A^{\text{opt}^{-1}} \preccurlyeq \Sigma^* \preccurlyeq (1 + 8\delta d^{1/(2L)})A^{\text{opt}^{-1}}. \tag{31}$$

But combining this with Equation (30), we get

$$(1-\xi)(1 - 8\delta d^{1/(2L)})A^{\text{opt}^{-1}} \preccurlyeq \Sigma^{out} \preccurlyeq (1+\xi)(1 + 8\delta d^{1/(2L)})A^{\text{opt}^{-1}},$$

which implies

$$\begin{aligned} -(16\zeta\delta d^{1/(2L)} + \xi)I \leq -(8(1+\xi))\delta d^{1/(2L)} + \xi)I \leq \left(I - {A^{opt}}^{1/2}\Sigma^{out}{A^{opt}}^{1/2}\right) \\ \leq (8(1-\xi)\delta d^{1/(2L)} + \xi)I \\ \leq (16\delta d^{1/(2L)} + \xi)I. \end{aligned} \tag{32}$$

Furthermore, plugging in the formula of y_q^L for the out of distribution instance, we have using Equation (31)

$$\begin{aligned} &(\text{TF}_L(Z_0^{out}; Q, P) - y_q^{out})^2 \\ &= \left({w^{out,*}}^\top {A^{out}}^{-1/2}(I - {A^{opt}}^{1/2}\Sigma^{out}{A^{opt}}^{1/2})^L {A^{opt}}^{1/2} x_q^{out}\right)^2 \\ &\leq \left({w^{out,*}}^\top {A^{opt}}^{-1} w^{out,*}\right)\left\|I - {A^{opt}}^{1/2}\Sigma^{out}{A^{opt}}^{1/2}\right\|^{2L} \\ &\times \left({x_q^{out}}^\top A^{opt} x_q^{out}\right) \\ &\leq (1 + 16\delta d^{1/(2L)})^2\left({w^{out,*}}^\top {\Sigma^*}^{-1} w^{out,*}\right) \\ &\times \left\|I - {A^{opt}}^{1/2}\Sigma^{out}{A^{opt}}^{1/2}\right\|^{2L}\left({x_q^{out}}^\top \Sigma^* x_q^{out}\right). \end{aligned}$$

But note that from Equation (32)

$$\begin{aligned} &(\text{TF}_L(Z_0^{out}; Q, P) - y_q^{out})^2 \\ &\leq (1 + 16\delta d^{1/(2L)})^2(16\delta d^{1/(2L)} + \xi)^{2L} \\ &\times \left\|x_q^{out}\right\|_{\Sigma^*}^2 \left\|w^{out,*}\right\|_{{\Sigma^*}^{-1}}^2. \end{aligned}$$

□