# Private Data Analysis - Homework 4

Yanning Dai (212008)

November 18, 2024

## LSTM-Based Name Classifier with Differential Privacy

## 1. Task Description

The goal of this experiment is to implement and evaluate a differentially private stochastic gradient descent (DPSGD) algorithm [1] for a classification task on the MNIST dataset. Using the Opacus library, we manually implemented the core components of DPSGD: Privacy Accounting and Gradient Clipping. The experiment explores how privacy guarantees (quantified by $\epsilon$) impact the utility of the trained model. By varying the privacy parameter $\epsilon$, we generate a curve to illustrate the trade-off between privacy guarantees and model accuracy.

## 2. Differentially Private SGD

Differentially Private SGD (DPSGD) extends traditional stochastic gradient descent by incorporating two privacy-preserving techniques: Gradient Clipping and Privacy Accounting.

### a. Gradient Clipping

Gradient Clipping limits the influence of individual data samples by bounding the $L_2$-norm of their gradients:

1. Compute the gradient $\nabla L(x_i)$ for each sample $x_i$.
2. If $\|\nabla L(x_i)\| > C$ (where $C$ is the clipping threshold, set to 1.0), scale the gradient:

$$\nabla L'(x_i) = \frac{C}{\|\nabla L(x_i)\|} \cdot \nabla L(x_i)$$

3. This ensures that no single data point disproportionately influences the model update.

### b. Privacy Accounting

Privacy Accounting tracks the cumulative privacy loss across training iterations by adding Gaussian noise to the batch gradient:

1. The noise is scaled by a multiplier ($\sigma$), computed using binary search to meet the desired privacy budget $(\epsilon, \delta)$.
2. Noise is added after dividing by the batch size $B$ to prevent noise amplification during averaging:

$$\nabla L'_{\text{batch}} = \nabla L_{\text{batch}} + \frac{\mathcal{N}(0, \sigma^2)}{B}$$

3. This ensures the added noise matches individual gradient noise scaling while reducing variance caused by averaging.

The DPSGD process integrates these two techniques as follows:

1. Gradients for each sample are clipped and aggregated to form the batch gradient.
2. Gaussian noise, scaled to the privacy budget, is added to the aggregated batch gradient.
3. The model parameters are updated using the noised batch gradient.

These processes were manually implemented to deepen the understanding of DPSGD and its impact on privacy and utility.

## 3. Dataset

The dataset used in this experiment is the MNIST handwritten digit dataset, consisting of 60,000 training images and 10,000 testing images. Each grayscale image is of size $28 \times 28$ and labeled with a digit from 0 to 9. The dataset was preprocessed using PyTorch utilities, where pixel values were normalized to the range [0, 1], and training batches were sampled at a rate of 0.1.

## 4. Experimental Setup

This section describes the hyperparameters used for the experiment and the procedure followed to evaluate the impact of differential privacy on model performance.

### a. Hyperparameters

The following hyperparameters were used in this experiment to train the model under differential privacy constraints:

- Total number of iterations: 100
- Number of epochs: 10
- Sampling rate: 0.1 (batch size = 6000 samples per iteration)
- Clipping threshold ($C$): 1.0
- Learning rate: 0.01
- Privacy parameter ($\delta$): $10^{-5}$

### b. Procedure

The procedure followed in the experiment consists of four main steps: privacy accounting, gradient clipping, model training, and evaluation. Below are the details for each step:

**Privacy Accounting and Noise Addition:** To ensure the model satisfies differential privacy constraints, Gaussian noise was added to the batch gradient during training. The process involved:

- For each $\epsilon$ value $[0.1, 0.5, 1.0, 2.0, 5.0, 10.0]$, the noise multiplier ($\sigma$) was determined using binary search to meet the desired privacy budget.
- Gaussian noise with standard deviation $\frac{\sigma \cdot C}{B}$ was added to the batch gradient. This approach minimizes variance reduction while maintaining privacy guarantees.

**Gradient Clipping:** To limit the influence of individual samples on the model, gradient clipping was applied:

- The $L_2$-norm of each sample's gradient was computed.
- Gradients exceeding the clipping threshold ($C = 1.0$) were scaled down to ensure they do not dominate the aggregated batch gradient.

**Model Training:** The model was trained with differential privacy guarantees for each $\epsilon$ value:

- For every $\epsilon$, the model was trained for 10 epochs.
- To ensure robust evaluation, each $\epsilon$ configuration was repeated 10 times, and the results were averaged.

**Evaluation:** After training, the model's performance was evaluated on the test set:

- Test accuracy was recorded for each trained model.
- The mean accuracy for each $\epsilon$ value was computed to analyze the privacy-utility trade-off.

## 5. Results and Discussions

This section discusses the results of noise computation, gradient clipping, and the privacy-utility trade-off observed in the experiment.

### a. Validation of Noise Computation and Gradient Clipping

The noise multiplier ($\sigma$) was computed using binary search to satisfy the desired privacy guarantee ($\epsilon = 1, \delta = 10^{-5}$). The privacy loss values converged as expected over the iterations, and the final noise multiplier was determined to be $\sigma = 4.2776$. This result confirms the correctness of the manual noise computation process.

To validate the gradient clipping implementation, the manually computed clipped gradients were compared with those from the Opacus library. The norms of the two sets of clipped gradients matched closely, with differences well within the specified tolerance ($10^{-5}$). These results confirm the accuracy of both noise computation and gradient clipping in the manual implementation of DPSGD.

### b. Privacy-Utility Trade-Off

The mean accuracy for different $\epsilon$ values was computed to evaluate the trade-off between privacy guarantees and model utility. The experimental results are summarized in Figure 1, which shows the relationship between $\epsilon$ and the test accuracy.

The results demonstrate a clear trend: as $\epsilon$ increases, the test accuracy improves due to reduced noise added during training. For smaller $\epsilon$ values (e.g., $\epsilon = 0.01$), the model accuracy was

significantly lower (82.89%), reflecting the impact of heavy noise and stronger privacy guarantees. In contrast, larger $\epsilon$ values (e.g., $\epsilon = 10.0$) resulted in higher accuracy (96.21%), as less noise was injected during training. This trade-off between privacy and utility highlights the inherent balance in differential privacy: smaller $\epsilon$ values offer stronger privacy but reduce model performance, while larger $\epsilon$ values provide weaker privacy with better utility.

The results confirm the accuracy of the manually implemented DPSGD framework and align with theoretical expectations. This demonstrates the framework's practicality for privacy-preserving machine learning and underscores the importance of choosing $\epsilon$ values carefully to balance privacy and utility.

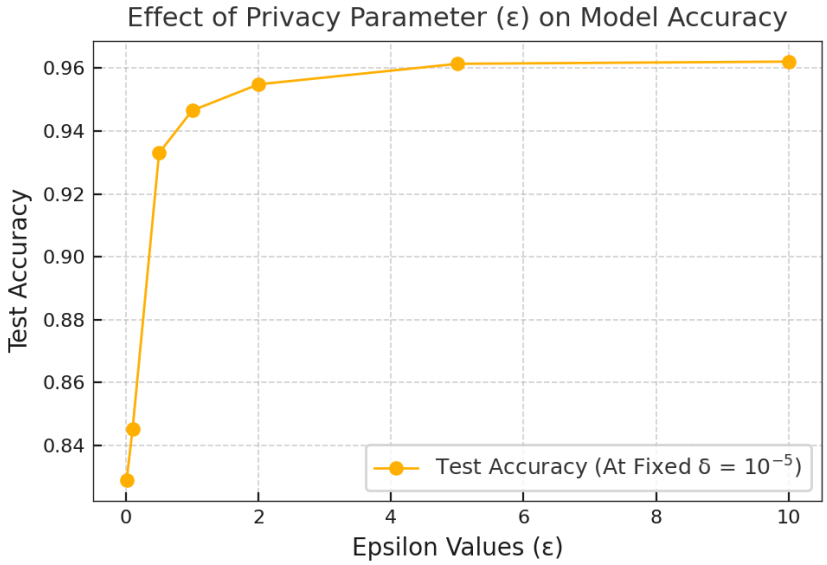


Figure 1: Effect of $\epsilon$ on Model Accuracy under Differential Privacy Constraints

## 6. Conclusion

This experiment successfully implemented and validated a DPSGD framework with manually designed Privacy Accounting and Gradient Clipping processes. The framework was evaluated on the MNIST classification task, demonstrating correctness in noise computation and gradient clipping. The observed privacy-utility trade-off highlights the practical challenges and considerations when applying differential privacy in machine learning.

## References

[1] Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. *Deep learning with differential privacy.* In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (2016), pp. 308–318.