Revisiting ML Training under Fully Homomorphic Encryption: Convergence Guarantees, Differential Privacy, and Efficient Algorithms

Yvonne Zhou [1] **Mingyu Liang** [1] **Ivan Brugere** [2] **Danial Dervovic** [2] **Yue Guo** [3 2] **Antigoni Polychroniadou** [3 2] **Min Wu** [1] **Dana Dachman-Soled** [1]

Abstract

We present the first theoretical convergence analysis of machine learning training under fully homomorphic encryption (FHE), combined with a differentially private (DP) training algorithm tailored to encrypted computation. Our approach improves computational efficiency over standard differentially private gradient descent (DP-GD) while achieving comparable utility. In particular, we prove convergence of approximate gradient descent using polynomial approximations of activation and loss functions, which are required for FHE compatibility. To preserve privacy in downstream tasks, we integrate differential privacy without relying on costly per-sample gradient clipping, enabling scalable encrypted learning. We also provide data-independent hyperparameter selection and theoretically grounded strategies for polynomial approximation which can be of independent interest. Together, these contributions advance the feasibility of efficient, private, and secure machine learning on sensitive data.

1. Introduction

As machine learning (ML) is increasingly applied to sensitive domains, protecting the privacy and security of training data has become critical, especially when resource-constrained clients outsource training to external servers. In such settings, privacy risks arise from: (1) a potentially semi-honest *server* that follows the protocol but attempts to infer information about the data; and (2) *end-users* who, through black-box or white-box access to a trained model, may extract information about the training data.

To address these threats, we combine *Fully Homomorphic Encryption (FHE)* and *Differential Privacy (DP)* to provide end-to-end privacy guarantees. FHE ensures that the server performs training entirely on encrypted data and never observes raw inputs, while DP limits information leakage to end-users after deployment, mitigating attacks such as model inversion (Fredrikson et al., 2015). Together, FHE and DP offer strong protection throughout the ML pipeline.

As illustrated in Figure 1, the workflow proceeds as follows. The client generates an FHE key pair and encrypts both its dataset and pre-sampled noise, which are then sent to the server. The server performs homomorphic DP-GD, adding the encrypted noise to the encrypted gradients at each iteration, and returns the final encrypted model to the client. A semi-honest server learns nothing about the client's data, while the client obtains DP guarantees against downstream users, even after decrypting and releasing the model. This framework naturally extends to a multi-client setting using cryptographic tools such as multi-key FHE (Ananth et al., 2020; López-Alt et al., 2012; Mukherjee and Wichs, 2016), where each client encrypts its data and noise under its own public key.[1] The protocol requires only a single round of interaction with the server, and a single round of interaction among the clients to jointly decrypt the final model.

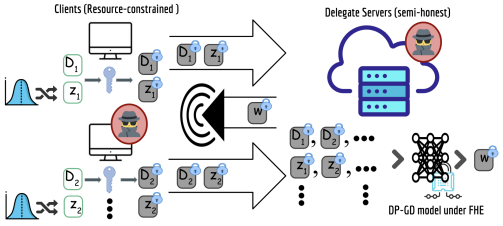


Figure 1. Secure and DP training with encrypted dataset

When performing gradient descent under FHE, the true gradient is often replaced with a polynomial approximation so that the arithmetic operations required to compute the gradient are compatible with the available FHE operations of "+" and "×" over the native FHE ring. To illustrate

[1]University of Maryland, College Park, MD 20742 [2]J.P. Morgan AI Research, New York, NY, 10017 [3]AlgoCRYPT CoE . Correspondence to: Yvonne Zhou <skyzhou@umd.edu>.

Proceedings of the 43^{rd} International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026.

[1]Each client independently contributes noise, so the variance grows linearly with the number of clients. Alternatively, the server could sample noise homomorphically under FHE, but this is significantly more computationally expensive.

our methodology, we concretely examine how polynomial approximation is employed in the case of neural networks.

Polynomial approximation for neural networks. Given a dataset D of N samples $(\mathbf{x}, y)$, consider the objective $f(\mathbf{w}) = \frac{1}{N}\sum_{(\mathbf{x},y)\in D} f_{(\mathbf{x},y)}(\mathbf{w})$, where $f_{(\mathbf{x},y)}(\mathbf{w}) = L_y(M_{\mathbf{w}}(\mathbf{x}))$, $M_{\mathbf{w}}$ is an L-layer feedforward neural network with activation ψ, L_y is the loss for label y, and $\nabla f(\mathbf{w})$ denotes the corresponding gradient.

A polynomial approximation to $\nabla f(\mathbf{w})$ is obtained by rerunning backpropagation while replacing each occurrence of ψ and its derivative ψ' (except in the final layer) with polynomial approximations p and p'. In the final layer, the term $\psi'(z_L)\, L_y'(a_L) = (L_y \circ \psi)'(z_L)$, where z_L and a_L denote the input and output of the final activation, is replaced by a polynomial approximation $\tilde{p}_y'$ of the combined derivative. Any continuous function can be approximated arbitrarily well by a polynomial of sufficient degree (Bernstein, 1912; Roulier, 1970). However, such approximations guarantee bounded error only on specified *closed intervals*, which prior work typically selected heuristically. A central goal of this work is to identify approximation intervals together with formal guarantees that all inputs encountered during gradient descent remain within them.

Crucially, the substitution described above yields a vector-valued function that is itself the gradient of another objective g (see Section 2.4). Thus, running approximate gradient descent using the substituted gradient is equivalent to running exact gradient descent on g.

Convergence of Approximate Gradient Descent. Although approximate gradient descent has performed well in practice (Kim et al., 2018; Han et al., 2018), its theoretical justification remains unclear. In particular, even if the gradients of the original and approximate objective functions, ∇f and ∇g, are close in norm (i.e., within ζ), this does not guarantee that essential properties of f are preserved by g. For instance, g may no longer be convex, even if f is, which implies that gradient descent on g may converge to a local minimum far from the global minimum of f. Thus, it is not immediately evident what can be said about the convergence of gradient descent to the minimum of f when optimization is performed with respect to the surrogate objective g.

Privacy-Preserving Learning via FHE and DP *Fully Homomorphic Encryption* (FHE) is an advanced cryptographic primitive that enables computations to be performed directly on encrypted data, producing encrypted outputs that, upon decryption, match the results of the corresponding computations carried out on the plaintext data. FHE enables a client to outsource the training process to an untrusted server without interaction: the client sends encrypted data to the server, which computes directly on ciphertexts and returns the final encrypted model weights. While FHE ensures the server learns nothing about the training data during the computation, it does not prevent information leakage *after* training. Once the client decrypts the final model, it may be used in downstream tasks or released publicly, potentially exposing sensitive information about the training data.

One might consider releasing the encrypted model together with the decryption key to trusted users. However, this reduces the setting to a closed-access deployment model and is incompatible with our goal of a *public-release* setting for broad use. To address this limitation, we incorporate *differential privacy* (DP), which provides formal guarantees that the released model does not overly depend on any single training example, and remains secure under arbitrary post-processing, without requiring trusted access or cryptographic keys. A widely used approach for achieving differential privacy in ML training is *differentially private gradient descent* (DP-GD), where *gradient perturbation* is performed–i.e. Gaussian noise is added to the gradient at each iteration of the optimization process (see Algorithm 2).

In our framework, the client encrypts the training data, and the server applies an *approximate* version of DP-GD, in which gradients of the original loss function f, are replaced with gradients of a polynomial approximation g, as discussed above. We first investigate the convergence guarantees of this noisy, approximate training procedure.

DP-GD relies on an additional step, *gradient clipping*, which limits the gradient's sensitivity by scaling it whenever its norm exceeds a chosen threshold. This operation controls the noise required to achieve DP. Performing gradient clipping *homomorphically* under FHE is highly inefficient [2]. It entails evaluating operations such as comparisons, square roots, and divisions directly on ciphertexts, each of which is costly due to bit-level computation and ciphertext expansion. Even after replacing these operations with polynomial approximations, the resulting circuit depth (the primary measure of FHE complexity) is about 24 per iteration–far larger than the depth of roughly 8 needed to implement an approximate version of standard (no-DP) gradient descent.

To overcome this limitation, we design an FHE-tailored training algorithm that avoids gradient clipping while providing formal DP guarantees and comparable model utility, significantly reducing computational overhead and improving scalability.

Gradient Perturbation vs. Output Perturbation. An alternative to gradient perturbation (DP-GD) is *output perturbation* (Output-GD), which enforces DP by adding calibrated noise to the final model parameters (Dwork et al.,

[2]The more advanced *adaptive gradient clipping* (Andrew et al., 2021) is even more expensive under FHE due to the additional logic required for gradient monitoring and threshold updates.

2006; Chaudhuri et al., 2011). Both prior work (Yu et al., 2019) and our experiments (Figure 2 and Table 4) show that Output-GD yields significantly worse utility under the same privacy budget, since it injects a single large noise term rather than distributing noise across iterations, which enables better composition, noise averaging, and stability. Further, the accuracy of Output-GD models exhibit high variance across runs, while gradient-perturbed models achieve consistent performance. Finally, Output-GD's sensitivity, and thus its noise scale, depends on the maximum weight magnitude attained during training. Under FHE, this would require either an expensive encrypted **Maximum** computation or sending the encrypted squared norms of all intermediate weights to the client for post-processing, incurring substantial communication overhead. and retaining privacy guarantees only in the single-client setting. For these reasons, we adopt gradient perturbation to achieve strong DP guarantees with higher practical utility.

1.1. Related Work

Secure gradient descent training using FHE Prior work has explored FHE for secure machine learning. (Gilad-Bachrach et al., 2016) study encrypted neural network inference using FHE. (Kim et al., 2018) implement logistic regression under the BGV scheme, (Han et al., 2018) propose an efficient logistic regression method with CKKS, and (Mihara et al., 2020) implement neural network training using CKKS. These studies focus on empirical evaluations and lack theoretical guarantees.

FHE + DP in federated learning Recent work combines homomorphic encryption (HE) and differential privacy (DP) for secure machine learning (Phong et al., 2018; Aziz et al., 2023; Sébert et al., 2023), typically applying HE to model updates (e.g., gradients) and requiring interactive client–server rounds. In contrast, our method encrypts the training data itself, enabling secure outsourcing to untrusted servers, supporting clients with limited resources. This allows the server to train large-scale models entirely without client participation, while preserving data privacy.

Outsourcing DP computation There is growing interest in outsourcing differential privacy (DP) computation. For example, (Dagher et al., 2020) propose a framework for confidential range-count queries using attribute-based encryption and DP, while (Ye et al., 2020) combine order-preserving encryption with DP to support secure outsourced data release. However, these works focus on DP data querying rather than machine learning tasks.

1.2. Our Contributions

To the best of our knowledge, this work presents the first theoretical convergence analysis of machine learning training within the context of FHE, along with a secure, DP training algorithm tailored for FHE. Our algorithm offers significantly greater efficiency than standard DP-GD under FHE, while maintaining comparable utility. Our key contributions are as follows:

Theoretical Convergence of Approximate Gradient Descent in FHE: While prior work has primarily focused on implementation-level designs for approximate gradient descent trainning under FHE, we provide the first formal convergence analysis of such training, where activation and loss functions are replaced by polynomial approximations—a necessary adaptation for compatibility with FHE. (Sec. 3)

Differential Privacy for Outsourced Training: Unlike existing FHE-based methods that focus solely on securely outsourcing standard training algorithms, we propose integrating differential privacy to mitigate information leakage risks from end users. Additionally, we offer the first comprehensive convergence analysis of DP training under approximate gradient descent. (Sec. 4)

Technical Innovations for DP-ML in FHE: We propose a method for provably DP gradient descent under FHE. We modify the objective function to allow strictly upper-bounding the norm of the iterates $\|\mathbf{w}_i\|$, which fixes the input range for polynomial approximations and ensures their approximation error is controlled. This allows us to bound the gradient sensitivity at each iteration and calibrate the noise correctly, while avoiding the costly gradient-clipping step, which is particularly expensive under FHE.

Crucially, naively replacing DP-GD components with polynomial approximations, regardless of whether clipping is used, can break DP guarantees, because the gradient sensitivity cannot be rigorously bounded. Polynomial approximations are only accurate over a prescribed input interval and can exhibit unbounded error outside it. Clipping does not resolve this issue, as the clipping operations themselves rely on polynomial approximations that can diverge beyond the interval. Prior work selected these intervals heuristically, with no guarantee that all inputs remain within bounds. In contrast, our approach provides a rigorous DP guarantee without heuristic assumptions (Sec. 5).

Our approach enables scalable DP training under FHE with markedly reduced computational cost, formal privacy guarantees, and comparable utility. Relative to a standard (unapproximated) DP-GD baseline trained on plaintext data, our FHE-trained models incur at most a 1.82% drop in accuracy and a 0.78% drop in AUC, while reducing per-iteration multiplicative depth by over $2.5\times$. In fact, despite the fact that we use a modified objective function, our training time is essentially the same as that of Output-GD, which runs standard (no-DP) GD under FHE (Sec. 7, Table 1 and 4).

Polynomial Approximation Strategies: Building on our convergence analysis, we derive practical guidelines for se-

lecting polynomial approximations for activation and loss functions. We validate our findings through extensive experiments. In our experiments, we observe that incorrectly setting the approximation intervals can lead to divergence in both non-DP and DP settings. Our techniques provide rigorous guarantees for these intervals, whereas prior algorithms require data-dependent tuning. We further illustrate the theoretical and empirical trade-offs across different polynomial approximation strategies. (Sec. 6, Appendix D)

Data-Independent Hyperparameter Selection: Our analysis enables the principled selection of key hyperparameters (e.g.,learning rate), without requiring access to the underlying dataset. Prior work did not address how to set these parameters in practice in a data-independent manner while guaranteeing convergence. (Sec. 6, Appendix D)

2. Notation and Background

We use boldface to represent vectors. $\|\mathbf{w}\|$ denotes the ℓ_2 norm of a vector $\mathbf{w}$. $\mathbf{I}_m$ denotes the identity matrix of dimension m. $[t]$ denotes the set of natural numbers less than or equal to t. $\log$ denotes natural log. A database D is a multiset of records $(\mathbf{x}, y)$, where $\mathbf{x} \in [-1, 1]^m$ and $y \in \{0, 1\}$. N denotes its size.

2.1. Properties of Functions

Definition 2.1 (β-smooth)**.** A function f is β-smooth, if $\langle \nabla f(\mathbf{w}) - \nabla f(\mathbf{v}), \mathbf{w} - \mathbf{v} \rangle \leq \beta \|\mathbf{w} - \mathbf{v}\|^2$

Definition 2.2 (μ-strong convexity)**.** A function f is μ-strongly convex, if $\langle \nabla f(\mathbf{w}) - \nabla f(\mathbf{v}), \mathbf{w} - \mathbf{v} \rangle \geq \mu \|\mathbf{w} - \mathbf{v}\|^2$

Definition 2.3 (Expected Curvature ν (Yu et al., 2019))**.** This notion yields parameters more closely aligned with experimental behavior. f achieves ν_{σ^2}-expected curvature around the optimum $\mathbf{w}^*$ with variance σ^2 if for all $\mathbf{w}$,

$$\mathbb{E}[\langle \nabla f(\mathbf{w} + \chi), \mathbf{w} + \chi - \mathbf{w}^* \rangle] \geq \nu_{\sigma^2} \mathbb{E}[\|\mathbf{w} + \chi - \mathbf{w}^*\|^2],$$

where the expectation is with respect to $\chi \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_m)$.

2.2. Differentially-Private Gradient Descent

Definition 2.4 ((ϵ, δ)-Differential Privacy)**.** A randomized mechanism $\mathcal{M} : \mathcal{D} \to \mathcal{R}$ satisfies (ϵ, δ)-differential privacy if for any two adjacent databases $D, D' \in \mathcal{D}$ (i.e. databases that differ in exactly one record) and for any subset of outputs $\mathcal{S} \subseteq \mathcal{R}$ it holds that $Pr[\mathcal{M}(D) \in \mathcal{S}] \leq e^{\epsilon} Pr[\mathcal{M}(D') \in \mathcal{S}] + \delta$.

DP learning is commonly implemented via Differentially Private Gradient Descent (DP-GD) and its stochastic variants (DP-SGD) (Abadi et al., 2016; Song et al., 2013; Bassily et al., 2014). Given the objective $f(\mathbf{w}) = \frac{1}{N} \sum_{(\mathbf{x},y) \in D} f_{(\mathbf{x},y)}(\mathbf{w})$, the per-sample gradient norm is $\|\nabla f_{(\mathbf{x},y)}(\mathbf{w})\|$. Since this quantity may be large or difficult to bound, DP-GD clips each per-sample gradient to a fixed threshold C, which determines the noise scale σ required for DP. Clipping is defined as

$$\mathsf{Cp}(\nabla f_{(\mathbf{x},y)}(\mathbf{w}), C) = \min\left\{1, \frac{C}{\|\nabla f_{(\mathbf{x},y)}(\mathbf{w})\|}\right\} \nabla f_{(\mathbf{x},y)}(\mathbf{w}).$$

Thus, in DP-GD, noise $\boldsymbol{\chi}_i \sim \mathcal{N}(0, \sigma^2 \mathbf{I}_m)$ is sampled at each step and model updates become: $\mathbf{w}_{i+1} = \mathbf{w}_i - \eta \left(\frac{1}{N} \sum_{(\mathbf{x},y) \in D} \mathsf{Cp}(\nabla f_{(\mathbf{x},y)}(\mathbf{w}_i), C) + \boldsymbol{\chi}_i \right)$.

Breakdown of DP guarantees under polynomial approximation. Even when an upper bound on $\|\nabla f_{(\mathbf{x},y)}(\mathbf{w})\|$ is available, no such guarantee holds for the polynomially approximated gradient $\|\nabla g_{(\mathbf{x},y)}(\mathbf{w})\|$. Polynomial approximations have bounded error only over fixed input intervals and may behave arbitrarily outside them. Prior to this work, there were no formal guarantees that the inputs encountered during gradient descent remained within these intervals.

2.3. Secure Machine Learning via FHE

Fully homomorphic encryption (FHE) (Gentry, 2009; Brakerski et al., 2011; Fan and Vercauteren, 2012; Ducas and Micciancio, 2014; Cheon et al., 2017) allows computations to be performed on encrypted data. Given encryptions $\mathsf{Enc}_{\mathsf{pk}}(m_1)$ and $\mathsf{Enc}_{\mathsf{pk}}(m_2)$, FHE allows computation of $\mathsf{Enc}_{\mathsf{pk}}(m_1 + m_2)$ and $\mathsf{Enc}_{\mathsf{pk}}(m_1 \cdot m_2)$ and, by composition, arbitrary arithmetic circuits, all *without decryption*. FHE-based training is computationally expensive, with cost dominated by the circuit's multiplicative depth.

2.4. Polynomial approximation of Neural Networks

FHE supports addition and multiplication over a native ring, allowing the computation of polynomials. However, the true gradient ∇f typically contains non-polynomial components that cannot be evaluated directly under FHE. Consequently, training under FHE replaces the true gradient with a polynomial approximation.

The approximate gradient is itself a gradient. Recall that for neural net training, objective function f has the form $f(\mathbf{w}) = \frac{1}{N} \sum_{(\mathbf{x},y) \in D} f_{(\mathbf{x},y)}(\mathbf{w})$, where $f_{(\mathbf{x},y)}(\mathbf{w}) = L_y(\psi^L(\mathbf{W}^L \cdots \boldsymbol{\psi}(\mathbf{W}^1 \mathbf{x})) \cdots))$, $\mathbf{w} = \mathbf{W}^1, \ldots, \mathbf{W}^L$ and matrices $\mathbf{W}^\ell$ are the weights in layer ℓ, and each $\boldsymbol{\psi}$ is a vector of activations, ψ, of appropriate dimension, applied in the hidden layers, and L_y is a loss function that depends on the label y. Then

$$\nabla f(\mathbf{W}^\ell) = \frac{1}{N} \sum_{(\mathbf{x},y) \in D} [\boldsymbol{\psi}'(\mathbf{z}_\ell) \odot (\mathbf{W}^{\ell+1})^T \cdot \boldsymbol{\psi}'(\mathbf{z}_{\ell+1}) \odot \cdots \odot (\mathbf{W}^L)^T \cdot \psi'(z_L) \cdot L_y'(a_L)] \mathbf{a}_{\ell-1}^T,$$

where for $j \in [L-1]$, $\mathbf{z}_j$ (resp. $\mathbf{a}_j$) are the inputs (resp. activations) of layer j, z_L (resp. a_L) is the input (resp. ac-

Algorithm 1 Approximate Gradient Descent.

Input: Running steps T, learning rate η_1.
Initialize $\mathbf{w}_{g,0} = \mathbf{0}$.
for $i = 0$ to $T-1$ **do**
  $\mathbf{w}_{g,i+1} \leftarrow \mathbf{w}_{g,i} - \eta_1 \nabla g(\mathbf{w}_{g,i})$,
end for
Onput: $\mathbf{w}_{g,T}$

tivation) of layer L, and $\odot$ denotes a Hadamard product. Replacing the activation functions ψ with polynomial approximations $\mathbf{p}$, replacing $\psi'(z_L) \cdot L'_y(a_L) = (L_y \circ \psi)'(z_L)$ with a polynomial approximation $\tilde{p}'_y$, and setting the polynomial $\tilde{p}_y$ to be an antiderivative of $\tilde{p}'_y$ yields the gradient of another function g:

$$\nabla g(\mathbf{W}^l) = \frac{1}{N} \sum_{(\mathbf{x},y)\in D} [\mathbf{p}'(\widetilde{\mathbf{z}}_\ell) \odot (\mathbf{W}^{\ell+1})^T \cdot \mathbf{p}'(\widetilde{\mathbf{z}}_{\ell+1}) \odot \cdots \odot (\mathbf{W}^L)^T \cdot \tilde{p}'_y(\widetilde{z}_L)]\widetilde{\mathbf{a}}_{\ell-1}^T,$$

where $g(\mathbf{w}) = \frac{1}{N}\sum_{(\mathbf{x},y)\in D} g_{(\mathbf{x},y)}(\mathbf{w})$ and $g_{(\mathbf{x},y)}(\mathbf{w}) = \tilde{p}_y(\mathbf{W}^L \cdots \mathbf{p}(\mathbf{W}^1\mathbf{x}))\cdots)))$.

For single-layer neural networks, ∇f has the form: $\nabla f(\mathbf{w}) = \frac{1}{N}\sum_{(\mathbf{x},y)\in\mathcal{D}}(L_y \circ \psi)'(\langle \mathbf{w}, \mathbf{x}\rangle) \cdot \mathbf{x}$ and ∇g has the form: $\nabla g(\mathbf{w}) = \frac{1}{N}\sum_{(\mathbf{x},y)\in\mathcal{D}} \tilde{p}'_y(\langle \mathbf{w}, \mathbf{x}\rangle) \cdot \mathbf{x}$.

3. Convergence of No-DP Approximate GD

Recall that we replace the gradient vector, $\nabla f(\mathbf{w})$, corresponding to objective function f with a vector of polynomial approximations, and that for functions f of interest, this approximate gradient is the *exact* gradient of a different function g (See Section 2.3). Consequently, we effectively run gradient descent with objective g. See Algorithm 1.

Let convex set $\mathcal{S} \subseteq \mathbb{R}^m$ be the domain of $f, g, \nabla f, \nabla g$. We assume ∇g and g satisfy the following:

Assumption 3.1 (Bounded Gradient Distance)**.**

$$\forall \mathbf{w} \in \mathcal{S}, \|\nabla f(\mathbf{w}) - \nabla g(\mathbf{w})\| \le \zeta.$$

Assumption 3.2 (Bounded Objective Distance)**.**

$$\forall \mathbf{w} \in \mathcal{S}, |f(\mathbf{w}) - g(\mathbf{w})| \le \alpha.$$

For all $\mathbf{w}, \mathbf{w}' \in \mathcal{S}$, a bound on $f(\mathbf{w}) - f(\mathbf{w}') - (g(\mathbf{w}) - g(\mathbf{w}'))$ follows directly from Assumption 3.1. Consequently, Assumption 3.2 can be omitted, see Appendix A.1.

Theorem 3.3. *Suppose objective functions f and g satisfy Assumptions 3.1 and 3.2, f is μ-strongly convex, g is β_g-smooth, and the learning rate $\eta_1 = \frac{1}{\beta_g}$. Fix $\rho > 0$ and let $T = \frac{2\beta_g(g(\mathbf{w}_{g,0}) - L)}{\rho^2}$, where $L \le g(\mathbf{w}_{g,T})$. Then we have the following guarantees on convergence of the output of Algorithm 1, $\mathbf{w}_{g,T}$, to the optimum, $\mathbf{w}^*_f$, of f:*

$$f(\mathbf{w}_{g,T}) - f(\mathbf{w}^*_f) \le 2\alpha + \frac{\rho^2 + 2\zeta\rho + \zeta^2}{2\mu}, \text{ and,}$$

$$\|\mathbf{w}_{g,T} - \mathbf{w}^*_f\|^2 \le \frac{4\alpha}{\mu} + \frac{\rho^2 + 2\zeta\rho + \zeta^2}{\mu^2}.$$

Remark 3.4. If objective function g has a global minimum, $\mathbf{w}^*_g$, then we can set $L = g(\mathbf{w}^*_g)$ in the above Theorem.

The proof of Theorem 3.3 can be found in Appendix A.

Algorithm 2 Approximate Gradient Descent with DP-noise.

Input: Running steps T, learning rate η_2, noise standard deviation σ.
Initialize $\mathbf{w}_{g,0} = \mathbf{0}$
for $i = 0$ to $T-1$ **do**
  $\mathbf{w}_{g,i+1} \leftarrow \mathbf{w}_{g,i} - \eta_2(\nabla g(\mathbf{w}_{g,i}) + \boldsymbol{\chi}_i)$,
  where $\boldsymbol{\chi}_i \sim \mathcal{N}(0, \sigma^2 \mathbf{I}_m)$.
end for
Ouput: $\mathbf{w}_{g,T}$

4. Convergence of DP-Approximate GD

We consider DP gradient descent with "approximate" objective function g, as specified in Algorithm 2. Since random noise is added to the gradient in each iteration, we employ an "average-case" analogue of strong convexity—called *expected curvature* (Yu et al., 2019) (see Definition 2.3) and denoted by ν—in the analysis. As in Section 3, we employ Assumption 3.1.

Theorem 4.1. *Suppose objective functions f and g satisfy Assumption 3.1, and f is β_f-smooth with ν expected curvature, then the output of Algorithm 2, $\mathbf{w}_T$, satisfies:*

$$\mathbb{E}[\|\mathbf{w}_{g,T} - \mathbf{w}^*_f\|^2] \le \left(1 - \frac{\eta_2\nu}{2}\right)^T \|\mathbf{w}_{g,0} - \mathbf{w}^*_f\|^2 + \frac{2\zeta^2}{\nu}\left(\frac{1}{\nu} + 1\right) + \frac{2\eta_2 m\sigma^2}{\nu}.$$

The proof appears in Appendix B.

5. DP-Approximate Gradient Descent Algorithm without Clipping

We consider here smooth, convex cost functions of the form $f(\mathbf{w}) = \frac{1}{N}\sum_{(\mathbf{x},y)\in D}\phi(\langle \mathbf{w}, \mathbf{x}\rangle, y) = \frac{1}{N}\sum_{(\mathbf{x},y)\in D} y\phi_1(\langle \mathbf{w}, \mathbf{x}\rangle) + (1-y)\phi_0(\langle \mathbf{w}, \mathbf{x}\rangle)$. [3] We denote by $\phi'(z, y)$ the function $\phi'(z, y) := y \cdot \frac{d\phi_1(x)}{dx}(z) + (1-y) \cdot \frac{d\phi_0(x)}{dx}(z)$. We assume $\phi'_{max} := \sup\{|\phi'(z, y)| : z \in (-\infty, \infty), y \in \{0, 1\}\} < \infty$.

[3] For notational simplicity, we write ϕ in place of $L_y \circ \psi$ as defined in Section 2.

Algorithm 3 Approximate Gradient Descent with Modified objective function f_κ^{+B}.

Input: Running steps T, learning rate η_3.
set $\sigma = \frac{2\Delta_2\sqrt{T\log(3/\delta)}}{\epsilon N}$, $\Delta_2 = 2(\phi'_{max} + e_f)\sqrt{m}$
Initialize $\mathbf{w}_0 = \mathbf{0}$
for $i = 0$ to $T-1$ **do**
$\mathbf{w}_{i+1} \leftarrow \mathbf{w}_i - \eta_3\Big(2\lambda P_\kappa(\Theta - \|\mathbf{w}_i\|^2)\mathbf{w}_i$
$+ \Big(\frac{1}{N}\sum_{(\mathbf{x},y)\in D} \tilde{p}'(\langle \mathbf{w}_i, \mathbf{x}\rangle, y)\mathbf{x}\Big) + \boldsymbol{\chi}_i\Big)$
where $\boldsymbol{\chi}_i \sim \mathcal{N}(0, \sigma^2\mathbf{I}_m)$.
end for
Output: $\mathbf{w}_T$

We begin by modifying the original objective f to $f_\kappa^{+B}(\mathbf{w}) := f(\mathbf{w}) - \lambda B_\kappa(\Theta - \|\mathbf{w}\|^2)$, where the additional term is a *barrier function*, a standard tool for enforcing constraints in optimization. We define $B_\kappa(\Theta - \|\mathbf{w}\|^2) := \ln(\Theta - \|\mathbf{w}\|^2)$ on the domain $\{\mathbf{w} : \|\mathbf{w}\|^2 \le (1-\kappa)\Theta\}$, and use it to control the magnitude of the iterates $\mathbf{w}_i$ in Algorithm 3. Specifically, the barrier term causes the modified objective to approach ∞ as $\|\mathbf{w}\|^2 \to \Theta$.

Bounding $\|\mathbf{w}_i\|$ immediately yields a bound on the hypothesis values, since $\langle \mathbf{w}_i, \mathbf{x}\rangle \le \sqrt{m}\,\|\mathbf{w}_i\|$. This bound is essential for differential privacy: it allows us to fix an approximation interval for $\phi'(z, y)$ and guarantees that all $z = \langle \mathbf{w}_i, \mathbf{x}\rangle$ encountered during gradient descent lie within this interval. Consequently, we can bound the sensitivity of the *approximate* gradient using the known sensitivity of the true gradient together with the maximum approximation error over the interval; this sensitivity, in turn, determines the noise required to achieve DP.

Because we require a bound on $\|\mathbf{w}_i\|$ at every iteration, not merely at the optimum, and because polynomial approximations and DP noise are present, the weight norm cannot be bounded directly by $\sqrt{\Theta}$. Instead, we derive an explicit upper bound on the iterates $\|\mathbf{w}_i\|$ in Lemma 5.5.

In Algorithm 3, we define $\tilde{p}'(z, y) := y\,\tilde{p}'_1(z) + (1-y)\,\tilde{p}'_0(z)$, where $\tilde{p}'_0$ and $\tilde{p}'_1$ are polynomial approximations of ϕ'_0 and ϕ'_1, respectively. The polynomial P_κ approximates $F_\kappa(x) := \frac{d}{dx}B_\kappa(x)$ over the interval $[\kappa\Theta, \Theta]$; note that $F_\kappa(x) = 1/x$ on this interval. All polynomials $\tilde{p}'_0$, $\tilde{p}'_1$, and P_κ are defined on $(-\infty, \infty)$.

Theorem 5.1. *[Differential Privacy of Algorithm 3] Fix constants $e_f \in [0, \phi'_{max}]$, $e_B \ge 0$, $\kappa \in (0,1)$. Let $\zeta_f = e_f\sqrt{m}$, $d \ge \frac{\|\nabla f(\mathbf{0})\|}{\sqrt{m}}$, $\mathsf{c}_\delta := \sqrt{2\ln(\frac{3T}{\delta})}$, and $R := \sqrt{(1-\kappa)\Theta} + \eta_3\left(\phi'_{max}\sqrt{m} + \zeta_f + 2\lambda e_B\sqrt{\Theta} + (\sqrt{m} + \mathsf{c}_\delta)\sigma\right)$. Let m_P, (resp. M_P) be the minimum (resp. maximum) value of P_κ on the interval $[\Theta - R^2, \kappa\Theta]$. If the following hold:*

- *For $y \in \{0,1\}$, $\tilde{p}'(z, y)$ has error at most e_f with respect to $\phi'(z, y)$ on the interval $z \in [-\sqrt{m}R, \sqrt{m}R]$.*
- *P_κ has error at most e_B with respect to F_κ on the interval $[\kappa\Theta, \Theta]$. and is monotonically* decreasing *on the interval $[\Theta - R^2, \kappa\Theta]$.*
- *$\mathsf{m}_P \ge 0$ and η_3 satisfies the following inequality:*

$$\eta_3 \le \min\left\{\frac{\kappa\Theta}{\lambda}, \frac{1}{\lambda(\mathsf{M}_P + \mathsf{m}_P) + \frac{(\phi'_{max}-d)\sqrt{m}}{2\sqrt{(1-\kappa)\Theta}}}\right\}. \tag{1}$$

- *κ satisfies the following inequality:*

$$\sqrt{(1-\kappa)\Theta} \ge \frac{-B + \sqrt{B^2 - 4AC}}{2A}, \tag{2}$$

where $\alpha = 2\eta_3\lambda\mathsf{m}_P$, $A = (2\alpha - \alpha^2)$, $B = -2\eta_3((1-\alpha)(d\sqrt{m} + \mathsf{c}_\delta\cdot\sigma) + \zeta_f)$, and $C = -\eta_3^2((\sqrt{m}\phi'_{max} + (\sqrt{m} + \mathsf{c}_\delta)\sigma)^2 - \zeta_f^2)$.

then the model, $\mathbf{w}_T$, outputted by Algorithm 3 achieves (ϵ, δ)-differential privacy.

Remark 5.2 (Role of monotonicity). Monotonicity of P_κ on $[\Theta - R^2, \Theta]$ is not required for DP. However, enforcing monotonicity simplifies the choice of κ and η_3 satisfying (2) and (14), and is used in our convergence analysis.

Remark 5.3 (Feasibility of parameters). For fixed $(\varepsilon, \delta), e_f, e_B, d$, constraints (1) and (2) can always be satisfied by choosing κ and η_3 sufficiently small. The required polynomials $\tilde{p}'$ and P_κ can likewise be constructed with sufficiently high degree to achieve the desired error (and monotonicity) over the specified intervals (DeVore, 1977; DeVore and Yu, 1985). See Appendix C.3 for details.

Remark 5.4 (Data-dependent refinement). The Client may compute $d = \|\nabla f(\mathbf{0})\|/\sqrt{m}$ and release it to the Server in a differentially private manner. When $d \ll \max\{\phi'_0(0), \phi'_1(0)\}$, as observed empirically, this yields significantly improved parameter choices. We adopt this refinement in our experiments, and account for releasing it in our privacy budget, which essentially has the effect of increasing the number of GD iterations by 1.

To prove Theorem 5.1, we first establish a bound on the magnitude of $\|\mathbf{w}_i\|$ in each iteration of Algorithm 3.

Lemma 5.5. *(Bounding magnitude of weights) Let $\mathbf{w}_i$ denote the i-th iterate of Algorithm 3. Under the parameter conditions of Theorem 5.1, with probability at least $1 - \frac{2\delta}{3}$, the following holds for all $i \in \{0, \ldots, T\}$:*

$$\|\mathbf{w}_i\| \le \sqrt{(1-\kappa)\Theta} + \eta_3\left(\phi'_{max}\sqrt{m} + \zeta_f + 2\lambda e_B\sqrt{\Theta} + (\sqrt{m} + \mathsf{c}_\delta)\sigma\right).$$

We note that Lemma 5.5—under the parameter settings of Theorem 5.1—guarantees that the weight bound fails with probability at most $\frac{2\delta}{3}$, and hence holds with probability at least $1-\frac{2\delta}{3}$. Conditioned on this event, Algorithm 3 satisfies $(\epsilon, \delta/3)$-DP. Hence, by a union bound, the overall algorithm satisfies (ϵ, δ)-DP. The proof of Lemma 5.5 is deferred to Appendix C.1.

Lemma 5.5 implies that $|\langle \mathbf{w}_i, \mathbf{x} \rangle| \leq \sqrt{m}R$, which allows us to fix the approximation interval for $\tilde{p}'(z, y)$ and bound the sensitivity.

Appendix C.2 shows that the ℓ_2 sensitivity of $\nabla g(\mathbf{w}_i)$ is $\frac{\Delta_2}{N}$, where $\Delta_2 \leq 2(\phi'_{\max} + e_f)\sqrt{m}$. Using the zCDP-based composition analysis of (Jayaraman et al., 2018; Bun and Steinke, 2016) and the inequality $\sqrt{\log(3/\delta) + \epsilon} - \sqrt{\log(3/\delta)} \geq \epsilon/(2\sqrt{2\log(3/\delta)})$ for $\epsilon \leq \log(3/\delta)$, Algorithm 3 is (ϵ, δ)-DP provided $\sigma \geq \frac{2\Delta_2\sqrt{T\log(3/\delta)}}{\epsilon N}$.

Convergence of Algorithm 3. To apply Theorem 4.1, which establishes convergence of approximate gradient descent with DP noise, to Algorithm 3, we extend the objective f_κ^{+B} to the domain $\{\mathbf{w} : \|\mathbf{w}\| \leq R\}$. We do so by extending the domain of $F_\kappa(x)$ to $x \in [\Theta - R^2, \Theta]$ and defining $B_\kappa(x)$ as its antiderivative. We further extend F_κ so that the resulting objective f_κ^{+B} is strongly convex and has bounded smoothness, enabling the application of Theorem 4.1. Finally, the maximum approximation error between F_κ and its polynomial approximation P_κ over this enlarged interval remains bounded by e_B. As a result, the gradient approximation error satisfies $\zeta \leq \zeta_f + 2\lambda e_B R$. See Appendix C.4.

6. Approximating Polynomials and Hyperparameter Selection

We first consider the non-DP setting, where training follows Algorithm 1. By Theorem 3.3, the number of iterations T required for convergence grows with the smoothness parameter β_g of the approximating polynomial, while the final optimization error $\|\mathbf{w}_{g,T} - \mathbf{w}_f^*\|^2$ is primarily controlled by ζ, which captures the approximation error. We empirically illustrate the tradeoff between smoothness/convergence rate and approximation/optimization error in Appendix D.1, showing that the minimax and least-squares polynomial approximations of a fixed degree exemplify this effect.

Appendix D.2 further shows that even mild underestimation of the approximation interval can cause divergence. In the FHE setting, however, the Server cannot detect when inputs fall outside this interval. In Lemma 5.5, we show that modifying the objective function (as in Algorithm 3) allows upperbounding the magnitude of the weights so that intervals can be fixed *a priori* with provable correctness; the non-DP case follows by setting $\sigma = 0$ in Algorithm 3.

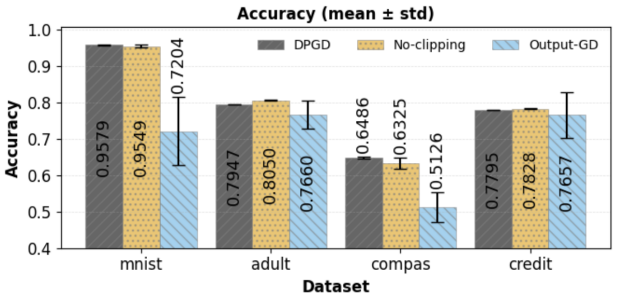


Figure 2. Visual comparison of model accuracy with error bars over 100 runs across four datasets under a privacy budget of ($\epsilon = 1$, $\delta = 10^{-5}$). Model 1: standard DP-GD; Model 2: augmented no-clipping DP-GD, where the sigmoid and barrier functions are respectively replaced by a 7th or 9th-degree polynomial and a 4th-degree polynomial; Model 3: DP model using output perturbation (Algorithm 2 in (Wu et al., 2017)). Error bars represent standard deviation across runs. (see Table 4 for numerical values.)

Beyond illustrating these tradeoffs empirically, Appendix D.3 provides data-independent guidance for selecting the polynomial approximation (e.g., degree and approximation interval) and hyperparameters (e.g., $\eta_3, \Theta, \lambda, \kappa$) such that all conditions of Theorem 5.1 are satisfied. This procedure enables DP training under FHE via Algorithm 3.

7. Experimental Results on DP FHE Training

We evaluate our proposed no-clipping approach (Algorithm 3) against standard DP-(S)GD, which enforces differential privacy via per-sample gradient clipping. For training under FHE, we use an SGD-based update for efficiency and compute the required noise scale using the subsampled DP analysis of (Wang et al., 2019). We first assess both methods in a plaintext setting to verify that the no-clipping approach maintains comparable model accuracy. We then implement both algorithms in the FHE domain (as Algorithm 5 vs Algorithm 4 in Section F) to compare runtime and accuracy. In the FHE setting, we conduct experiments under two security configurations with different ring dimensions (RingDim = 32768 and RingDim = 131072). While the two methods achieve similar accuracy, our approach reduces the multiplicative depth per iteration from 24 to 9, resulting in roughly a $3\times$ reduction (RingDim = 32768) and a $5\times$ reduction (RingDim = 131072) in computational cost (Tables 1, 2, and 5).

Data Preprocessing: We applied standard preprocessing steps to handle missing values, encode categorical variables numerically, and normalize all features to the range $[-1, 1]$. To mitigate resource and computational constraints, high-dimensional datasets, such as MNIST, were compressed via Principal Component Analysis (PCA) (Tipping and Bishop, 1999; Minka, 2000), using the scikit-learn implementation (Pedregosa et al., 2011). We do not include PCA

in the privacy accounting of our DP guarantee. Since it is applied uniformly to both our method and the DP-SGD baseline, DP-PCA is orthogonal to our analysis and does not affect comparisons.

Approximation Algorithms: We approximate the sigmoid and square-root functions using either a minimax polynomial, which minimizes $\sup_{x\in[a,b]} |f(x) - g(x)|$, or a least-squares polynomial, which minimizes $\int_a^b (f(x) - g(x))^2\, dx$ over the interval $[a, b]$. Efficient algorithms for computing optimal minimax and least-squares polynomials of fixed degree on a given interval are well known (Remez, 1934; Björck, 1996). The inverse operation is approximated using Taylor expansions or least-squares polynomials. Per-sample clipping requires a comparison operation, which we approximate following (Mazzone et al., 2025).

7.1. Empirical Results

We first evaluate both algorithms in the plaintext setting to enable a direct comparison of accuracy without confounding errors introduced by FHE. Results in this setting are still subject to randomness from DP noise, data splits, and other sources; therefore, each experiment was repeated 100 times and the results were averaged. We conducted this evaluation on four datasets: Adult (Becker and Kohavi, 1996), Compas (Angwin et al., 2016), Credit (Yeh, 2009), and a restructured version of MNIST (LeCun et al., 1998; Han, 2018). For MNIST, we considered a binary classification task distinguishing digits 3 and 8. As shown in Figure 2 and Table 4 (in Appendix), the no-clipping approach achieves accuracy comparable to standard DP-GD (e.g., a maximum drop of 1.61% in accuracy and 1.05% in AUC.)

We next analyze both methods in the FHE setting. As shown in Table 1 and 2, our proposed method attains accuracy comparable to standard DP-GD while matching the runtime efficiency of Output-GD, yielding at least a $3\times$ and $5\times$ speedup over DP-GD for RingDim = 32768 and 131072, respectively. Unlike Output-GD, which trades accuracy for speed, our approach preserves strong model performance. To accommodate limited computational resources, each iteration trains on a random subsample of 100 data points for DP-GD and Output-GD (98 for our method). Notably, as fewer parallel threads are used (i.e., each thread handles more data), the relative speedup over DP-GD increases, demonstrating superior scalability.[4]

Multiplicative Depth Analysis Table 1 shows that our method trains significantly faster than standard DP-GD. To explain this, we analyze the computational complexity in terms of multiplicative depth, a key metric drives overhead and noise growth in FHE (see Appendix Table 5). Standard DP-GD (Algorithm 4) is bottlenecked by costly per-sample gradient clipping under FHE. Our algorithm (Algorithm 5) removes clipping and instead uses a lightweight "Barrier Calculation" (depth 6) running once per iteration in parallel with simplified gradient (depth 8). The critical path depth is thus $\max(8, 6) = 8$. Including the weight update, total depth per iteration drops from 24 to 9, a $2.5\times$ reduction that explains the observed FHE speedup.

Observations. Our empirical comparison shows that our proposed no-clipping algorithm outperforms the traditional clipping-based approach for DP-ML under FHE. Both methods achieve comparable accuracy, with the no-clipping model at most a 0.01% accuracy drop and a 0.33% AUC decrease. Compared to output perturbation (Output-GD), which runs standard (no-DP) GD under FHE and adds noise to the output, our method essentially matches its computation time while delivering significantly higher accuracy.

Beyond efficiency, our approach addresses fundamental limitations of DP-GD under FHE. The standard approach requires a polynomial approximation of ϕ' over an input interval determined by the data-dependent hypothesis value $z = \langle \mathbf{w}, \mathbf{x} \rangle$, which varies dynamically during training. If this interval is chosen incorrectly, the approximation of ϕ', and thus also the approximate squared gradient norm, may become unbounded, leading to divergence or severe errors in training. For example, during MNIST training, using a square-root approximation over $[0, 10]$ caused divergence after only 10 iterations. Under FHE, such failures cannot be remedied by iteratively retuning approximation intervals, since interactive corrections are not possible. In contrast, our method explicitly bounds z and provides formal guarantees on the required approximation intervals (see Theorem 5.1).

These issues affect not only convergence but also DP. In particular, clipping involves normalization of the gradient, which relies on polynomial approximations of $1/x$. If the gradient norm falls outside the approximation interval, this normalization can become arbitrarily inaccurate, and the effective clipping threshold can no longer be assumed to equal C. As a result, the DP sensitivity bound may be violated. Our no-clipping approach explicitly accounts for all approximation errors and therefore yields provable differential privacy guarantees (see Theorem 5.1 and Algorithm 3).

Finally, the no-clipping method offers substantial practical gains under FHE, reducing multiplicative depth by more than $2.5\times$ and cutting training time by more than one-third, making it both theoretically rigorous and practically viable for privacy-preserving ML.

8. Discussion and Limitations

This work provides the first formal analysis of DP for gradient descent with polynomial approximations in the FHE

[4]Experiments can be reproduced via: github.com/dpfhe096-design/dp_fhe(including e.g., T, η, and FHE-related settings).

Table 1. Model performance under FHE on an AMD EPYC 9534 with multi-thread parallelism ($\epsilon = 1, \delta = 10^{-5}$, RingDim = 32768)

Data	Training Model	ACC	AUC	10-threads	20-threads	30-threads	50-threads
mnist	Algo 4 (DP-SGD)	93.62%	98.21%	600.1 sec/iter	338.0 sec/iter	283.0 sec/iter	187.2 sec/iter
	Algo 5 (No Clip)	93.99%	98.22%	148.2 sec/iter	93.0 sec/iter	86.9 sec/iter	58.2 sec/iter
	Output-SGD	74.18%	88.59%	147.5 sec/iter	90.1 sec/iter	82.9 sec/iter	55.7 sec/iter
credit	Algo 4 (DP-SGD)	78.00%	68.88%	446.1 sec/iter	258.7 sec/iter	227.5 sec/iter	164.0 sec/iter
	Algo 5 (No Clip)	77.99%	68.69%	132.2 sec/iter	79.4 sec/iter	70.4 sec/iter	53.7 sec/iter
	Output-SGD	76.76%	54.82%	133.9 sec/iter	82.7 sec/iter	75.7 sec/iter	53.9 sec/iter

Table 2. Model performance under FHE on an AMD EPYC 7502 with multi-thread parallelism ($\epsilon = 1, \delta = 10^{-5}$, RingDim = 131072, 128-bit security)

Data	Training Model	ACC	AUC	10-threads	20-threads	30-threads	50-threads
mnist	Algo 4 (DP-SGD)	93.77%	98.20%	7996.8 sec/iter	4240.0 sec/iter	3504.4 sec/iter	1813.5 sec/iter
	Algo 5 (No Clip)	93.97%	98.20%	566.7 sec/iter	442.2 sec/iter	409.8 sec/iter	350.5 sec/iter
	Output-SGD	73.87%	86.68%	524.5 sec/iter	434.9 sec/iter	397.4 sec/iter	351.1 sec/iter
credit	Algo 4 (DP-SGD)	77.96%	69.18%	8028.1 sec/iter	3910.5 sec/iter	3236.3 sec/iter	1843.3 sec/iter
	Algo 5 (No Clip)	77.99%	68.85%	521.9 sec/iter	449.2 sec/iter	394.1 sec/iter	343.1 sec/iter
	Output-SGD	76.76%	55.61%	551.4 sec/iter	415.8 sec/iter	379.9 sec/iter	342.0 sec/iter

setting. We introduce an efficient training algorithm that is provably DP even in the presence of polynomial approximation error. We establish convergence and end-to-end privacy under encryption, and show that polynomial smoothness and approximation error govern convergence and model utility, enabling principled choices of approximation scheme, degree, and interval. Together, these results strengthen the theory and practice of private, encrypted learning.

Our combined convergence and DP analysis, however, is currently limited to smooth, convex objectives[5], where iterate norms can be tightly controlled to enable sensitivity analysis. Extending these results to general neural networks is an important direction for future work. The core mechanism, augmenting the objective with a barrier, may extend beyond convex settings by helping bound weight norms, stabilize activations, and support sensitivity control under polynomial approximations, though formal guarantees remain open for future work.

Impact Statement

This paper advances machine learning by enabling fully private model training through the combination of Fully Homomorphic Encryption (FHE) and Differential Privacy (DP). We provide the first theoretical convergence analysis for gradient descent with polynomial approximations compatible with FHE, together with a differentially private training algorithm designed for this encrypted setting. Our approach enables learning directly on encrypted data while also producing models that are themselves differentially private, protecting individuals not only during training but also against information leakage in downstream use.

By removing the need for costly per-sample gradient clipping, our methods substantially improve the practicality of secure, outsourced learning for data owners who lack computational resources, particularly in sensitive domains such as healthcare, finance, and government services. Although FHE-based training incurs higher computational costs than plaintext learning, our work reduces these overheads and improves scalability, bringing end-to-end private learning closer to real-world deployment. We believe the societal benefits of enabling verifiable, confidential training and privacy-preserving model release, without requiring trust in the training infrastructure, outweigh the remaining computational costs and any modest impact on model utility.

[5]We note that adding the barrier function to the objective induces strong convexity over the interior of the feasible region.

Acknowledgments and Disclosure of Funding

The authors thank the anonymous reviewers for their insightful comments and valuable feedback on this work. Dana Dachman-Soled is supported in part by NSF grants CNS-2154705, CNS-1933033, and IIS2147276 is jointly provided with Min Wu. We gratefully acknowledge support from the JP Morgan Chase Faculty Research Award.

References

Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, oct 2016. doi: 10.1145/2976749.2978318. URL https://doi.org/10.1145%2F2976749.2978318.

Prabhanjan Ananth, Abhishek Jain, Zhengzhong Jin, and Giulio Malavolta. Multi-key fully-homomorphic encryption in the plain model. In *Theory of Cryptography Conference*, pages 28–57. Springer, 2020.

Galen Andrew, Om Thakkar, Brendan McMahan, and Swaroop Ramaswamy. Differentially private learning with adaptive clipping. *Advances in Neural Information Processing Systems*, 34:17455–17466, 2021.

J. Angwin, J. Larson, S. Mattu, and L. Kirchner. How we analyzed the compas recidivism algorithm. ProPublica, 2016. https://www.propublica.org/article/how-we-analyzed-the-compas-recidivism-algorithm.

Rezak Aziz, Soumya Banerjee, Samia Bouzefrane, and Thinh Le Vinh. Exploring homomorphic encryption and differential privacy techniques towards secure federated learning paradigm. *Future internet*, 15(9):310, 2023.

Raef Bassily, Adam Smith, and Abhradeep Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *2014 IEEE 55th annual symposium on foundations of computer science*, pages 464–473. IEEE, 2014.

Barry Becker and Ronny Kohavi. Adult. UCI Machine Learning Repository, 1996. DOI: https://doi.org/10.24432/C5XW20.

S Bernstein. Proof of the theorem of weierstrass based on the calculus of probabilities. *Communications of the Kharkov Mathematical Society*, 13:1–2, 1912.

Åke Björck. *Numerical Methods for Least Squares Problems*. SIAM (Society for Industrial and Applied Mathematics), 1996. doi: 10.1137/1.9781611971484.

Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. Fully homomorphic encryption without bootstrapping. Cryptology ePrint Archive, Paper 2011/277, 2011. URL https://eprint.iacr.org/2011/277.

Mark Bun and Thomas Steinke. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Theory of cryptography conference*, pages 635–658. Springer, 2016.

Kamalika Chaudhuri, Claire Monteleoni, and Anand D Sarwate. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12(3), 2011.

Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 409–437. Springer, 2017.

Gaby G Dagher, Fung Benjamin, Mohammed Noman, and Jeremy Clark. Secdm: privacy-preserving data outsourcing framework with differential privacy. *Knowledge and Information Systems*, 62(5):1923–1960, 2020.

Ronald A. DeVore. Monotone approximation by polynomials. *SIAM Journal on Mathematical Analysis*, 8(5): 906–921, 1977. doi: 10.1137/0508069. Early result on uniform monotone polynomial approximation.

Ronald A. DeVore and Xiang Ming Yu. Pointwise estimates for monotone polynomial approximation. *Constructive Approximation*, 1:323–331, 1985. doi: 10.1007/BF02392357. Gives estimates for monotone polynomial approximation of continuous monotone functions.

Léo Ducas and Daniele Micciancio. FHEW: Bootstrapping homomorphic encryption in less than a second. Cryptology ePrint Archive, Paper 2014/816, 2014. URL https://eprint.iacr.org/2014/816.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Paper 2012/144, 2012. URL https://eprint.iacr.org/2012/144.

Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 1322–1333, 2015.

Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 169–178, 2009.

Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *International conference on machine learning*, pages 201–210. PMLR, 2016.

Kyoohyung Han. HELR: Homomorphic encryption for logistic regression. https://github.com/KyoohyungHan/HELR, 2018.

Kyoohyung Han, Seungwan Hong, Jung Hee Cheon, and Daejun Park. Efficient logistic regression on large encrypted data. *Cryptology ePrint Archive*, 2018.

Bargav Jayaraman, Lingxiao Wang, David Evans, and Quanquan Gu. Distributed learning without distress: Privacy-preserving empirical risk minimization. *Advances in neural information processing systems*, 31, 2018.

Miran Kim, Yongsoo Song, Shuang Wang, Yuhou Xia, Xiaoqian Jiang, et al. Secure logistic regression based on homomorphic encryption: Design and evaluation. *JMIR medical informatics*, 6(2):e8805, 2018.

Yann LeCun, Corinna Cortes, and Christopher J. C. Burges. THE MNIST DATABASE of handwritten digits, 1998. URL http://yann.lecun.com/exdb/mnist/. Accessed: 2018.

Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 1219–1234, 2012.

Federico Mazzone, Maarten Everts, Florian Hahn, and Andreas Peter. Efficient ranking, order statistics, and sorting under {CKKS}. In *34th USENIX Security Symposium (USENIX Security 25)*, pages 8541–8558, 2025.

Kentaro Mihara, Ryohei Yamaguchi, Miguel Mitsuishi, and Yusuke Maruyama. Neural network training with homomorphic encryption, 2020. URL https://arxiv.org/abs/2012.13552.

Thomas P. Minka. Automatic choice of dimensionality for pca. In *Advances in Neural Information Processing Systems*, volume 13, pages 598–604, 2000. URL https://proceedings.neurips.cc/paper/2000/file/7503cfacd12053d309b6bed5c89de212-Paper.pdf.

Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key fhe. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 735–763. Springer, 2016.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Le Trieu Phong, Yoshinori Aono, Takuya Hayashi, Lihua Wang, and Shiho Moriai. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Transactions on Information Forensics and Security*, 13(5): 1333–1345, 2018. doi: 10.1109/TIFS.2017.2787987.

E. Ya. Remez. Sur la détermination des polynômes d'approximation de degré donné. *Comm. Soc. Math. Kharkov*, 10:41, 1934.

John A Roulier. Permissible bounds on the coefficients of approximating polynomials. *Journal of Approximation Theory*, 3(2):117–122, 1970. ISSN 0021-9045. doi: https://doi.org/10.1016/0021-9045(70)90018-3. URL https://www.sciencedirect.com/science/article/pii/0021904570900183.

Andreas Santucci, Robin Brown, and Reza Zadeh. Lecture 8: Introduction to optimization for machine learning. Stanford University CME 323 Lecture Notes, 2020. URL https://stanford.edu/~rezab/dao/notes/L08/cme323_lec8.pdf. Lecture notes, April 23, 2020.

Shuang Song, Kamalika Chaudhuri, and Anand D. Sarwate. Stochastic gradient descent with differentially private updates. In *2013 IEEE Global Conference on Signal and Information Processing*, pages 245–248, 2013. doi: 10.1109/GlobalSIP.2013.6736861.

Arnaud Grivet Sébert, Marina Checri, Oana Stan, Renaud Sirdey, and Cédric Gouy-Pailler. Combining homomorphic encryption and differential privacy in federated learning. In *2023 20th Annual International Conference on Privacy, Security and Trust (PST)*, pages 1–7, 2023. doi: 10.1109/PST58708.2023.10320195.

Michael E. Tipping and Christopher M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):

611–622, 1999. URL http://www.miketipping.com/papers/met-mppca.pdf.

Yu-Xiang Wang, Borja Balle, and Shiva Prasad Kasiviswanathan. Subsampled rényi differential privacy and analytical moments accountant. In *The 22nd international conference on artificial intelligence and statistics*, pages 1226–1235. PMLR, 2019.

Xi Wu, Fengan Li, Arun Kumar, Kamalika Chaudhuri, Somesh Jha, and Jeffrey Naughton. Bolt-on differential privacy for scalable stochastic gradient descent-based analytics. In *Proceedings of the 2017 ACM international conference on management of data*, pages 1307–1322, 2017.

Heng Ye, Jiqiang Liu, Wei Wang, Ping Li, Tong Li, and Jin Li. Secure and efficient outsourcing differential privacy data release scheme in cyber–physical system. *Future Generation Computer Systems*, 108:1314–1323, 2020.

I-Cheng Yeh. Default of Credit Card Clients. UCI Machine Learning Repository, 2009. DOI: https://doi.org/10.24432/C55S3H.

Da Yu, Huishuai Zhang, Wei Chen, Tie-Yan Liu, and Jian Yin. Gradient perturbation is underrated for differentially private convex optimization. *arXiv preprint arXiv:1911.11363*, 2019.

A. Proof of Theorem 3.3

We begin by restating Theorem 3.3.

Theorem 3.3. *Suppose objective functions f and g satisfy Assumptions 3.1 and 3.2, f is μ-strongly convex, g is β_g-smooth, and the learning rate $\eta_1 = \frac{1}{\beta_g}$. Fix $\rho > 0$ and let $T = \frac{2\beta_g(g(\mathbf{w}_{g,0})-L)}{\rho^2}$, where $L \leq g(\mathbf{w}_{g,T})$. Then we have the following guarantees on convergence of the output of Algorithm 1, $\mathbf{w}_{g,T}$, to the optimum, $\mathbf{w}_f^*$, of f:*

$$f(\mathbf{w}_{g,T}) - f(\mathbf{w}_f^*) \leq 2\alpha + \frac{\rho^2 + 2\zeta\rho + \zeta^2}{2\mu}, \text{ and,}$$

$$\|\mathbf{w}_{g,T} - \mathbf{w}_f^*\|^2 \leq \frac{4\alpha}{\mu} + \frac{\rho^2 + 2\zeta\rho + \zeta^2}{\mu^2}.$$

To prove Theorem 3.3, we analyze gradient descent on the (non-convex) approximate objective g. By Claim A.1, within T iterations there exists an iterate $\mathbf{w}_{g,\tilde{t}}$, for some $\tilde{t} \in [T]$, such that $\|\nabla g(\mathbf{w}_{g,\tilde{t}})\| \leq \rho$. Assumption 3.1 then implies $\|\nabla f(\mathbf{w}_{g,\tilde{t}})\| \leq \rho + \zeta$.

The same claim guarantees $g(\mathbf{w}_{g,T}) \leq g(\mathbf{w}_{g,\tilde{t}})$, and Assumption 3.2 yields $f(\mathbf{w}_{g,T}) \leq f(\mathbf{w}_{g,\tilde{t}}) + 2\alpha$. Using the bound on $\|\nabla f(\mathbf{w}_{g,\tilde{t}})\|$ and the strong convexity of f, we upper bound $f(\mathbf{w}_{g,\tilde{t}}) - f(\mathbf{w}_f^*)$, where $\mathbf{w}_f^*$ is the minimizer of f. Combining these inequalities gives a bound on $f(\mathbf{w}_{g,T}) - f(\mathbf{w}_f^*)$, which directly implies a bound on $\|\mathbf{w}_{g,T} - \mathbf{w}_f^*\|^2$. We now present the detailed step-by-step proof.

To prove Theorem 3.3 we begin with the following claim, restated from (Santucci et al., 2020).

Claim A.1 (In (Santucci et al., 2020), Theorem 8.3)**.** *Fix $\rho > 0$ and let $T = \frac{2\beta_g(g(\mathbf{w}_{g,0})-L)}{\rho^2}$. Then there exists $\tilde{t} \in [T]$ such that $g(\mathbf{w}_{g,T}) \leq g(\mathbf{w}_{g,\tilde{t}})$ and $\|\nabla g(\mathbf{w}_{g,\tilde{t}})\| \leq \rho$, where $\mathbf{w}_{g,\tilde{t}}, \mathbf{w}_{g,T}$ are defined as in Algorithm 1 with $\eta_1 = \frac{1}{\beta_g}$.*

We now complete the proof of Theorem 3.3, given Claim A.1.

Proof of Theorem 3.3. Let $T, \tilde{t}$ be as in Claim A.1. Then

$$\begin{aligned}
&f(\mathbf{w}_{g,T}) - f(\mathbf{w}_f^*) = f(\mathbf{w}_{g,T}) - f(\mathbf{w}_{g,\tilde{t}}) + f(\mathbf{w}_{g,\tilde{t}}) - f(\mathbf{w}_f^*) \\
&= g(\mathbf{w}_{g,T}) - g(\mathbf{w}_{g,\tilde{t}}) + \big(f(\mathbf{w}_{g,T}) - f(\mathbf{w}_{g,\tilde{t}}) - (g(\mathbf{w}_{g,T}) - g(\mathbf{w}_{g,\tilde{t}}))\big) + f(\mathbf{w}_{g,\tilde{t}}) - f(\mathbf{w}_f^*) \\
&\leq g(\mathbf{w}_{g,T}) - g(\mathbf{w}_{g,\tilde{t}}) + 2\alpha + f(\mathbf{w}_{g,\tilde{t}}) - f(\mathbf{w}_f^*) \qquad (3) \\
&\leq 2\alpha + f(\mathbf{w}_{g,\tilde{t}}) - f(\mathbf{w}_f^*), \qquad (4)
\end{aligned}$$

where (3) follows from Assumption 3.2 and (4) follows from the fact that $g(\mathbf{w}_{g,T}) \leq g(\mathbf{w}_{g,\tilde{t}})$.

Since $\|\nabla g(\mathbf{w}_{g,\tilde{t}})\| \leq \rho$, we have by Assumption 3.1 that $\|\nabla f(\mathbf{w}_{g,\tilde{t}})\|^2 \leq \rho^2 + 2\zeta\rho + \zeta^2$ and by the Polyak–Lojasiewicz condition that

$$f(\mathbf{w}_{g,\tilde{t}}) - f(\mathbf{w}_f^*) \leq \frac{\rho^2 + 2\zeta\rho + \zeta^2}{2\mu}.$$

Substituting into (4), we obtain

$$f(\mathbf{w}_{g,T}) - f(\mathbf{w}_f^*) \leq 2\alpha + \frac{\rho^2 + 2\zeta\rho + \zeta^2}{2\mu}.$$

By the Quadratic-Growth condition, we can also bound,

$$\begin{aligned}
\|\mathbf{w}_{g,T} - \mathbf{w}_f^*\|^2 &\leq \frac{2}{\mu} \cdot (f(\mathbf{w}_{g,T}) - f(\mathbf{w}_f^*)) \\
&= \frac{4\alpha}{\mu} + \frac{\rho^2 + 2\zeta\rho + \zeta^2}{\mu^2}.
\end{aligned}$$

□

A.1. Removing Assumption 3.2

In this section, we will show the following bound:

$$f(\mathbf{w}_{g,T}) - f(\mathbf{w}_{g,\tilde{t}}) - (g(\mathbf{w}_{g,T}) - g(\mathbf{w}_{g,\tilde{t}})) \leq \zeta \cdot \|\mathbf{w}_{g,T} - \mathbf{w}_{g,\tilde{t}}\|. \tag{5}$$

This bound allows us to eliminate the additional assumption that

$$\forall \mathbf{w} \in \mathcal{D}, |f(\mathbf{w}) - g(\mathbf{w})| \leq \alpha.$$

Specifically, in the analysis in Appendix A above, (3) can be replaced with

$$g(\mathbf{w}_{g,T}) - g(\mathbf{w}_{g,\tilde{t}}) + \zeta \cdot \|\mathbf{w}_{g,T} - \mathbf{w}_{g,\tilde{t}}\| + f(\mathbf{w}_{g,\tilde{t}}) - f(\mathbf{w}_f^*),$$

yielding the final bounds of

$$f(\mathbf{w}_{g,T}) - f(\mathbf{w}_f^*) \leq \zeta \cdot \|\mathbf{w}_{g,T} - \mathbf{w}_{g,\tilde{t}}\| + \frac{\rho^2 + 2\zeta\rho + \zeta^2}{2\mu}$$

and

$$\|\mathbf{w}_{g,T}) - \mathbf{w}_f^*\|^2 \leq \frac{2\zeta \cdot \|\mathbf{w}_{g,T} - \mathbf{w}_{g,\tilde{t}}\|}{\mu} + \frac{\rho^2 + 2\zeta\rho + \zeta^2}{\mu^2}.$$

We now prove the upperbound given in (5): let C be the line that goes from $\mathbf{w}_{g,\tilde{t}}$ to $\mathbf{w}_{g,T}$. This line can be parameterized in terms of a single variable τ such that $\ell(\tau) = \mathbf{w}_{g,\tilde{t}} + \tau \cdot (\mathbf{w}_{g,T} - \mathbf{w}_{g,\tilde{t}})$, where $\tau \in [0, 1]$. We have that

$$\begin{aligned}
\int_C \langle \nabla f(\mathbf{w}) - \nabla g(\mathbf{w}), d\mathbf{w} \rangle &= \int_0^1 \langle \nabla f(\ell(\tau)) - \nabla g(\ell(\tau)), \frac{d\ell(\tau)}{d\tau} \rangle d\tau \\
&= \int_0^1 \langle \nabla f(\ell(\tau)) - \nabla g(\ell(\tau)), \mathbf{w}_{g,T} - \mathbf{w}_{g,\tilde{t}} \rangle d\tau \\
&\leq \int_0^1 ||\nabla f(\ell(\tau)) - \nabla g(\ell(\tau)))|| \cdot ||\mathbf{w}_{g,T} - \mathbf{w}_{g,\tilde{t}}|| d\tau \\
&\leq \int_0^1 \zeta \cdot ||\mathbf{w}_{g,T} - \mathbf{w}_{g,\tilde{t}}|| d\tau \qquad (6) \\
&= \zeta \cdot ||\mathbf{w}_{g,T} - \mathbf{w}_{g,\tilde{t}})||, \qquad (7)
\end{aligned}$$

where (6) follows from Assumption 3.1 holding on the convex set $\mathcal{S}$. Since $\mathbf{w}_{g,\tilde{t}}, \mathbf{w}_{g,T} \in \mathcal{S}$, the line segment $\ell(\tau) = \mathbf{w}_{g,\tilde{t}} + \tau \cdot (\mathbf{w}_{g,T} - \mathbf{w}_{g,\tilde{t}})$ lies entirely in $\mathcal{S}$ for all $\tau \in [0, 1]$, implying

$$\|\nabla f(\ell(\tau)) - \nabla g(\ell(\tau))\| \leq \zeta \quad \text{for all } \tau \in [0, 1].$$

On the other hand, by the gradient theorem for line integrals

$$\begin{aligned}
\int_C \langle \nabla f(\mathbf{w}) - \nabla g(\mathbf{w}), d\mathbf{w} \rangle &= f(\mathbf{w}_{g,T}) - g(\mathbf{w}_{g,T}) - (f(\mathbf{w}_{g,\tilde{t}}) - g(\mathbf{w}_{g,\tilde{t}})) \\
&= f(\mathbf{w}_{g,T}) - f(\mathbf{w}_{g,\tilde{t}}) - (g(\mathbf{w}_{g,T}) - g(\mathbf{w}_{g,\tilde{t}})) \qquad (8)
\end{aligned}$$

Combining (7) and (8)we obtain:

$$f(\mathbf{w}_{g,T}) - f(\mathbf{w}_{g,\tilde{t}}) - (g(\mathbf{w}_{g,T}) - g(\mathbf{w}_{g,\tilde{t}})) \leq \zeta \cdot ||\mathbf{w}_{g,T} - \mathbf{w}_{g,\tilde{t}})||,$$

as desired.

B. Proof of Theorem 4.1

We begin by restating Theorem 4.1.

Theorem 4.1. *Suppose objective functions f and g satisfy Assumption 3.1, and f is β_f-smooth with ν expected curvature, then the output of Algorithm 2, $\mathbf{w}_T$, satisfies:*

$$\begin{aligned}&\mathbb{E}[\|\mathbf{w}_{g,T} - \mathbf{w}_f^*\|^2] \\ &\leq \left(1 - \frac{\eta_2\nu}{2}\right)^T \|\mathbf{w}_{g,0} - \mathbf{w}_f^*\|^2 + \frac{2\zeta^2}{\nu}\left(\frac{1}{\nu} + 1\right) + \frac{2\eta_2 m\sigma^2}{\nu}.\end{aligned}$$

To prove Theorem 4.1, we analyze the evolution of the expected squared distance between the iterate $\mathbf{w}_{g,i}$, obtained by optimizing the approximate objective g, and the optimal solution $\mathbf{w}_f^*$ of the true objective f.

We begin by expanding $\mathbf{w}_{g,i+1}$ as the sum of $\mathbf{w}_{g,i}$, the gradient update step, and the injected DP noise, and express $\mathbb{E}\left[\|\mathbf{w}_{g,i+1} - \mathbf{w}_f^*\|^2\right]$ in terms of $\mathbf{w}_{g,i}$. Using Assumption 3.1, we relate the gradient of the approximate objective to that of the true objective, replacing $\nabla g(\mathbf{w}_{g,i})$ with $\nabla f(\mathbf{w}_{g,i})$ plus the error term, ζ.

This reduction allows us to apply standard convexity, smoothness, and expected curvature properties of f to bound the expected progress made in each iteration, following a classical gradient descent convergence argument. Iterating this bound for $i = 0, \ldots, T-1$ yields a convergence guarantee on $\mathbb{E}\left[\|\mathbf{w}_{g,T} - \mathbf{w}_f^*\|^2\right]$, completing the proof.

Proof. Below, we set $\eta_2 \leq \frac{1}{2\beta_f}$.

We first bound $\mathbb{E}[\|\mathbf{w}_{g,i+1} - \mathbf{w}_f^*\|^2]$ in terms of $\mathbb{E}[\|\mathbf{w}_{g,i} - \mathbf{w}_f^*\|^2]$. We begin by expanding $\mathbb{E}[\|\mathbf{w}_{g,i+1} - \mathbf{w}_f^*\|^2]$:

$$\begin{aligned}\mathbb{E}[\|\mathbf{w}_{g,i+1} - \mathbf{w}_f^*\|^2] &= \mathbb{E}[\|\mathbf{w}_{g,i} - \eta_2\nabla g(\mathbf{w}_{g,i}) - \eta_2\boldsymbol{\chi}_i - \mathbf{w}_f^*\|^2] \\ &= \mathbb{E}[\|\mathbf{w}_{g,i} - \mathbf{w}_f^*\|^2] - 2\eta_2\mathbb{E}[\langle\nabla g(\mathbf{w}_{g,i}), \mathbf{w}_{g,i} - \mathbf{w}_f^*\rangle] + \eta_2^2\mathbb{E}[\|\nabla g(\mathbf{w}_{g,i})\|^2] + \eta_2^2 m\sigma^2 \qquad (9)\end{aligned}$$

We next replace the term $-2\eta_2\mathbb{E}[\langle\nabla g(\mathbf{w}_{g,i}), \mathbf{w}_{g,i} - \mathbf{w}_f^*\rangle]$ in (9) with $-2\eta_2\mathbb{E}[\langle\nabla f(\mathbf{w}_{g,i}), \mathbf{w}_{g,i} - \mathbf{w}_f^*\rangle] + 2\eta_2\mathbb{E}[\langle\mathbf{w}_{g,i} - \mathbf{w}_f^*, \nabla f(\mathbf{w}_{g,i}) - \nabla g(\mathbf{w}_{g,i})\rangle]$ to obtain:

$$\begin{aligned}(9) = \mathbb{E}[\|\mathbf{w}_{g,i} - \mathbf{w}_f^*\|^2] &- 2\eta_2\mathbb{E}[\langle\nabla f(\mathbf{w}_{g,i}), \mathbf{w}_{g,i} - \mathbf{w}_f^*\rangle] \\ &+ \eta_2^2\mathbb{E}[\|\nabla g(\mathbf{w}_{g,i})\|^2] + 2\eta_2\mathbb{E}[\langle\mathbf{w}_{g,i} - \mathbf{w}_f^*, \nabla f(\mathbf{w}_{g,i}) - \nabla g(\mathbf{w}_{g,i})\rangle] + \eta_2^2 m\sigma^2 \qquad (10)\end{aligned}$$

We next use expected curvature, convexity, and smoothness to upper bound the term $-2\eta_2\mathbb{E}[\langle\nabla f(\mathbf{w}_{g,i}), \mathbf{w}_{g,i} - \mathbf{w}_f^*\rangle]$ as follows:

$$\begin{aligned}-2\eta_2\mathbb{E}[\langle\nabla f(\mathbf{w}_{g,i}), \mathbf{w}_{g,i} - \mathbf{w}_f^*\rangle] &= -\eta_2\mathbb{E}[\langle\nabla f(\mathbf{w}_{g,i}), \mathbf{w}_{g,i} - \mathbf{w}_f^*\rangle] - \eta_2\mathbb{E}[\langle\nabla f(\mathbf{w}_{g,i}), \mathbf{w}_{g,i} - \mathbf{w}_f^*\rangle] \\ &\leq -\eta_2\nu\mathbb{E}[\|\mathbf{w}_{g,i} - \mathbf{w}_f^*\|^2] - \eta_2\mathbb{E}[\langle\nabla f(\mathbf{w}_{g,i}), \mathbf{w}_{g,i} - \mathbf{w}_f^*\rangle] \qquad (11) \\ &\leq -\eta_2\nu\mathbb{E}[\|\mathbf{w}_{g,i} - \mathbf{w}_f^*\|^2] - \eta_2\mathbb{E}[f(\mathbf{w}_{g,i}) - f(\mathbf{w}_f^*)]] \qquad (12) \\ &\leq -\eta_2\nu\mathbb{E}_[\|\mathbf{w}_{g,i} - \mathbf{w}_f^*\|^2] - \frac{\eta_2}{2\beta_f}\mathbb{E}[\|\nabla f(\mathbf{w}_{g,i})\|^2] \qquad (13) \\ &\leq -\eta_2\nu\mathbb{E}[\|\mathbf{w}_{g,i} - \mathbf{w}_f^*\|^2] - 2\eta_2^2\mathbb{E}[\|\nabla f(\mathbf{w}_{g,i})\|^2], \qquad (14)\end{aligned}$$

where (11) holds due to the expected curvature of f being at least ν, (12) holds due to convexity of f, (13) holds due to the β_f-smoothness condition, and (14) holds due to setting $\eta_2 \leq \frac{1}{2\beta_f}$, which implies that $-\frac{\eta_2}{2\beta_f}\mathbb{E}[\|\nabla f(\mathbf{w}_{g,i})\|^2] \leq -2\eta_2^2\mathbb{E}[\|\nabla f(\mathbf{w}_{g,i})\|^2]$.

Replacing $-2\eta_2\mathbb{E}[\langle\nabla f(\mathbf{w}_{g,i}), \mathbf{w}_{g,i} - \mathbf{w}_f^*\rangle]$ in (10) with (14) yields:

$$\begin{aligned}(10) \leq \mathbb{E}[\|\mathbf{w}_{g,i} - \mathbf{w}_f^*\|^2] &- \eta_2\nu\mathbb{E}[\|\mathbf{w}_{g,i} - \mathbf{w}_f^*\|^2] - 2\eta_2^2\mathbb{E}[\|\nabla f(\mathbf{w}_{g,i})\|^2] \\ &+ \eta_2^2\mathbb{E}[\|\nabla g(\mathbf{w}_{g,i})\|^2] + 2\eta_2\mathbb{E}[\langle\mathbf{w}_{g,i} - \mathbf{w}_f^*, \nabla f(w_{g,i}) - \nabla g(w_{g,i})\rangle] + \eta_2^2 m\sigma^2, \qquad (15)\end{aligned}$$

Re-arranging terms and using Assumption 3.1 to upperbound the following term

$$\begin{aligned}\mathbb{E}[\|\nabla g(\mathbf{w}_{g,i})\|^2] &= \mathbb{E}[\|\nabla f(\mathbf{w}_{g,i})\|^2] + 2\mathbb{E}[\langle \nabla f(\mathbf{w}_{g,i}), \nabla g(\mathbf{w}_{g,i}) - \nabla f(\mathbf{w}_{g,i})\rangle] + \mathbb{E}[\|\nabla g(\mathbf{w}_{g,i}) - \nabla f(\mathbf{w}_{g,i})\|^2] \\ &\leq \mathbb{E}[\|\nabla f(\mathbf{w}_{g,i})\|^2] + 2\mathbb{E}[\langle \nabla f(\mathbf{w}_{g,i}), \nabla g(\mathbf{w}_{g,i}) - \nabla f(\mathbf{w}_{g,i})\rangle] + \zeta^2\end{aligned}$$

we obtain:

$$\begin{aligned}(15) \leq &\Big(\mathbb{E}[\|\mathbf{w}_{g,i} - \mathbf{w}_f^*\|^2] - \eta_2\nu\mathbb{E}[\|\mathbf{w}_{g,i} - \mathbf{w}_f^*\|^2] \\ &+ 2\eta_2\mathbb{E}[\langle \mathbf{w}_{g,i} - \mathbf{w}_f^*, \nabla g(\mathbf{w}_{g,i}) - \nabla f(\mathbf{w}_{g,i})\rangle]\Big) \\ &+ \Big(- \eta_2^2\mathbb{E}[\|\nabla f(\mathbf{w}_{g,i})\|^2] + 2\eta_2^2\mathbb{E}[\langle \nabla f(\mathbf{w}_{g,i}), \nabla g(\mathbf{w}_{g,i}) - \nabla f(\mathbf{w}_{g,i})\rangle]\Big) \\ &+ \eta_2^2\zeta^2 + \eta_2^2 m\sigma^2 \\ \leq &(1 - \frac{\eta_2\nu}{2})\mathbb{E}[\|\mathbf{w}_{g,i} - \mathbf{w}_f^*\|^2 + \frac{2\eta_2\zeta^2}{\nu} + 2\eta_2^2\zeta^2 + \eta_2^2 m\sigma^2 \qquad (16) \\ = &(1 - \frac{\eta_2\nu}{2})\mathbb{E}[\|\mathbf{w}_{g,i} - \mathbf{w}_f^*\|^2] + 2\eta_2\zeta^2\left(\frac{1}{\nu} + \eta_2\right) + \eta_2^2 m\sigma^2, \qquad (17)\end{aligned}$$

where (16) holds due to the following two claims:

Claim B.1.

$$\mathbb{E}[\langle \mathbf{w}_{g,i} - \mathbf{w}_f^*, \nabla f(\mathbf{w}_{g,i}) - \nabla g(\mathbf{w}_{g,i})\rangle] \leq \frac{\nu}{4}\mathbb{E}[\|\mathbf{w}_{g,i} - \mathbf{w}_f^*\|^2] + \frac{\zeta^2}{\nu}$$

Proof.

$$\begin{aligned}\mathbb{E}[\langle \mathbf{w}_{g,i} - \mathbf{w}_f^*, \nabla f(\mathbf{w}_{g,i}) - \nabla g(\mathbf{w}_{g,i})\rangle] &= \mathbb{E}[\langle \sqrt{\frac{\nu}{2}}(\mathbf{w}_{g,i} - \mathbf{w}_f^*), \sqrt{\frac{2}{\nu}}(\nabla f(\mathbf{w}_{g,i}) - \nabla g(\mathbf{w}_{g,i}))\rangle] \\ &\leq \frac{\nu}{4}\mathbb{E}[\|\mathbf{w}_{g,i} - \mathbf{w}_f^*\|^2] + \frac{1}{\nu}\mathbb{E}[\|\nabla f(\mathbf{w}_{g,i}) - \nabla g(\mathbf{w}_{g,i})\|^2]\rangle] \\ &\leq \frac{\nu}{4}\mathbb{E}[\|\mathbf{w}_{g,i} - \mathbf{w}_f^*\|^2] + \frac{\zeta^2}{\nu},\end{aligned}$$

where the final inequality holds due to Assumption 3.1. □

Claim B.2.

$$-\mathbb{E}[\|\nabla f(\mathbf{w}_{g,i})\|^2] + 2\mathbb{E}[\langle \nabla f(\mathbf{w}_{g,i}), \nabla g(\mathbf{w}_{g,i}) - \nabla f(\mathbf{w}_{g,i})\rangle] \leq \zeta^2$$

Proof.

$$\begin{aligned}&- \mathbb{E}[\|\nabla f(\mathbf{w}_{g,i})\|^2] + 2\mathbb{E}[\langle \nabla f(\mathbf{w}_{g,i}), \nabla g(\mathbf{w}_{g,i}) - \nabla f(\mathbf{w}_{g,i})\rangle] \\ &\leq -\mathbb{E}[\|\nabla f(\mathbf{w}_{g,i})\|^2] + \mathbb{E}[\|\nabla f(\mathbf{w}_{g,i})\|^2] + \mathbb{E}[\|\nabla g(\mathbf{w}_{g,i}) - \nabla f(\mathbf{w}_{g,i})\|^2] \\ &\leq \zeta^2,\end{aligned}$$

where the final inequality holds due to Assumption 3.1. □

The analysis from above culminating with (17) yields the following bound on $\mathbb{E}[\|\mathbf{w}_{g,i+1} - \mathbf{w}_f^*\|^2]$ in terms of $\mathbb{E}[\|\mathbf{w}_{g,i} - \mathbf{w}_f^*\|^2]$:

$$\mathbb{E}[\|\mathbf{w}_{g,i+1} - \mathbf{w}_f^*\|^2] \leq (1 - \frac{\eta_2\nu}{2})\mathbb{E}[\|\mathbf{w}_{g,i} - \mathbf{w}_f^*\|^2] + 2\eta_2\zeta^2\left(\frac{1}{\nu} + \eta_2\right) + \eta_2^2 m\sigma^2. \qquad (18)$$

Applying the bound in (18) iteratively and taking expectation with respect to $\mathbf{w}_T, \mathbf{w}_{T-1}, \ldots, \mathbf{w}_0$ yields the desired bound:

$$\begin{aligned}\mathbb{E}[\|\mathbf{w}_{g,T} - \mathbf{w}_f^*\|^2] &\leq \left(1 - \frac{\eta_2\nu}{2}\right)^T \|\mathbf{w}_{g,0} - \mathbf{w}_f^*\|^2 + \left(2\eta_2\zeta^2\left(\frac{1}{\nu} + \eta_2\right) + \eta_2^2 m\sigma^2\right)\sum_{i=0}^{T}\left(1 - \frac{\eta_2\nu}{2}\right)^i \\ &\leq \left(1 - \frac{\eta_2\nu}{2}\right)^T \|\mathbf{w}_{g,0} - \mathbf{w}_f^*\|^2 + \left(2\eta_2\zeta^2\left(\frac{1}{\nu} + \eta_2\right) + \eta_2^2 m\sigma^2\right)\frac{2}{\eta_2\nu} \\ &= \left(1 - \frac{\eta_2\nu}{2}\right)^T \|\mathbf{w}_{g,0} - \mathbf{w}_f^*\|^2 + \frac{4\zeta^2}{\nu}\left(\frac{1}{\nu} + \eta_2\right) + \frac{2\eta_2 m\sigma^2}{\nu}.\end{aligned}$$

□

C. Proof of Theorem 5.1 and Missing Analyses from Section 5

We begin by restating Theorem 5.1.

Theorem 5.1. *[Differential Privacy of Algorithm 3] Fix constants $e_f \in [0, \phi'_{max}], e_B \geq 0, \kappa \in (0,1)$. Let $\zeta_f = e_f\sqrt{m}$, $d \geq \frac{\|\nabla f(\mathbf{0})\|}{\sqrt{m}}$, $\mathsf{c}_\delta := \sqrt{2\ln(\frac{3T}{\delta})}$, and $R := \sqrt{(1-\kappa)\Theta} + \eta_3\left(\phi'_{max}\sqrt{m} + \zeta_f + 2\lambda e_B\sqrt{\Theta} + (\sqrt{m} + \mathsf{c}_\delta)\sigma\right)$. Let m_P, (resp. M_P) be the minimum (resp. maximum) value of P_κ on the interval $[\Theta - R^2, \kappa\Theta]$. If the following hold:*

- *For $y \in \{0,1\}$, $\tilde{p}'(z,y)$ has error at most e_f with respect to $\phi'(z,y)$ on the interval $z \in [-\sqrt{m}R, \sqrt{m}R]$.*
- *P_κ has error at most e_B with respect to F_κ on the interval $[\kappa\Theta, \Theta]$. and is monotonically* decreasing *on the interval $[\Theta - R^2, \kappa\Theta]$.*
- *$\mathsf{m}_P \geq 0$ and η_3 satisfies the following inequality:*

$$\eta_3 \leq \min\left\{\frac{\kappa\Theta}{\lambda}, \frac{1}{\lambda(\mathsf{M}_P + \mathsf{m}_P) + \frac{(\phi'_{max} - d)\sqrt{m}}{2\sqrt{(1-\kappa)\Theta}}}\right\}. \tag{1}$$

- *κ satisfies the following inequality:*

$$\sqrt{(1-\kappa)\Theta} \geq \frac{-B + \sqrt{B^2 - 4AC}}{2A}, \tag{2}$$

where $\alpha = 2\eta_3\lambda\mathsf{m}_P$, $A = (2\alpha - \alpha^2)$, $B = -2\eta_3((1-\alpha)(d\sqrt{m} + \mathsf{c}_\delta \cdot \sigma) + \zeta_f)$, and $C = -\eta_3^2((\sqrt{m}\phi'_{max} + (\sqrt{m} + \mathsf{c}_\delta)\sigma)^2 - \zeta_f^2)$.

then the model, $\mathbf{w}_T$, outputted by Algorithm 3 achieves (ϵ, δ)-differential privacy.

To prove Theorem 5.1, we first bound the maximum magnitude of the weights, R, encountered during gradient descent in Lemma 5.5 (see Appendix C.1 for the proof). We then use this to define the approximation interval $[-\sqrt{m}R, \sqrt{m}R]$ for approximations $\tilde{p}'_0$ and $\tilde{p}'_1$ relative to target functions ϕ'_0 and ϕ'_1. This then allows us to bound the sensitivity in Appendix C.2. Finally, by employing standard DP analysis to set the noise variance σ, we achieve the full DP guarantee.

C.1. Proof of Lemma 5.5

We begin by restating Lemma 5.5.

Lemma 5.5. *(Bounding magnitude of weights) Let $\mathbf{w}_i$ denote the i-th iterate of Algorithm 3. Under the parameter conditions of Theorem 5.1, with probability at least $1 - \frac{2\delta}{3}$, the following holds for all $i \in \{0, \ldots, T\}$:*

$$\begin{aligned}\|\mathbf{w}_i\| \leq &\sqrt{(1-\kappa)\Theta} \\ &+ \eta_3\left(\phi'_{max}\sqrt{m} + \zeta_f + 2\lambda e_B\sqrt{\Theta} + (\sqrt{m} + \mathsf{c}_\delta)\sigma\right).\end{aligned}$$

We prove the lemma by induction on the iteration index i. The base case is immediate, since Algorithm 3 is initialized at $\mathbf{w}_0 = \mathbf{0}$.

For the inductive step, we analyze a single update of the form

$$\mathbf{w}_{i+1} = \mathbf{w}_i - \eta_3\Big(2\lambda P_\kappa(\Theta - \|\mathbf{w}_i\|^2)\mathbf{w}_i + \frac{1}{N}\sum_{(\mathbf{x},y)\in D} \tilde{p}'(\langle \mathbf{w}_i, \mathbf{x}\rangle, y)\mathbf{x} + \boldsymbol{\chi}_i\Big).$$

The update decomposes naturally into three components. The first term corresponds to the gradient of the barrier function; since $P_\kappa(x)$ approximates $1/x$, this term is approximately $\frac{2\lambda}{\Theta - \|\mathbf{w}_i\|^2}\mathbf{w}_i$ and acts to reduce the norm of $\mathbf{w}_i$ as it approaches the boundary. The second term approximates the gradient of the original objective f. The third term accounts for the injected Gaussian noise, whose magnitude we bound over all T iterations with high probability.

As $\|\mathbf{w}_i\|^2$ approaches Θ, the barrier term increasingly pulls the iterate inward, while the data-dependent gradient and noise may push it outward. We therefore split the analysis into two cases.

In the first case, when $\|\mathbf{w}_i\| \le \sqrt{(1-\kappa)\Theta}$, we choose parameters so that the barrier term cannot increase the norm. Bounding the contributions of the objective gradient, approximation error, and noise then yields

$$\|\mathbf{w}_{i+1}\| \le \sqrt{(1-\kappa)\Theta} + \eta_3\big(\phi'_{\max}\sqrt{m} + \zeta_f + 2\lambda e_B\sqrt{\Theta} + (\sqrt{m} + \mathsf{c}_\delta)\sigma\big).$$

In the second case, when $\|\mathbf{w}_i\| \ge \sqrt{(1-\kappa)\Theta}$, we show that the inward pull of the barrier term dominates any outward contribution from the objective gradient and noise. This requires a refined decomposition of these terms into components parallel and orthogonal to $\mathbf{w}_i$. Under the parameter constraints in (1) and (2), this analysis implies $\|\mathbf{w}_{i+1}\| \le \|\mathbf{w}_i\|$.

Combining both cases with the inductive hypothesis yields the desired bound on $\|\mathbf{w}_i\|$ for all iterations, completing the proof.

We now proceed with the formal proof

Proof. We first note that due to the setting of c_δ, by standard analysis, with probability $1 - \frac{2\delta}{3}$, $\|\boldsymbol{\chi}_i\| \le (\sqrt{m} + \mathsf{c}_\delta)\sigma$ and $\frac{\langle \boldsymbol{\chi}_i, \mathbf{w}_i\rangle}{\|\mathbf{w}_i\|} \le \mathsf{c}_\delta \cdot \sigma$, for all $i \in \{0, \dots, T\}$.

Assuming the above bounds hold for all $i \in \{0, \dots, T\}$, we proceed to argue by induction:

Base case: $\mathbf{w}_0$ is initialized to the all 0 vector in so $\|\mathbf{w}_0\| = 0$.

Inductive case. Assume that $\|\mathbf{w}_i\| \le \sqrt{(1-\kappa)\Theta} + \eta_3\left(\phi'_{max}\sqrt{m} + \zeta_f + 2\lambda e_B\sqrt{\Theta} + (\sqrt{m} + \mathsf{c}_\delta)\sigma\right)$.

We further split the inductive case into two sub-cases:

Sub-Case 1: Assume that $\|\mathbf{w}_i\| \le \sqrt{(1-\kappa)\Theta}$.

We have

$$\begin{aligned}
\|\mathbf{w}_{i+1}\| &= \|\mathbf{w}_i - \eta_3 \sum_{(\mathbf{x},y)\in D} p'(\langle \mathbf{w}_i, \mathbf{x}\rangle)\mathbf{x} - 2\tilde{p}'(z)\eta_3\lambda P_\kappa(\Theta - \|\mathbf{w}_i\|^2)\mathbf{w}_i - \eta_3\boldsymbol{\chi}_i\| \\
&\le \|\mathbf{w}_i - \eta_3\nabla f_\kappa^{+B}(\mathbf{w}_i)\| + \eta_3\zeta_f + 2\eta_3\lambda e_B\sqrt{(1-\kappa)\Theta} + \eta_3\|\boldsymbol{\chi}_i\| \\
&\le \|\mathbf{w}_i - 2\eta_3\lambda F_\kappa(\Theta - \|\mathbf{w}_i\|^2)\mathbf{w}_i\| + \eta_3\phi'_{max}\sqrt{m} + \eta_3\zeta_f + 2\eta_3\lambda e_B\sqrt{(1-\kappa)\Theta} + \eta_3\|\boldsymbol{\chi}_i\| \\
&\le \|\mathbf{w}_i\| + \eta_3\phi'_{max}\sqrt{m} + \eta_3\zeta_f + 2\eta_3\lambda e_B\sqrt{(1-\kappa)\Theta} + \eta_3\|\boldsymbol{\chi}_i\| \qquad (19) \\
&\le \|\mathbf{w}_i\| + \eta_3\left(\phi'_{max}\sqrt{m} + \zeta_f + 2\lambda e_B\sqrt{\Theta} + (\sqrt{m} + \mathsf{c}_\delta)\sigma\right),
\end{aligned}$$

where (19) holds since in Sub-Case 1, $F_\kappa(\Theta - \|\mathbf{w}_i\|^2) \le \frac{1}{\kappa\Theta}$ and since $\eta_3 \le \frac{\kappa\Theta}{\lambda}$, we have that $|1 - 2\eta_3\lambda F_\kappa(\Theta - \|\mathbf{w}_i\|^2)| \le 1$.

Sub-Case 2: Assume that $\|\mathbf{w}_i\| \ge \sqrt{(1-\kappa)\Theta}$.

First recall that $\|\nabla f(\mathbf{w}_i)\| \le \phi'_{max} \cdot \sqrt{m}$. Further, since f is convex, we have that $\langle \nabla f(\mathbf{w}_i) - \nabla f(\mathbf{0}), \mathbf{w}_i\rangle \ge 0$. This implies that

$$\langle \nabla f(\mathbf{w}_i), \mathbf{w}_i \rangle \geq \langle \nabla f(\mathbf{0}), \mathbf{w}_i \rangle \geq -\|\nabla f(\mathbf{0})\| \|\mathbf{w}_i\|.$$

We can write

$$\nabla f(\mathbf{w}_i) = a\widetilde{\mathbf{w}} + b\widehat{\mathbf{w}},$$

where $\widetilde{\mathbf{w}} = \frac{\mathbf{w}_i}{\|\mathbf{w}_i\|}$, $\widehat{\mathbf{w}}$ is a unit vector orthogonal to $\widetilde{\mathbf{w}}$, and

$$-\|\nabla f(\mathbf{0})\| \leq a \leq \phi'_{max} \cdot \sqrt{m} \quad \text{and} \quad c := \sqrt{a^2 + b^2} \leq \phi'_{max} \cdot \sqrt{m}. \tag{20}$$

Further, since we assume

$$\frac{|\langle \boldsymbol{\chi}_i, \mathbf{w}_i \rangle|}{\|\mathbf{w}_i\|} \leq \mathsf{c}_\delta \cdot \sigma,$$

$\boldsymbol{\chi}_i$ can be written as:

$$\boldsymbol{\chi}_i = a'\widetilde{\mathbf{w}} + b'\overline{\mathbf{w}},$$

where $\overline{\mathbf{w}}$ is a unit vector orthogonal to $\widetilde{\mathbf{w}}$ and

$$|a'| \leq \mathsf{c}_\delta \cdot \sigma \quad \text{and} \quad c' := \sqrt{(a')^2 + (b')^2} \leq (\sqrt{m} + \mathsf{c}_\delta)\sigma. \tag{21}$$

We will use the following fact later in the proof:

$$(a + a')^2 + (|b| + |b'|)^2 \leq (c + c')^2. \tag{22}$$

This can be seen since:

$$\begin{aligned}
(a + a')^2 + (|b| + |b'|)^2 &= a^2 + b^2 + (a')^2 + (b')^2 + 2aa' + 2|b||b'| \\
&= a^2 + b^2 + (a')^2 + (b')^2 + 2\langle (a, |b|), (a', |b'|) \rangle \\
&\leq a^2 + b^2 + (a')^2 + (b')^2 + 2\|(a, |b|)\| \cdot \|(a', |b'|)\| \\
&= c^2 + c'^2 + 2c \cdot c' \\
&= (c + c')^2.
\end{aligned}$$

We thus have

$$\begin{aligned}
&\|\mathbf{w}_{i+1}\| = \|\mathbf{w}_i - \eta_3 \sum_{(\mathbf{x},y)\in D} p'(\langle \mathbf{w}_i, \mathbf{x} \rangle)\mathbf{x} - 2\tilde{p}'(z)\eta_3\lambda P_\kappa(\Theta - \|\mathbf{w}_i\|^2)\mathbf{w}_i - \eta_3\boldsymbol{\chi}_i\| \\
&\leq \|\mathbf{w}_i - 2\tilde{p}'(z)\eta_3\lambda P_\kappa(\Theta - \|\mathbf{w}_i\|^2)\mathbf{w}_i - \eta_3\nabla f(\mathbf{w}_i) - \eta_3\boldsymbol{\chi}_i\| + \eta_3\zeta_f \\
&= \sqrt{\|\mathbf{w}_i - 2\tilde{p}'(z)\eta_3\lambda P_\kappa(\Theta - \|\mathbf{w}_i\|^2)\mathbf{w}_i - \eta_3 \cdot a\widetilde{\mathbf{w}} - \eta_3 \cdot a'\widetilde{\mathbf{w}}\|^2 + \eta_3^2\|b\widehat{\mathbf{w}} + b'\overline{\mathbf{w}}\|^2}} + \eta_3\zeta_f \\
&\leq \sqrt{\|\mathbf{w}_i - 2\tilde{p}'(z)\eta_3\lambda P_\kappa(\Theta - \|\mathbf{w}_i\|^2)\mathbf{w}_i - \eta_3 \cdot a\widetilde{\mathbf{w}} - \eta_3 \cdot a'\widetilde{\mathbf{w}}\|^2 + \eta_3^2(\|b\widehat{\mathbf{w}}\| + \|b'\overline{\mathbf{w}}\|)^2} + \eta_3\zeta_f \\
&\leq \sqrt{\|\mathbf{w}_i - 2\tilde{p}'(z)\eta_3\lambda P_\kappa(\Theta - \|\mathbf{w}_i\|^2)\mathbf{w}_i - \eta_3 \cdot a\widetilde{\mathbf{w}} - \eta_3 \cdot a'\widetilde{\mathbf{w}}\|^2 + \eta_3^2(|b| + |b'|)^2} + \eta_3\zeta_f \\
&= \sqrt{\left((1 - 2\tilde{p}'(z)\eta_3\lambda P_\kappa(\Theta - \|\mathbf{w}_i\|^2))\|\mathbf{w}_i\| - \eta_3(a + a')\right)^2 + \eta_3^2(|b| + |b'|)^2} + \eta_3\zeta_f \\
&= \sqrt{\left(1 - 2\tilde{p}'(z)\eta_3\lambda P_\kappa(\Theta - \|\mathbf{w}_i\|^2)\right)^2\|\mathbf{w}_i\|^2 - 2\eta_3\left(1 - 2\tilde{p}'(z)\eta_3\lambda P_\kappa(\Theta - \|\mathbf{w}_i\|^2)\right)\|\mathbf{w}_i\|(a + a') + \eta_3^2(a + a')^2 + \eta_3^2(|b| + |b'|)^2} + \eta_3\zeta \\
&\leq \sqrt{\left(1 - 2\tilde{p}'(z)\eta_3\lambda P_\kappa(\Theta - \|\mathbf{w}_i\|^2)\right)^2\|\mathbf{w}_i\|^2 - 2\eta_3\left(1 - 2\tilde{p}'(z)\eta_3\lambda P_\kappa(\Theta - \|\mathbf{w}_i\|^2)\right)\|\mathbf{w}_i\|(a + a') + \eta_3^2(c + c')^2} + \eta_3\zeta_f \qquad (23) \\
&\leq \sqrt{\left(1 - 2\tilde{p}'(z)\eta_3\lambda P_\kappa(\Theta - \|\mathbf{w}_i\|^2)\right)^2\|\mathbf{w}_i\|^2 - 2\eta_3\left(1 - 2\tilde{p}'(z)\eta_3\lambda P_\kappa(\Theta - \|\mathbf{w}_i\|^2)\right)\|\mathbf{w}_i\|(a + a') + \eta_3^2 \cdot (\sqrt{m}\phi'_{max} + (\sqrt{m} + \mathsf{c}_\delta)\sigma)^2 + } \qquad (24)
\end{aligned}$$

where (23) follows from (22) and (24) follows from the bounds on c, c' given in (20) and (21).

We now aim to upperbound the quantity

$$\left(1-2\tilde{p}'(z)\eta_3\lambda P_\kappa(\Theta-\|\mathbf{w}_i\|^2)\right)^2\|\mathbf{w}_i\|^2-2\eta_3\left(1-2\tilde{p}'(z)\eta_3\lambda P_\kappa(\Theta-\|\mathbf{w}_i\|^2)\right)\|\mathbf{w}_i\|(a+a') \tag{25}$$

from (24).

First, for every fixed value $\|\mathbf{w}_i\|$ and for every fixed value $V := 2\tilde{p}'(z)\eta_3\lambda P_\kappa(\Theta-\|\mathbf{w}_i\|^2)$ such that $1-V$ is positive, (25) is maximized by taking $(a+a')$ to be as small a *negative value* as possible, which by (20), is $(a+a') = -(d\sqrt{m}+\mathsf{c}_\delta\cdot\sigma)$. Substituting this into (25) we obtain:

$$(1-V)^2\|\mathbf{w}_i\|^2+2\eta_3(1-V))\|\mathbf{w}_i\|(d\sqrt{m}+\mathsf{c}_\delta\cdot\sigma). \tag{26}$$

Further, for every fixed value $\|\mathbf{w}_i\|$ and for all values V, such that $(1-V)$ is positive, (27) is maximized by taking V as small as possible, which in Sub-Case 2, is $V = 2\tilde{p}'(z)\eta_3\lambda\mathsf{m}_P$, since $\eta_3 \le \frac{1}{2\mathsf{m}_P}$. So when $(1-V)$ is positive, this yields

$$\mathsf{Pos} := (1-2\tilde{p}'(z)\eta_3\lambda\mathsf{m}_P)^2\|\mathbf{w}_i\|^2+2\eta_3(1-2\tilde{p}'(z)\eta_3\lambda\mathsf{m}_P))\|\mathbf{w}_i\|(d\sqrt{m}+\mathsf{c}_\delta\cdot\sigma).$$

Second, for every fixed value $\|\mathbf{w}_i\|$ and for every fixed value $V := 2\tilde{p}'(z)\eta_3\lambda P_\kappa(\Theta-\|\mathbf{w}_i\|^2)$ such that $1-V$ is negative, (25) is maximized by taking $(a+a')$ to be as large a *positive value* as possible, which by (20), is $(a+a') = (\phi'_{max}\cdot\sqrt{m}+\mathsf{c}_\delta\cdot\sigma)$. Substituting this into (25) we obtain:

$$(1-V)^2\|\mathbf{w}_i\|^2-2\eta_3(1-V))\|\mathbf{w}_i\|(\phi'_{max}\cdot\sqrt{m}+\mathsf{c}_\delta\cdot\sigma). \tag{27}$$

Further, for every fixed value $\|\mathbf{w}_i\|$ and for all values V, such that $(1-V)$ is negative, (27) is maximized by taking V as large as possible, which in Sub-Case 2, is $V = 2\tilde{p}'(z)\eta_3\lambda\mathsf{M}_P$. So when $(1-V)$ is negative, this yields

$$\mathsf{Neg} := (1-2\tilde{p}'(z)\eta_3\lambda\mathsf{M}_P)^2\|\mathbf{w}_i\|^2-2\eta_3(1-2\tilde{p}'(z)\eta_3\lambda\mathsf{M}_P))\|\mathbf{w}_i\|(\phi'_{max}\cdot\sqrt{m}+\mathsf{c}_\delta\cdot\sigma).$$

Thus,

$$(24) \le \sqrt{\max\{\mathsf{Pos},\mathsf{Neg}\}+\eta_3^2\cdot(\sqrt{m}\phi'_{max}+(\sqrt{m}+\mathsf{c}_\delta)\sigma)^2}+\eta_3\zeta_f.$$

Finally, we show that $\mathsf{Neg} \le \mathsf{Pos}$. Since $\phi'_{max} \ge d$, this is implied by

$$\mathsf{Neg}+\eta_3^2(\phi'_{max}\cdot\sqrt{m}+\mathsf{c}_\delta\cdot\sigma)^2 \le \mathsf{Pos}+\eta_3^2(d\sqrt{m}+\mathsf{c}_\delta\cdot\sigma)^2,$$

or equivalently

$$\left((1-2\eta_3\lambda\mathsf{M}_P)\|\mathbf{w}_i\|-\eta_3(\phi'_{max}\cdot\sqrt{m}+\mathsf{c}_\delta\cdot\sigma)\right)^2 \le \left((1-2\eta_3\lambda\mathsf{m}_P)\|\mathbf{w}_i\|+\eta_3(d\sqrt{m}+\mathsf{c}_\delta\cdot\sigma)\right)^2. \tag{28}$$

(28), in turn, is implied by

$$(-1+2\eta_3\lambda\mathsf{M}_P)\|\mathbf{w}_i\|+\eta_3(\phi'_{max}\cdot\sqrt{m}+\mathsf{c}_\delta\cdot\sigma) \le (1-2\eta_3\lambda\mathsf{m}_P)\|\mathbf{w}_i\|+\eta_3(d\sqrt{m}+\mathsf{c}_\delta\cdot\sigma).$$

This is implied by

$$\eta_3 \le \frac{1}{\lambda(\mathsf{M}_P+\mathsf{m}_P)+\frac{(\phi'_{max}-d)\sqrt{m}}{2\sqrt{(1-\kappa)\Theta}}},$$

which is the restriction we place on η_3 in (1).

Thus, we have that

$$\|\mathbf{w}_{i+1}\| \le \sqrt{(1-2\tilde{p}'(z)\eta_3\lambda\mathsf{m}_P)^2\|\mathbf{w}_i\|^2+2\eta_3(1-2\tilde{p}'(z)\eta_3\lambda\mathsf{m}_P))\|\mathbf{w}_i\|(d\sqrt{m}+\mathsf{c}_\delta\cdot\sigma)+\eta_3^2\cdot(\sqrt{m}\phi'_{max}+(\sqrt{m}+\mathsf{c}_\delta)\sigma)^2}+\eta_3\zeta_f.$$

We next argue that

$$\sqrt{(1-2\tilde{p}'(z)\eta_3\lambda\mathsf{m}_P)^2\|\mathbf{w}_i\|^2+2\eta_3(1-2\tilde{p}'(z)\eta_3\lambda\mathsf{m}_P))\|\mathbf{w}_i\|(d\sqrt{m}+\mathsf{c}_\delta\cdot\sigma)+\eta_3^2\cdot(\sqrt{m}\phi'_{max}+(\sqrt{m}+\mathsf{c}_\delta)\sigma)^2}+\eta_3\zeta_f \le \|\mathbf{w}_i\|,$$

which implies that $\|\mathbf{w}_{i+1}\| \leq \|\mathbf{w}_i\|$.

Since $\alpha = 2\eta_3\lambda \mathsf{m}_P$ is positive and less than 1, the inequality is satisfied as long as

$$\|\mathbf{w}_i\| \geq \frac{-B + \sqrt{B^2 - 4AC}}{2A},$$

where $A = (2\alpha - \alpha^2)$, $B = -2\eta_3((1-\alpha)(d\sqrt{m} + \mathsf{c}_\delta \cdot \sigma) + \zeta_f)$, and $C = -\eta_3^2((\sqrt{m}\phi'_{max} + (\sqrt{m} + \mathsf{c}_\delta)\sigma)^2 - \zeta_f^2)$. Since we are in Sub-Case 2, $\|\mathbf{w}_i\| \geq \sqrt{(1-\kappa)\Theta}$. Recall that we set κ so that:

$$\sqrt{(1-\kappa)\Theta} \geq \frac{-B + \sqrt{B^2 - 4AC}}{2A}.$$

Thus, the inequality is indeed satisfied in Sub-Case 2. By the inductive hypothesis this implies that $\|\mathbf{w}_{i+1}\| \leq \|\mathbf{w}_i\| \leq \sqrt{(1-\kappa)\Theta} + \eta_3\left(\phi'_{max}\sqrt{m} + \zeta_f + 2\lambda e_B\sqrt{\Theta} + (\sqrt{m} + \mathsf{c}_\delta)\sqrt{m}\sigma\right)$. This concludes the proof of Lemma 5.5. □

C.2. Bounding the Sensitivity of Gradient Updates in Algorithm 3

Since $P_\kappa(\mathbf{w}_i)$ is data-independent, we can upper bound the ℓ_2 sensitivity of $\nabla g(\mathbf{w}_i)$ by $\frac{\Delta_2}{N}$, where

$$\begin{aligned}
\Delta_2 = \|\nabla g_{(\mathbf{x},y)}(\mathbf{w}_i) - \nabla g_{(\mathbf{x}',y')}(\mathbf{w}_i)\| &= \|\tilde{p}'(\langle \mathbf{w}_i, \mathbf{x}\rangle, y)\cdot \mathbf{x} - \tilde{p}'(\langle \mathbf{w}_i, \mathbf{x}'\rangle, y')\cdot \mathbf{x}'\| \\
&\leq \|\tilde{p}'(\langle \mathbf{w}_i, \mathbf{x}\rangle, y)\cdot \mathbf{x}\| + \|\tilde{p}'(\langle \mathbf{w}_i, \mathbf{x}'\rangle, y')\cdot \mathbf{x}'\| \\
&= |\tilde{p}'(\langle \mathbf{w}_i, \mathbf{x}\rangle, y)|\cdot\|\mathbf{x}\| + |\tilde{p}'(\langle \mathbf{w}_i, \mathbf{x}'\rangle, y')|\cdot\|\mathbf{x}'\| \\
&\leq (|\phi'(\langle \mathbf{w}_i, \mathbf{x}\rangle, y)| + e_f)\cdot\|\mathbf{x}\| + (|\phi'(\langle \mathbf{w}_i, \mathbf{x}'\rangle, y')| + e_f)\cdot\|\mathbf{x}'\| \qquad (29) \\
&\leq 2(\phi'_{max} + e_f)\sqrt{m}.
\end{aligned}$$

Crucially, (29) follows since, via Lemma 5.5, $\langle \mathbf{w}_i, \mathbf{x}\rangle$ is guaranteed to be contained in the interval $[-\sqrt{m}R, \sqrt{m}R]$ and $\tilde{p}'$ is guaranteed to have error at most e_f with respect to ϕ' on this interval.

Thus we finally obtain $\Delta_2 \leq 2(\phi'_{max} + e_f)\sqrt{m}$, where $\phi'_{max} = \sup\{|\phi'(a,b)| : a \in (-\infty, \infty), b \in \{0,1\}\}$.

C.3. Choosing polynomials and parameters for Algorithm 3

Choosing parameters is delicate since the constraint on κ in (2) depends on the setting of η_3 but the constraint on η_3 in (1) depends on the settings of κ and P_κ.

If one just wants to find *some* feasible setting of parameters, this can be done as follows: First, the constraint (2) on κ given in Theorem 5.1 is implied by the following constraint (assuming $\alpha \leq 1$, which occurs when $\eta_3 \leq \frac{1}{2\lambda \mathsf{m}_P}$ and is implied by (1)):

$$\sqrt{(1-\kappa)\Theta} \geq \frac{2(d\sqrt{m} + \mathsf{c}_\delta\cdot\sigma + \zeta_f) + \sqrt{8(d\sqrt{m} + \mathsf{c}_\delta\sigma + \zeta_f)^2 + 4((\sqrt{m}\phi'_{max} + (\sqrt{m} + \mathsf{c}_\delta)\sigma)^2 - \zeta_f^2)}}{2\lambda \mathsf{m}_P}. \qquad (30)$$

Note that this constraint is now independent of η_3. Furthermore, the right hand side of (30) decreases as κ decreases. This is because the value m_P, which appears in the denominator of the right hand side, is equal to the value of $P_\kappa(\kappa\Theta)$ (since P_κ will be chosen such that it is decreasing on the interval $[\Theta - R^2, \kappa\Theta]$. By the approximation error bound, we can lower bound $P_\kappa(\kappa\Theta) \geq \frac{1}{\kappa\Theta} - e_B$ and this quantity increases as κ decreases. On the other hand, the left hand side of (30) increases as κ decreases. Thus, there must be a setting of κ that satisfies the equation, while all remaining parameters are fixed.

Once κ is fixed, we can determine P_κ as follows: First, we find a polynomial approximation, $\tilde{P}_\kappa$, of $1/x$ on the interval $[\frac{\kappa\Theta}{2}, \Theta]$ that preserves monotonicity and has error at most $\frac{e_B}{2}$. Such an approximation can be found by taking a sufficiently high degree polynomial using the works of (DeVore, 1977; DeVore and Yu, 1985). Since η_3 is not yet determined, we assume $\eta_3 \leq \frac{1}{2\beta_f}$ (which is the estimated learning rate needed for convergence).

To construct P_κ, we add an adjustment $h(x)$ to $\tilde{P}'_\kappa(x)$ and then set $P_\kappa := \tilde{P}_\kappa(x) + \tilde{h}(x)$ to be an antiderivative of $\tilde{P}'(x) + h(x)$. We choose $h(x)$ so that $\tilde{P}'_\kappa(x) + h(x)$ is negative on the interval $[\Theta - \tilde{R}^2, \frac{\kappa\Theta}{2}]$. Further, we choose an $h(x)$

that is non-positive on $[\Theta - \tilde{R}^2, \Theta]$, $\tilde{P}'(x) + h(x)$, so that $\tilde{P}'(x) + h(x)$ is negative on $[\frac{\kappa\Theta}{2}, \kappa\Theta]$. Taken together, these ensure that P_κ is monotonically decreasing on the interval $[\Theta - \tilde{R}^2, \kappa\Theta]$. To ensure that the error between P_κ and $1/x$ does not increase by more than $\frac{e_B}{2}$ beyond the error of at most $\frac{e_B}{2}$ between P_κ and $1/x$ on the interval $[\kappa\Theta, \Theta]$, we choose $h(x)$ such that $|\tilde{h}(\kappa\Theta)|$ is decreasing on $[\kappa\Theta, \Theta]$ and such that $|\tilde{h}(\kappa\Theta)| \leq \frac{e_B}{2}$. Details follow.

Let $M := \max\{|\Theta - \tilde{R}^2|, (1-\kappa)\Theta\}$. Let $q \geq 0$ be a sufficiently large integer such that for all $x \in [\Theta - \tilde{R}^2, \frac{\kappa\Theta}{2}]$,

$$\tilde{P}'_\kappa(x) - \frac{e_B(q+1)}{2M}\left(1 - \frac{x - \kappa\Theta}{M}\right)^q \leq -1. \tag{31}$$

Define

$$h(x) := -\frac{e_B(q+1)}{2M}\left(1 - \frac{x - \kappa\Theta}{M}\right)^q.$$

Note that (31) implies that $\tilde{P}'_\kappa(x) + h(x) < 0$ on the interval $[\Theta - \tilde{R}^2, \frac{\kappa\Theta}{2}]$. The fact that $\tilde{P}_\kappa$ is monotonically decreasing on $[\frac{\kappa\Theta}{2}, \Theta]$ and $h(x)$ is non-positive on $[\frac{\kappa\Theta}{2}, \Theta]$ implies that $\tilde{P}'_\kappa(x) + h(x) < 0$ on the interval $[\frac{\kappa\Theta}{2}, \Theta]$. Let

$$\tilde{h}(x) := \frac{e_B}{2}\left(1 - \frac{x - \kappa\Theta}{M}\right)^{q+1}$$

be an antiderivative of $h(x)$. Then $P_\kappa(x) := \tilde{P}_\kappa(x) + \tilde{h}(x)$ is monotonically decreasing on the interval $[\Theta - \tilde{R}^2, \kappa\Theta]$. Moreover, since $|\tilde{h}(x)|$ is decreasing on the interval $[\kappa\Theta, \Theta]$, the maximum error between P_κ and $\frac{1}{x}$ is at most $\frac{e_B}{2} + |\tilde{h}(\kappa\Theta)| \leq e_B$ on the interval $[\kappa\Theta, \Theta]$.

Now that P_κ is fixed, we set η_3 sufficiently small so that it satisfies (1) and is at most $\frac{1}{2\beta_f}$. Note that this also fixes the value of R.

Finally, approximations p'_0 and p'_1 to ϕ'_0, ϕ'_1 with error at most e_f over the interval $[-R\sqrt{m}, R\sqrt{m}]$ can be found by taking sufficiently high degree using the works of (Bernstein, 1912; Roulier, 1970).

The above always leads to a feasible setting of parameters, but it may not be the best practical choice. In Appendix D.3, we describe a heuristic procedure that leads to better parameters in practice.

C.4. On the Convergence of Algorithm 3

In order to apply Theorem 4.1, we must define F_κ on the entire interval $[\Theta - R^2, \Theta]$, whereas it is currently only defined on $[\kappa\Theta, \Theta]$. Once we do this, we can set B_κ to be an antiderivative of F_κ so that $f_\kappa^{+B}(\mathbf{w}) = f(\mathbf{w}) - \lambda B_\kappa(\Theta - \|\mathbf{w}\|^2)$ is fully defined on the domain $\{\mathbf{w} : \|\mathbf{w}\| \leq R\}$. We then analyze the strong convexity and smoothness of f_κ^{+B} over the domain. Finally, we determine an upperbound on ζ by bounding the norm of the difference between the gradient of f_κ^{+B}:

$$\nabla f_\kappa^{+B}(\mathbf{w}) = \frac{1}{N}\sum_{(\mathbf{x},y)\in D} y\phi'_1(\langle \mathbf{w}, \mathbf{x}\rangle) + (1-y)\phi'_0(\langle \mathbf{w}, \mathbf{x}\rangle) + 2\lambda F_\kappa(\Theta - \|\mathbf{w}\|^2)$$

and the approximate gradient:

$$\frac{1}{N}\sum_{(\mathbf{x},y)\in D} y\tilde{p}'_1(\langle \mathbf{w}, \mathbf{x}\rangle) + (1-y)\tilde{p}'_0(\langle \mathbf{w}, \mathbf{x}\rangle) + 2\lambda P_\kappa(\Theta - \|\mathbf{w}\|^2).$$

Defining F_κ. Recall that $F_\kappa(x) = \frac{1}{x}$ for $x \in [\kappa\Theta, \Theta]$ and that $P_\kappa(x)$ is defined on the entire interval $[\Theta - R^2, \kappa\Theta]$. Recall that we further require P_κ to be monotonically decreasing on the interval $[\Theta - R^2, \kappa\Theta]$. For $x \in [\Theta - R^2, \kappa\Theta]$, we set $F_\kappa(x) := (\frac{1}{\kappa\Theta} - P(\kappa\Theta)) + P_\kappa(x)$.

The Hessian of $-\lambda B_\kappa(\Theta - \|\mathbf{w}\|^2)$. The Hessian of the function $-\lambda B_\kappa(\Theta - \|\mathbf{w}\|^2)$ has the form

$$4\lambda B''(\Theta - \|\mathbf{w}\|^2)\cdot \mathbf{w}\cdot\mathbf{w}^T - 2\lambda B'(\Theta - \|\mathbf{w}\|^2)\cdot \mathbf{I}_m.$$

Since the first term is an outer product, it has a single non-zero eigenvalue which is equal to $4\lambda B''(\Theta - \|\mathbf{w}\|^2)\|\mathbf{w}\|^2$. The second term has m eigenvalues $-2\lambda B'(\Theta - \|\mathbf{w}\|^2)$. Re-written in terms of F_κ, these are equal to $-4\lambda F'_\kappa(\Theta - \|\mathbf{w}\|^2)\|\mathbf{w}\|^2$. and $2\lambda F_\kappa(\Theta - \|\mathbf{w}\|^2)$.

Strong Convexity of f_κ^{+B}. Given the above, the strong convexity parameter of $-\lambda B_\kappa$ is equal to the minimum value of $-4\lambda F'_\kappa(\Theta - \|\mathbf{w}\|^2)\|\mathbf{w}\|^2 + 2\lambda F_\kappa(\Theta - \|\mathbf{w}\|^2)$ over the domain $\{\mathbf{w} : \|\mathbf{w}\|^2 \leq R\}$. Since F_κ is strictly decreasing, $-4\lambda F'_\kappa(\Theta - \|\mathbf{w}\|^2)$ is always non-negative, therefore the lower bound of the first term is 0 and occurs when $\|\mathbf{w}\|^2 = 0$. The second term is also non-negative and the lower bound is $\frac{2\lambda}{\Theta}$ and occurs when $\|\mathbf{w}\|^2 = 0$. Thus, the total strong convexity parameter of f_κ^{+B} is equal to $\mu_{f_\kappa^{+B}} = \mu_f + \frac{2\lambda}{\Theta}$.

Smoothness of f_κ^{+B}. The smoothness parameter of $-\lambda B_\kappa$ is equal to the maximum value of $-4\lambda F'_\kappa(\Theta - \|\mathbf{w}\|^2)\|\mathbf{w}\|^2 + 2\lambda F_\kappa(\Theta - \|\mathbf{w}\|^2)$ over the domain $\{\mathbf{w} : \|\mathbf{w}\| \leq R\}$. Let $\mathsf{m}_{F'}$ denote the minimum of $F'_\kappa(x)$ over the interval $[\Theta - R^2, \kappa\Theta]$. The minimum of $F'_\kappa(x)$ over the interval $[\kappa\Theta, \Theta]$ is $-1/(\kappa\Theta)^2$. Since $F_\kappa(x)$ is decreasing on $[\Theta - R^2, \Theta]$, the maximum of the second term is $F_\kappa(\Theta - R^2) = (\frac{1}{\kappa\Theta} - P(\kappa\Theta)) + P_\kappa(\Theta - R^2)$ on the interval $[\Theta - R^2, \kappa\Theta]$ and $\frac{1}{\kappa\Theta}$ on the interval $[\kappa\Theta, \Theta]$. Thus, the total smoothness parameter of f_κ^{+B} is equal to $\beta_{f_\kappa^{+B}} = \beta_f + \max\{-4\lambda\mathsf{m}_{F'} + (\frac{1}{\kappa\Theta} - P(\kappa\Theta)) + P_\kappa(\Theta - R^2), \frac{4\lambda(1-\kappa)\Theta}{(\kappa\Theta)^2} + \frac{1}{\kappa\Theta}\}$.

Setting ζ. By the definition of F_κ on the interval $[\Theta - R^2, \kappa\Theta]$, the difference between $|F_\kappa(x) - P_\kappa(x)| = |F_\kappa(\kappa\Theta) - P_\kappa(\kappa\Theta)|$. On the other hand, $|F_\kappa(\kappa\Theta) - P_\kappa(\kappa\Theta)| \leq e_B$. Thus, the maximum error between F_κ and P_κ on the entire interval $[\Theta - R^2, \Theta]$ is at most e_B. The difference in gradients is therefore at most $2\lambda e_B\|\mathbf{w}\| \leq 2\lambda e_B R$. Thus, the total value of ζ is at most $e_f\sqrt{m} + 2\lambda e_B R = \zeta_f + 2\lambda e_B R$.

D. Guidance on Polynomial Approximation and Hyperparameter Selection

For concreteness, let us assume that the objective funciton f has the form

$$f(\mathbf{w}) = \sum_{(\mathbf{x},y)\in D} y\phi_1(\langle\mathbf{w},\mathbf{x}\rangle) + (1-y)\phi_0(\langle\mathbf{w},\mathbf{x}\rangle).$$

In particular, our experiments pertain to the logistic regression cost function, which has the above form. In this case, the approximate objective function g has the form

$$g(\mathbf{w}) = \sum_{(\mathbf{x},y)\in D} y\tilde{p}_1(\langle\mathbf{w},\mathbf{x}\rangle) + (1-y)\tilde{p}_0(\langle\mathbf{w},\mathbf{x}\rangle),$$

where for $y \in \{0,1\}$, $\tilde{p}'_y$ is an e_f-error approximation to ϕ'_y over some interval $[-z_{min}, z_{max}]$.

The quality and effectiveness of training under the approximate objective function g are governed by three critical parameters:

1. The smoothness constant β_g of the function g.
2. The approximation error e_f incurred when replacing the true gradient term ϕ'_y with its polynomial approximation $\tilde{p}'_y$.
3. The approximation interval $[-z_{\min}, z_{\max}]$ over which the guarantees of the polynomial approximation hold.

These parameters impact the training outcome in the following ways:

1. The smoothness constant β_g directly influences the convergence rate of the gradient descent algorithm. Specifically, a smaller β_g yields faster convergence.
2. The approximation error e_f affects the proximity of the output weights $\mathbf{w}_{g,T}$, produced by approximate gradient descent, to the optimal weights $\mathbf{w}_f^*$ that minimize the original objective function f. A smaller e_f leads to outputs that more closely approximate the true optimum.
3. The choice of approximation interval is crucial for ensuring convergence. If the interval $[-z_{\min}, z_{\max}]$ is not selected appropriately, the guarantees provided by the polynomial approximation may not hold. In such cases, the approximate gradient descent algorithm can diverge, producing weight vectors with unbounded magnitude (i.e., $\|\mathbf{w}_{g,T}\| \to \infty$), and leading the approximate objective value $g(\mathbf{w}_{g,T})$ to diverge toward $-\infty$. This does not contradict Theorem 3.3/Claim A.1, because the theorem/claim assumes the existence of a lower bound L on the objective. In the cases considered here, g has no global minimum and therefore no such lower bound exists.

We will illustrate each of these effects in section D.1 and D.2, including an example that demonstrates the consequences of an improperly chosen approximation interval. Motivated by these considerations, the next subsection D.3 provides guidance for selecting appropriate polynomial approximations and training hyperparameters, enabling stable, privacy-preserving training under fully homomorphic encryption.

D.1. Smoothness, Approximation Error, and Convergence Behavior

In the setting of outsourced machine learning under fully homomorphic encryption (FHE), eliminating the need for data-dependent hyperparameter tuning is essential. Unlike conventional training pipelines, where convergence can be monitored and hyperparameters such as the learning rate or the number of training steps can be adjusted dynamically, outsourced training under non-interactive FHE must operate entirely over encrypted data and without interaction.

To support such a non-interactive protocol, one must either (1) enable the client to privately determine appropriate hyperparameters in a lightweight *pre-processing* phase and communicate them in a differentially-private manner to the server, or (2) allow the server to select hyperparameters in a manner that is fully data-independent. In both cases, hyperparameters must be fixed in advance to enable one-shot execution by the server, without revealing intermediate computation or requiring feedback from the client.

Importantly, improperly selected hyperparameters may prevent convergence entirely. This is precisely where our theoretical results provide critical guidance. Theorem 3.3/Claim A.1 establish that setting the learning rate to $\eta = \frac{1}{\beta_g}$ and the number of gradient descent iterations to

$$T = \frac{2\beta_g \left(g(\mathbf{w}_{g,0}) - L\right)}{\rho^2}, \tag{32}$$

for a fixed $\rho > 0$, ensures convergence. These values depend on the properties of the polynomial approximation used for training and can be determined without reference to the underlying data distribution.

Moreover, our experimental results demonstrate that these hyperparameters can be estimated in a data-independent manner and still yield effective training performance in practice. This enables efficient, private, and non-interactive outsourced training entirely on the encrypted data.

In our experiments, we consider the original objective function f to be the cost function for logistic regression and we set

- Learning rate as $\eta_1 \approx \frac{1}{\beta_{\text{approx}}}$, where $\beta_{\text{approx}} = \max\{\tilde{p}'(z)\} \cdot m$, is an upperbound on the smoothness constant β_g, and $\max\{\tilde{p}'(z)\}$ is the maximum value of $\tilde{p}'_0(z), \tilde{p}'_1(z)$ over the interval $[-z_{min}, z_{max}]$. [6]
- Training steps $T \approx \frac{2\beta_{\text{approx}}(g(\mathbf{w}_{g,0}))}{\rho^2}$ (taking $L = 0$).

This work is the first to explicitly address the challenge of hyperparameter selection in the context of fully homomorphic encryption (FHE)-based training by proposing a principled method for estimating both the learning rate and the number of training iterations *in advance*, based on the properties of the chosen polynomial approximation.

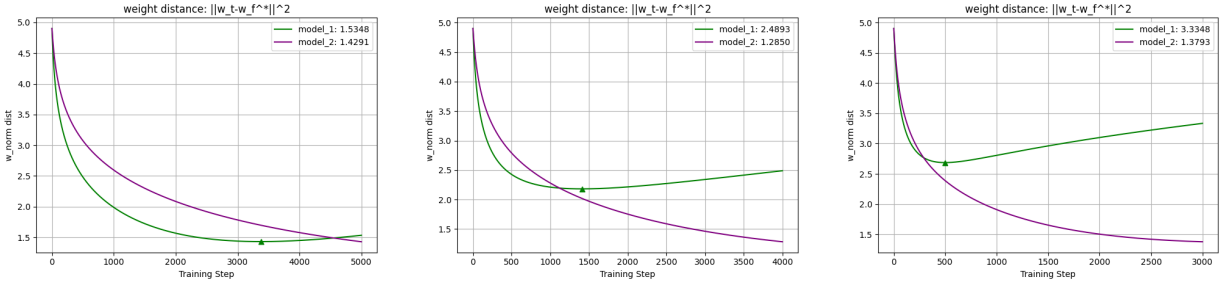
After establishing how to select the learning rate and training steps given a fixed polynomial approximation, we now turn to the choice of the polynomial itself. In contrast to prior work, which either relies on manual, data-dependent, tuning or left them unspecified, we provide analytically grounded estimates. This advance enables fully automated, non-interactive, and privacy-preserving training under FHE that is both practical and efficient. Notably, prior FHE-based training works typically adopted a single polynomial approximation scheme, such as minimax or least-squares, without a principled justification of the resulting trade-offs. Our analysis shows that the choice of polynomial approximation has a direct impact on training behavior such as convergence rate and the optimization error. Even with same predefined approximation interval and with the same fixed degree, different approximation schemes can obtain polynomials with different smoothness and approximation error, further influencing overall training behavior.

Our convergence analysis provides practical guidance for selecting the polynomial approximation used in training. In particular, the theoretical results indicate that convergence is faster when the smoothness parameter β_g is smaller. As

[6] Note that one could instead use the potentially smaller value of $\frac{\max\{\tilde{p}'(z)\}}{N} \sum_{\mathbf{x}} \mathbf{x}^T\mathbf{x}$ as an upperbound on the smoothness. This quantity is data-dependent, but the Client can easily compute $\frac{1}{N}\sum_{\mathbf{x}} \mathbf{x}^T\mathbf{x}$ in a pre-processing step and release it to the Server in a differentially private manner.

Table 3. Summary of Approximate-GD Training with Fixed Hyperparameters (7-Degree Polynomial, $\rho = 0.1$ for MNIST, $\rho = 0.05$ for others) using either minimax (MM) or least squares (LS) approximation. LR stands for learning rate.

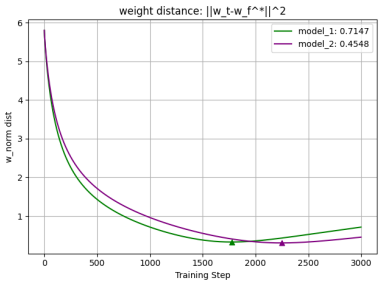
Dataset	Pair	model	Method	Error	Smoothness	LR Est.	$T \cdot \rho^2$ Est.	Steps@ρ	Accuracy (%)
MNIST	1	model-1	MM@[-20, 20]	0.1920	19.1185	0.0523	26.50	2650	96.3529
		model-2	LS@[-20, 20]	0.1439	35.1523	0.0284	48.73	4873	96.3937
	2	model-1	MM@[-25, 25]	0.2335	15.3325	0.0652	21.26	2125	96.2609
		model-2	LS@[-25, 25]	0.1834	27.4613	0.0364	38.07	3806	96.3328
	3	model-1	MM@[-30, 30]	0.3904	13.6093	0.0735	18.87	1886	96.1875
		model-2	LS@[-30, 30]	0.4250	16.8512	0.0593	23.36	2336	96.2694
Adult	1	model-1	MM@[-10, 10]	0.0632	2.3223	0.4306	3.22	1287	84.4332
		model-2	LS@[-10, 10]	0.0498	2.5594	0.3907	3.55	1419	84.3991
	2	model-1	MM@[-15, 15]	0.1356	1.6680	0.5995	2.31	924	84.3650
		model-2	LS@[-15, 15]	0.0955	2.6682	0.3748	3.70	1479	84.3309
	3	model-1	MM@[-20, 20]	0.1920	1.2681	0.7886	1.76	703	84.3309
		model-2	LS@[-20, 20]	0.1439	2.3315	0.4289	3.23	1292	84.2285
Compas	1	model-1	MM@[-7, 7]	0.0224	1.7384	0.5752	2.41	963	67.6212
		model-2	LS@[-7, 7]	0.0230	1.8121	0.5518	2.51	1004	67.6212
	2	model-1	MM@[-10, 10]	0.0632	1.4291	0.6997	1.98	792	67.4827
		model-2	LS@[-10, 10]	0.0498	1.5750	0.6349	2.18	873	67.5751
	3	model-1	MM@[-15, 15]	0.1356	1.0264	0.9742	1.42	569	67.2055
		model-2	LS@[-15, 15]	0.0955	1.6420	0.6090	2.28	910	67.4436
Credit	1	model-1	MM@[-10, 10]	0.0632	4.1087	0.2434	5.70	2278	80.6229
		model-2	LS@[-10, 10]	0.0498	4.5281	0.2208	6.28	2510	80.6452
	2	model-1	MM@[-15, 15]	0.1356	2.9510	0.3389	4.09	1636	80.4378
		model-2	LS@[-15, 15]	0.0955	4.7207	0.2118	6.54	2617	80.4341
	3	model-1	MM@[-20, 20]	0.1920	2.2435	0.4457	3.11	1244	80.2970
		model-2	LS@[-20, 20]	0.1439	4.1250	0.2424	5.72	2287	80.2822



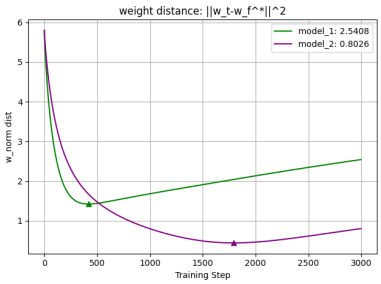
(a) Mnist Pair-1

(b) Mnist Pair-2

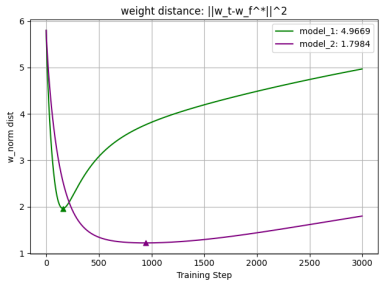
(c) Mnist Pair-3



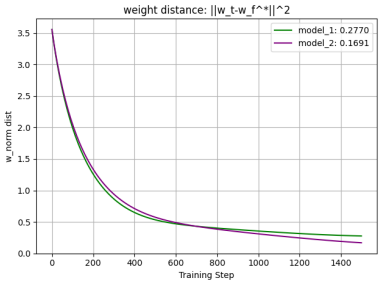
(d) Adult Pair-1



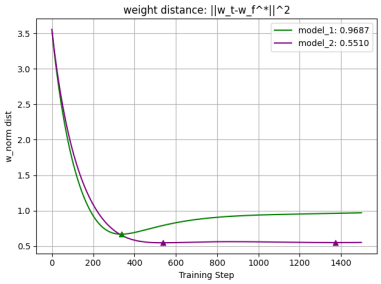
(e) Adult Pair-2



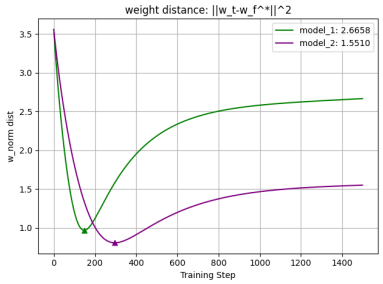
(f) Adult Pair-3



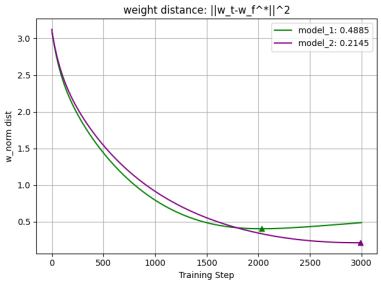
(g) Compas Pair-1



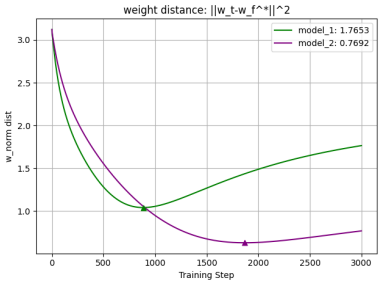
(h) Compas Pair-2



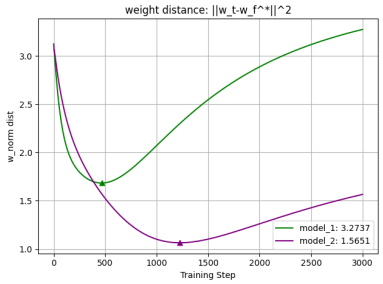
(i) Compas Pair-3



(j) Credit Pair-1



(k) Credit Pair-2



(l) Credit Pair-3

Figure 3. Pairwise comparisons of $\|\mathbf{w}_T - \mathbf{w}^*\|^2$ during training for two polynomial approximations. Model 1 uses a minimax polynomial approximation, while Model 2 uses a least-squares approximation. In each subfigure, the boxed values report the final optimization error $\|\mathbf{w}_T - \mathbf{w}^*\|^2$ after T iterations. We observe that Model 1 has a smaller $T\rho^2$ value (see Table 3) and converges more rapidly, consistent with its stronger smoothness properties. In contrast, Model 2 has lower approximation error (Table 3) and, in most cases, achieves a smaller final optimization error.

illustrated in Figure 3 and summarized in Table 3, we compare training using the minimax and least-squares polynomial approximations of a fixed degree. Interestingly, the experimental results establish that the two approximation schemes exhibit a clear and systematic trade-off. The minimax approximation consistently yields smaller smoothness (see Table 3), and indeed exhibits faster convergence, and this is the case even when the learning rate is tuned to the data, as opposed to being set a priori (see Figure 3). In contrast, the least squares approximation consistently achieves smaller approximation error (see Table 3) and typically also achieves smaller optimization error $\|\mathbf{w}_{g,T} - \mathbf{w}_f^*\|^2$ (see Figure 3).

Specifically, training with polynomial approximations that yield smaller values of $T \cdot \rho^2$, where T and ρ are defined in terms of β_g, results in more rapid convergence toward the optimal solution $\mathbf{w}_f^*$. Notably, this quantity can be approximated using a data-independent estimate β_{approx}, which may be computed by the server. This implies that selecting a polynomial approximation with smaller β_{approx} allows similar training performance to be achieved using fewer gradient steps, thereby improving overall efficiency.

On the other hand, Theorem 3.3 shows that a larger value of ζ, which captures the distance between the gradient of the approximate objective g and that of the true objective f, corresponds to a greater deviation between the weights output by approximate gradient descent and the true minimizer $\mathbf{w}_f^*$.

Recall that the gradient of the original objective function f takes the form:

$$\nabla f(\mathbf{w}) = \sum_{(\mathbf{x},y)\in D} \phi'_y(\langle \mathbf{w}, \mathbf{x}\rangle) \cdot \mathbf{x},$$

while the gradient of the approximated objective g is:

$$\nabla g(\mathbf{w}) = \sum_{(\mathbf{x},y)\in D} \tilde{p}'_y(\langle \mathbf{w}, \mathbf{x}\rangle) \cdot \mathbf{x}.$$

We adopt the heuristic that the approximation error e_f, defined as the maximum deviation between ϕ'_y and $\tilde{p}'_y$, contributes directly to ζ. Importantly, e_f is a data-independent quantity that can be computed by the server prior to training.

Empirical results support this relationship: in nearly all tested cases, larger values of e_f lead to final output weights that lie farther from the optimum (see Figure 3). These findings highlight the importance of careful polynomial selection, guided by both theoretical analysis and pre-computable approximation metrics.

D.2. Approximation Interval and Training Stability

As discussed in Section D, selecting an appropriate approximation interval is critical for ensuring both the accuracy and stability of training under polynomial approximations. If the interval is chosen too large, the approximation error increases for a fixed polynomial degree. This phenomenon is clearly demonstrated in Table 3, where smaller intervals consistently yield lower approximation errors for the same degree of polynomial.

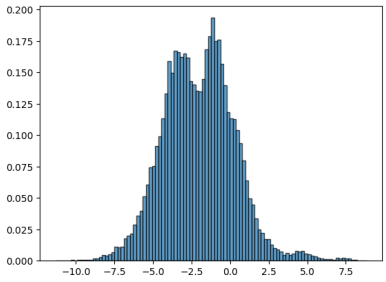
On the other hand, if the interval is chosen too small, there is a significant risk that the quantity $|\langle \mathbf{w}_i, \mathbf{x}\rangle|$ will exceed the approximation range during training. While in the case of logistic regression loss function is lower-bounded by zero regardless of the input magnitude, the polynomial approximation g may be *unbounded below* outside the interval. Specifically, as $|\langle \mathbf{w}_i, \mathbf{x}\rangle| \to \infty$, the value of the loss $g(\mathbf{w})$ can diverge to $-\infty$, resulting in the lower bound parameter L in Equation (32) becoming unbounded. This can lead to instability in training: the approximate gradient descent procedure may diverge, producing weight vectors $\mathbf{w}_{g,T}$ with unbounded magnitude and corresponding loss values that tend toward $-\infty$.

An illustrative example of this behavior arises when training on the Adult dataset with the approximation interval set to $[-7, 7]$. As shown in Figure 4, during later iterations, the value of $|\langle \mathbf{w}_i, \mathbf{x}\rangle|$ exceeds the interval, causing the magnitude of the weights to diverge and the loss to descend toward negative infinity. This ultimately leads to numerical instability, resulting in NaN values in both the training loss and gradient computations.

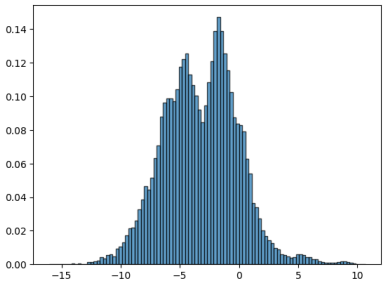
It is unclear how one can determine a suitable approximation interval for $|\langle \mathbf{w}_i, \mathbf{x}\rangle|$ in a non-dynamic or data-independent manner. In such cases, training can instead be performed using the modified objective function f_κ^{+B}, which incorporates a barrier function as defined in Section 5. If differential privacy is not required, the barrier can be employed without injecting noise.

This approach enforces a strict upper bound on the quantity $|\langle \mathbf{w}_i, \mathbf{x}\rangle|$ throughout training, regardless of the number of iterations T (See Lemma 5.5). Consequently, with appropriately chosen approximation functions $\tilde{p}'$ and P_κ (See Section 5),

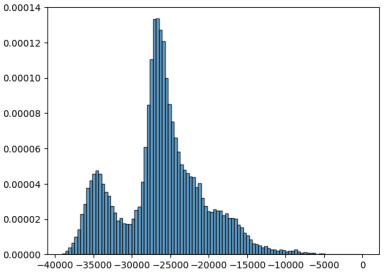
the objective $g(\mathbf{w}_T)$ remains bounded from below by a constant L, and convergence is guaranteed via Theorem 3.3. Figure 5 illustrates successful convergence using this barrier-augmented objective.



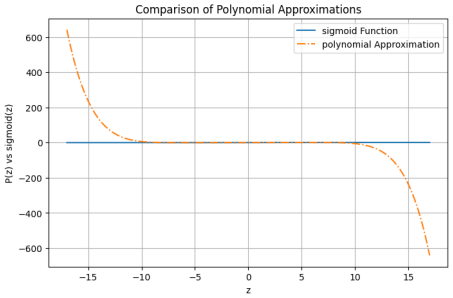
(a) z values at t=1000



(b) z values at t=1080



(c) z values at t=1086



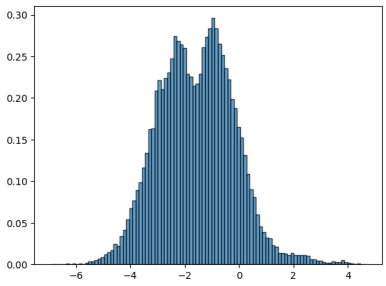
(d) Comparison of sigmoid function and its polynomial approximation over a wide range. They exhibit close on $[-10, 10]$, but diverge outside this range. Polynomial grows unbounded as $z \to \pm\infty$, while sigmoid remains bounded between 0 and 1.

```
Cost after 1170 iteration is 0.2785552835431717, with w norm^2: 41.509432558147715, and max of z is: 14.184978563480527
Cost after 1171 iteration is 0.2740211284546876, with w norm^2: 41.936257603775 36, and max of z is: 14.284006869246149
Cost after 1172 iteration is 0.26850617639793584, with w norm^2: 42.40597584846951, and max of z is: 14.392338610300104
Cost after 1173 iteration is 0.2616688567796334, with w norm^2: 42.927605675737794, and max of z is: 14.511752939577 37
Cost after 1174 iteration is 0.2529936094496 1896, with w norm^2: 43.51324505390738, and max of z is: 14.644574215783795
Cost after 1175 iteration is 0.24166722518571 47, with w norm^2: 44.17961722236565, and max of z is: 14.793914270345963
Cost after 1176 iteration is 0.22633534660544202, with w norm^2: 44.95070174590769, and max of z is: 14.964062964658089
Cost after 1177 iteration is 0.2045818592407282, with w norm^2: 45.86249226642235, and max of z is: 15.161148729450503
Cost after 1178 iteration is 0.1716924323891098, with w norm^2: 46.9722781992271, and max of z is: 15.394323067850474
Cost after 1179 iteration is 0.11727913292188241, with w norm^2: 48.3786186441613, and max of z is: 15.678045721234113
Cost after 1180 iteration is 0.014092292358617198, with w norm^2: 50.270393862541 56, and max of z is: 16.036927276056375
Cost after 1181 iteration is -0.23164737208495334, with w norm^2: 53.072203966022485, and max of z is: 16.51736041669211
Cost after 1182 iteration is -1.1445281471484663, with w norm^2: 58.022070164261535, and max of z is: 17.220829019495227
Cost after 1183 iteration is -12.082794552504282, with w norm^2: 71.09745355777156, and max of z is: 18.428482333713966
Cost after 1184 iteration is -50118.617574744785, with w norm^2: 220.46878878911455, and max of z is: 21.340312961447 54
Cost after 1185 iteration is -1.1418642138079625e+29, with w norm^2: 152084851.82915044, and max of z is: 42.40279664556737
Cost after 1186 iteration is -7.6114629611211962e+199, with w norm^2: 7.177319919381082e+50, and max of z is: 39058.14363790047
Cost after 1187 iteration is -inf, with w norm^2: inf, and max of z is: 8.356680500765925e+25
Cost after 1188 iteration is nan, with w norm^2: nan, and max of z is: 2.511238839773593e+175
Cost after 1189 iteration is nan, with w norm^2: nan, and max of z is: nan
/tmp/ipython-input-4-436742036.py:124: RuntimeWarning: overflow encountered in multiply
```

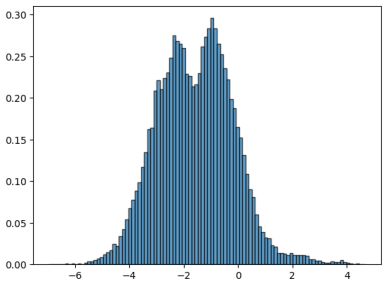
(e) Due to the unbounded growth of the linear hypothesis value z in the later training iteration, value $\tilde{p}'(z)$ diverges, and so does gradient and training loss, leading to a runtime error with `nan`.

Figure 4. We trained on the Adult dataset using a polynomial approximation over the range $[-7, 7]$. However, as training progresses, the linear hypothesis value $z = \langle \mathbf{w}, \mathbf{x} \rangle$ sometimes exceeds this range. Since the polynomial $\tilde{p}'(z)$ is only accurate within $[-7, 7]$, evaluating it outside this range causes it to diverge, eventually leading to runtime errors as $\tilde{p}'(z) \to \infty$.

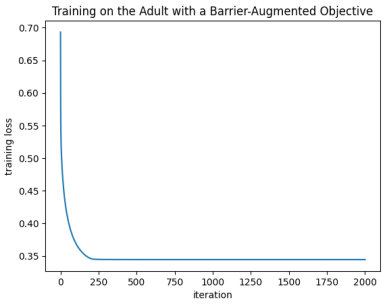
A comparison between Figure 4 and Figure 5 clearly demonstrates the advantage of training with a barrier-augmented objective function (as in Algorithm 3). Both experiments were conducted using the same dataset, polynomial approximation, learning rate, and number of training steps. The only distinction is that the model in Figure 5 was trained using an objective that incorporates a barrier function. The results show that the barrier-augmented objective ensures stable convergence. As illustrated in Figure 5c, the training process converges reliably under this formulation. Specifically, at training steps 1080 and 1086, the inner product values $z = \langle \mathbf{w}_i, \mathbf{x} \rangle$ remain bounded within the interval $[-7, 7]$, as shown in Figures 5a and 5b, respectively. In contrast, training on the standard (non-barrier) objective exhibits unbounded growth in z. At step 1080, the values expand to the range $[-15, 10]$ (Figure 4b), and by step 1086, they diverge dramatically, reaching nearly -4000 (Figure 4c).



(a) z values at t=1080



(b) z values at t=1086



(c) training loss converge over training with barrier-augmented objective

Figure 5. We trained on the Adult dataset using the same polynomial approximation, but with a modified objective function incorporating a barrier function (as in Algorithm 3). Throughout the whole training process, the linear hypothesis value z remained bounded and and the training successfully converged.

D.3. Data-Independent Selection of Polynomial Approximations and Hyperparameters

We now describe a fully data-independent procedure for selecting the polynomial approximations and training hyperparameters for the training procedure in Algorithm 3, for our target application of logistic regression. This procedure ensures that all theoretical conditions are satisfied and can be executed *a priori*, without access to the underlying dataset. The sequence of steps presented in this section is just one possible way to select the parameters and approximations. While Theorem 5.1 provide the constraints that determine feasible values for guaranteeing DP, alternative orders or strategies for adjusting Θ, λ, and κ, and the polynomial approximations are equally valid, as long as all conditions in theorem are eventually satisfied. This flexibility allows practitioners to adopt different sequences or heuristics for parameter selection without affecting the correctness or differential privacy guarantees of Algorithm 3.

We begin by initializing the learning rate following smoothness-based analysis, we initialize the learning rate as $\eta_3 \approx \frac{1}{2\beta_{\text{approx}}}$, where $\beta_{\text{approx}} = m/4$. We also fix a target approximation tolerance E_f for ϕ', which is the sigmoid function for the case of logistic regression (e.g. in practice, we initialize $E_f = 0.05$.)

Next, we select the parameters governing the barrier function, namely the threshold Θ and the penalty coefficient λ. The threshold Θ determines the maximum admissible magnitude of $\|\mathbf{w}_i\|$, and therefore the maximum magnitude of the inner products $|\langle \mathbf{w}_i, \mathbf{x} \rangle|$. Thus, Θ defines the approximation interval required by Theorem 5.1, while λ controls the strength of the barrier penalty. Larger values of Θ and smaller values of λ generally improve model accuracy, whereas smaller Θ and larger λ enforce stricter stability. In practice, we initialize Θ to the feature dimension and set λ to a small value (e.g., 0.001). We then verify whether this choice satisfies constraint (2) in Theorem 5.1. A larger Θ increases the approximation interval, which in turn raises the polynomial approximation error for a fixed degree. Conversely, if λ is too small, the RHS of (2) increases and may violate the constraint. If the constraint is not satisfied, we iteratively decrease Θ and increase λ until the inequality holds. This adjustment process is entirely data-independent and requires no training.

We select the parameter κ simultaneously with Θ and λ. The parameter κ specifies the interval on which the polynomial P_κ must approximate the derivative of the barrier function, $1/x$. Although decreasing κ increases the LHS and decreases the RHS (by increasing m_P) of (2) and facilitates feasibility, doing so may increase approximation error. To balance these effects, we adopt an adaptive strategy: we initialize κ to a relatively large value (e.g., 0.01), which yields smaller approximation error (e_B), and then progressively decrease κ until constraint (2) is satisfied.

With κ fixed, we select the corresponding polynomial P_κ and compute the true approximation error e_B, defined as the maximum approximation error of P_κ from $1/x$ over the interval $[\kappa\Theta, \Theta]$. Several methods can be used to construct P_κ, including Newton-Raphson, Taylor series, or least-squares approximation. The goal is to approximate $1/x$ accurately over $[\kappa\Theta, \Theta]$, keeping e_B small. This e_B error is then used to determine the approximation interval for the sigmoid function, whose formula is defined in the first point in Theorem 5.1. A smaller e_B narrows the required approximation interval for ϕ', corresponding to the sigmoid function in logistic regression, which in turn reduces the approximation error e_f for a given polynomial degree. Additionally, the polynomial should be monotone decreasing over the interval $[\Theta - R^2, \kappa\Theta]$, with the left tail kept as low as possible. This ensures that the maximum value, denoted by M_P, over this interval does not become excessively large, which allows a larger learning rate while maintaining stability. The Newton-Raphson algorithm

can achieve very high accuracy but often requires a higher polynomial degree, whereas Taylor series approximations can work with low-degree polynomials but sacrifice the accuracy. The choice of method can be made based on the desired trade-off between computational efficiency and precision. For simplicity and sufficient precision, we adopt a least-squares approach. In our experiments, a polynomial of degree 4 obtained via the least-squares method was adequate across all datasets. Using P_κ, we compute the maximum and minimum values attained over the interval $[\Theta - R^2, \kappa\Theta]$, denoted by M_P and m_P, respectively. These quantities are used to verify whether the current learning rate satisfies condition (1) in Theorem 5.1. If the condition is not satisfied, the learning rate is reduced, and the procedure restarts from the step of computing the approximation interval. In our experiments, the initially chosen learning rates were typically sufficient to satisfy this condition, so few adjustments were required.

Finally, we select the polynomial degree for the sigmoid approximation. For each candidate degree, we verify three conditions: (i) the resulting approximation error e_f does not exceed the preset tolerance E_f; (ii) the stability parameter $\alpha = 2\eta_3\lambda\mathsf{m}_P$ satisfies $\alpha \leq 1$; and (iii) the constraint (2) holds. If any of these conditions are violated, we return to the barrier-parameter selection step-adjusting κ, Θ, λ as necessary, and repeat the procedure.

These three parameters can be adjusted in combination: increasing λ, or decreasing κ and Θ pushes the system closer to satisfying the constraint (2). In practice, when the LHS and RHS of (2) are already close (e.g., within a ratio of 100), it is generally sufficient to keep λ and Θ fixed and adjust only κ, which simplifies the tuning procedure.

This process yields a complete set of polynomial approximations and hyperparameters that can be fixed *a priori*. As a result, it enables stable, non-interactive training under fully homomorphic encryption while satisfying all theoretical constraints.

E. Additional Experimental Results

Table 4. Accuracy and AUC for different models over 100 runs across four (4) datasets under a privacy budget of ($\varepsilon = 1$, $\delta = 10^{-5}$). Results are reported as mean $\pm$ standard deviation. Model 1: standard DP-GD; Model 2: augmented no-clipping DP-GD, where the sigmoid and barrier functions are respectively replaced by a 7th or 9th-degree polynomial and a 4th-degree polynomial; Model 3: DP model using output perturbation.

Dataset	Model	Accuracy	AUC
mnist	model1_dpgd	0.9579 ± 0.0004	0.9898 ± 0.0001
	model2_noclipping	0.9549 ± 0.0038	0.9890 ± 0.0010
	model3_output_GD	0.7204 ± 0.0937	0.8687 ± 0.0716
adult	model1_dpgd	0.7947 ± 0.0001	0.8692 ± 0.0002
	model2_noclipping	0.8050 ± 0.0015	0.8750 ± 0.0014
	model3_output_GD	0.7660 ± 0.0384	0.8191 ± 0.0222
compas	model1_dpgd	0.6486 ± 0.0028	0.7034 ± 0.0015
	model2_noclipping	0.6325 ± 0.0150	0.6929 ± 0.0121
	model3_output_GD	0.5126 ± 0.0418	0.5490 ± 0.0741
credit	model1_dpgd	0.7795 ± 0.0000	0.6411 ± 0.0003
	model2_noclipping	0.7828 ± 0.0016	0.6501 ± 0.0022
	model3_output_GD	0.7657 ± 0.0623	0.6012 ± 0.0355

Table 5. Multiplicative depth cost per iteration for the DP training approach with and without gradient clipping and for Output-GD, which runs no-DP under FHE

Computational Step	DP-GD (w/ Clip)	DP-GD (No Clip)	Output-GD
Path A: Gradient Calculation (Per-sample)			
$\mathbf{z} = \langle \mathbf{w}, \mathbf{x} \rangle$ (Inner Product)	+1	+1	+1
Sigmoid Poly	+4	+4	+4
Gradient $((y_{pred} - y) * \mathbf{x})$	+2	+2	+2
— *Per-Sample Clipping* —			
$\|\|\text{grad}_i\|\|^2$ (Norm)	+1	—	—
$\sqrt{\text{norm}}$ (Sqrt Poly)	+4	—	—
$\max\{C, \|\|\text{grad}_i\|\|\}$	+6	—	—
Inverse $\sqrt{\text{norm}}$ (using TS)	+3	—	—
Scaling Logic (grad_i*scal)	+1	—	—
Average Gradient	+1	+1	+1
Path B: Barrier Calculation (Parallel to A)			
$\|\|\mathbf{w}\|\|^2$ (Squared Norm)	—	+1	—
Inverse Term (using LS)	—	+3	—
Barrier Term	—	+2	—
Critical Path Depth	23	$\max(8, 6) = 8$	8
Weight Update (grad * lr)	+1	+1	+1
Total Cost per Iteration	**24**	**9**	**9**

F. Protocols

Algorithm 4 Traditional DP-(S)GD Protocol using FHE

Input: Private data (X, y), learning rate η, epochs T, clipping norm $C = 1$, DP noise vectors $\{z_t\}_{t=1}^{T}$, sigmoid approximation function P_{sigmoid}, BatchSize n

// Client Side: Key Generation

$(\texttt{pk}, \texttt{sk}) \leftarrow \texttt{HE.KeyGen()}$

// Client Side: Data Encryption

$\texttt{Enc\_X} \leftarrow \texttt{HE.Encrypt}(\texttt{pk}, X)$

$\texttt{Enc\_y} \leftarrow \texttt{HE.Encrypt}(\texttt{pk}, y)$

$\texttt{Enc\_z} \leftarrow \texttt{HE.Encrypt}(\texttt{pk}, \{z_t\}_{t=1}^{T})$

$\texttt{Send } (\texttt{Enc\_X}, \texttt{Enc\_y}, \texttt{Enc\_z}) \texttt{ to Server}$

// Server Side: Encrypted Training

$\texttt{Enc\_w} \leftarrow \texttt{HE.Encrypt}(\texttt{pk}, 0^d)$

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

  $\texttt{Enc\_grad\_sum} \leftarrow \texttt{HE.Encrypt}(\texttt{pk}, 0^d)$

  $I_t \leftarrow \text{SampleRandomIndices}(N, n)$

  for i in I_t **do**

    $\texttt{Enc\_logit}_i \leftarrow \langle \texttt{Enc\_X}[i], \texttt{Enc\_w} \rangle$

    $\texttt{Enc\_pred}_i \leftarrow P_{\text{sigmoid}}(\texttt{Enc\_logit}_i)$

    $\texttt{Enc\_error}_i \leftarrow \texttt{Enc\_pred}_i - \texttt{Enc\_y}[i]$

    $\texttt{Enc\_grad}_i \leftarrow \texttt{Enc\_X}[i] \cdot \texttt{Enc\_error}_i$

    $\texttt{Enc\_grad}_i \leftarrow P_{\text{clip}}(\texttt{Enc\_grad}_i, C)$

    $\texttt{Enc\_grad\_sum} \leftarrow \texttt{Enc\_grad\_sum} + \texttt{Enc\_grad}_i$

  end for

  $\texttt{Enc\_grad} \leftarrow \frac{1}{n} \cdot \texttt{Enc\_grad\_sum} + \texttt{Enc\_z}[t]$

  $\texttt{Enc\_w} \leftarrow \texttt{Enc\_w} - \eta \cdot \texttt{Enc\_grad}$

end for

// Server sends $\texttt{Enc\_w to Client}$

// Client Side: Decryption

$w \leftarrow \texttt{HE.Decrypt}(\texttt{sk}, \texttt{Enc\_w})$

Output: Trained model weights w

Algorithm 5 Proposed non-clipping DP-(S)GD protocol

Input: Private data (X, y), learning rate η, epochs T, DP noise vectors $\{z_t\}_{t=1}^{T}$, sigmoid approximation function P_{sigmoid}; barrier function's parameters: λ and Θ, inverse approximation function P_{rep}, BatchSize n
Output: Trained model weights w
// Client Side: Key Generation
$(\texttt{pk}, \texttt{sk}) \leftarrow \texttt{HE.KeyGen()}$
// Client Side: Data Encryption
$\texttt{Enc\_X} \leftarrow \texttt{HE.Encrypt}(\texttt{pk}, X)$
$\texttt{Enc\_y} \leftarrow \texttt{HE.Encrypt}(\texttt{pk}, y)$
$\texttt{Enc\_z} \leftarrow \texttt{HE.Encrypt}(\texttt{pk}, \{z_t\}_{t=1}^{T})$
$\texttt{Send (Enc\_X, Enc\_y, Enc\_z) to Server}$
// Server Side: Encrypted Training
$\texttt{Enc\_w} \leftarrow \texttt{HE.Encrypt}(\texttt{pk}, 0^d)$
for $t = 1$ to T **do**
  $\texttt{Enc\_grad\_sum} \leftarrow \texttt{HE.Encrypt}(\texttt{pk}, 0^d)$
  $I_t \leftarrow \text{SampleRandomIndices}(N, n)$
  for i in I_t **do**
    $\texttt{Enc\_logit}_i \leftarrow \langle \texttt{Enc\_X}[i], \texttt{Enc\_w} \rangle$
    $\texttt{Enc\_pred}_i \leftarrow P_{\text{sigmoid}}(\texttt{Enc\_logit}_i)$
    $\texttt{Enc\_error}_i \leftarrow \texttt{Enc\_pred}_i - \texttt{Enc\_y}[i]$
    $\texttt{Enc\_grad}_i \leftarrow \texttt{Enc\_X}[i] \cdot \texttt{Enc\_error}_i$
    $\texttt{Enc\_grad\_sum} \leftarrow \texttt{Enc\_grad\_sum} + \texttt{Enc\_grad}_i$
  end for
  $\texttt{Enc\_barrier} \leftarrow P_{rep}(\Theta - \|\texttt{Enc\_w}\|^2)$
  $\texttt{Enc\_grad} \leftarrow \frac{1}{n} \cdot \texttt{Enc\_grad\_sum} + 2 \cdot \lambda \cdot \texttt{Enc\_w} \cdot \texttt{Enc\_barrier} + \texttt{Enc\_z}[t]$
  $\texttt{Enc\_w} \leftarrow \texttt{Enc\_w} - \eta \cdot \texttt{Enc\_grad}$
end for
// Server sends $\texttt{Enc\_w to Client}$
// Client Side: Decryption
$w \leftarrow \texttt{HE.Decrypt}(\texttt{sk}, \texttt{Enc\_w})$
Output: w