CS 3110 Data Privacy
Final Writeup
James Bellizia, Mateo Bolen
12-08-2025

Problem Statement

We are interested in predicting the overall effective contribution of an ultimate frisbee player based on their other statistics, and comparing the efficacy of models that do and do not use differential privacy. The dataset we used has names, teams, divisions, and statistics about each player– we are concerned with how we might model a given player's contribution based on their statistics without exposing any individual player in the dataset. In the gradient descent portion of our code, we looked into how privacy-preserving gradient descent models compared to non-privacy-preserving models in terms of accuracy and loss, attempting to demonstrate how we can preserve data utility while maintaining person-level privacy. In addition to training models directly on differentially private (DP) versions of the dataset, we also generated a synthetic player dataset using DP private marginals. The goal of this approach is to provide a different method of privacy-preserving data release, while retaining approximate correlations between key player statistics.

Gradient Descent

Technical Description

The dataset we used is scoring and defensive statistics from the 2024 College Ultimate Championships. We removed the player name, team name, and plus/minus per game columns, as they are either unhelpful in our statistical analysis (names) or leak data about the label (plus/minus per game). For our features, we kept every other column (including division, turns, D's, assists, points, etc.) and removed only plus/minus, one-hot-encoding the categorical columns like division and gender. Plus/minus was then used to create a new label column– called is_positive in our code– which tells whether a given player's plus/minus score was positive (positive effect on team performance, denoted 1) or negative (negative effect on team performance, denoted -1). We then split the dataset into features (X) and labels (y, made up of only the is_positive label), and assigned a random 80% of the data to be the training set and the other 20% to be the test set. NOTE: when performing averages of model performance e.g. for average model accuracy, we re-randomize the subset of the data that is passed to the training and test sets in order to better estimate overall model accuracy.

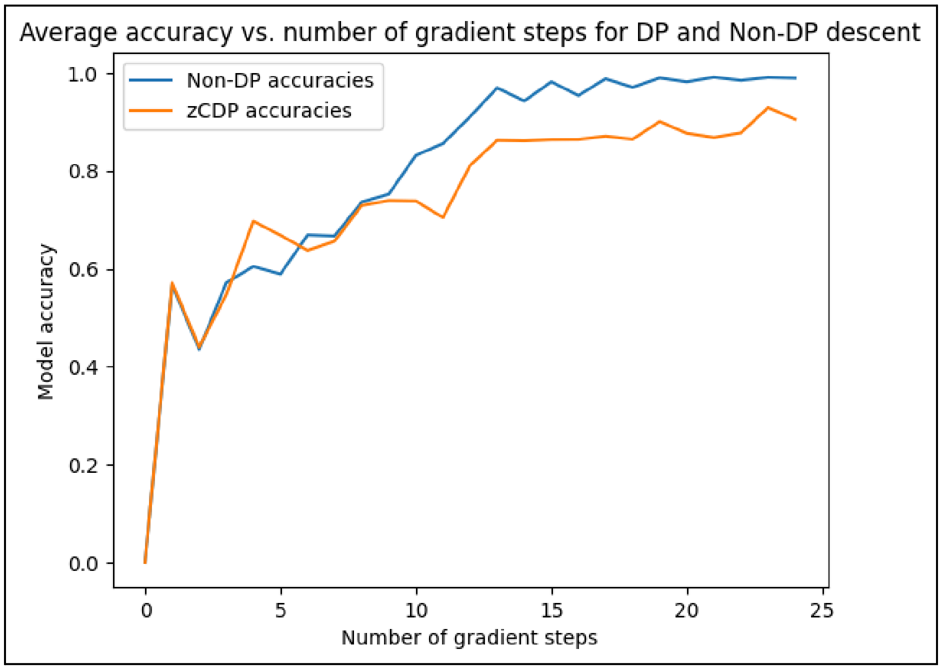
For each of our models, we used the logistic loss function, as we are simply classifying whether a given player is positive or negative. NOTE: we could have used a linear regression model to predict a real-valued plus/minus score, but we were concerned that because plus/minus directly correlates to the statistics in the dataset, there would be data leakage.

We define a gradient function that calculates the rate of change of loss for each direction, and use that to build our non-differentially-private gradient descent model, which, for a given number of iterations, subtracts the average gradient from the existing model theta position, resulting in a model that can predict with some accuracy if a player will have a positive or negative effect. We define the accuracy of a model to be the percentage of accurate model predictions.

We then define a zero-concentrated differentially private gradient descent function, which clips the gradient's L2 norm to a defined parameter b (we chose to be 15), ensuring bounded sensitivity. We then add noise to the size of the gradient vector as well as the vector sum, dividing to give us a resulting noisy average gradient, which we can subtract (as before) to obtain our next iteration of the model. We repeat this a specified number of times. NOTE: we used zero-concentrated differential privacy due to its ability to maintain tighter privacy bounds compared to epsilon, delta DP by sequential composition, as seen in Homework 9.

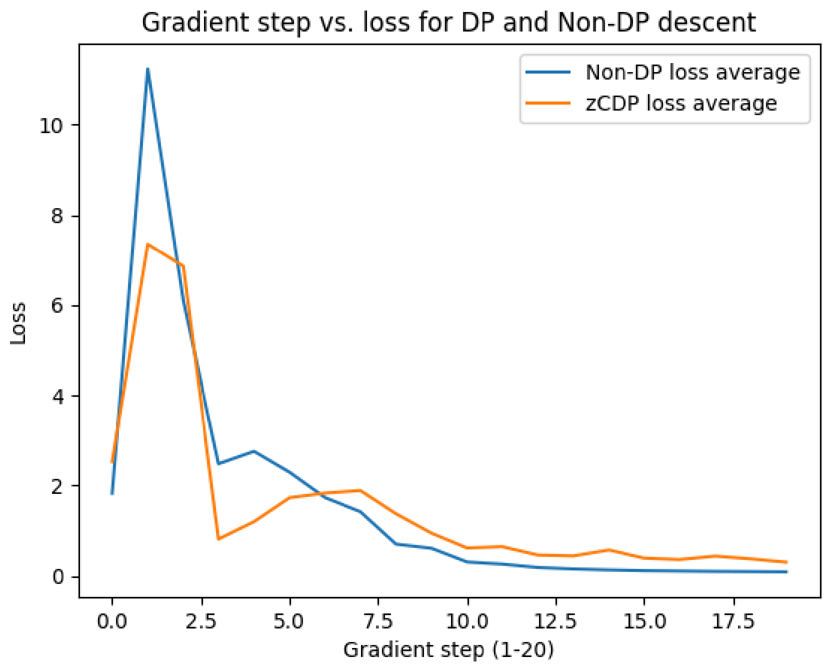
Testing / Results

We define an average accuracy function for both our non and zero-concentrated gradient descent models, which randomizes the sample of the training and test data and takes the average accuracy for a given number of gradient descent steps, using an estimate for rho of epsilon = 1, delta = 10^-5 and up to 25 gradient steps per training run. We plotted the number of gradient descent steps against the average accuracy of each model, seen below:



As we can see, the differentially private model performs quite well on average, achieving nearly 90% accuracy after 25 descent steps compared to the non-DP model's almost 100%. Of course, this dataset makes the label relatively easy to predict– we would expect that the statistics that directly inform a player's plus/minus score would also be able to inform whether a player has a positive or negative effect. In hindsight, this model is a softball even for our differentially private model– it would have been more interesting, perhaps, to drop most of the statistics columns and keep one or two as well as the categorical columns to see how well the models could predict effectiveness. Nonetheless, we see the effect of differential privacy on a model's accuracy– even on a dataset that is "easy", the differentially private model loses about 10% accuracy.

After plotting accuracy, we were interested in comparing the loss functions for each model at every iteration, so we tracked and returned loss arrays in each models and plotted loss against gradient descent for each model:



As we could expect, the loss of the non-DP model is lower (after several steps) than the DP model– we expect that because we are adding noise with every iteration, our descent will not be as perfect every time. Regardless, this graph makes it clear that on this dataset, loss remains close to 0 for both differentially private and regular models after only around 10 gradient steps, meaning these models are both highly accurate in their gradient predictions. NOTE: we think that the spike at the beginning of this graph is caused by overestimates in the initial gradient step, which quickly corrects itself after a few more steps.

Synthetic data

From the 2024 College Ultimate Championships data set that we are using, we can make a synthetic data set that correlates with each division to predict plus minus. This synthetic data set would be useful for analyzing the data of frisbee players while being differentially private.

We first compute 1-way DP marginals for categorical variables such as division. For each attribute, we calculate a histogram of observed values and add Laplace noise to each count to satisfy differential privacy. The noisy counts are clipped to non-negative values and normalized to produce a probability distribution. These probabilities are then used to sample synthetic values for each player.

To preserve relationships between attributes, we also compute two-way DP marginals for pairs of variables, such as division versus Turns, plus_minus versus is_positive, and other game statistics. Using the same Laplace mechanism, we generate noisy joint distributions that capture correlations between these attributes under the privacy constraints. For each synthetic player, we first sample the primary variable (e.g., division) from its DP 1-way marginal, then sample other attributes conditionally using the corresponding 2-way DP marginals.

Finally, we derive additional features such as turns_per_game, ds_per_game, and ast_per_game from the sampled attributes, and create the binary label is_positive based on plus_minus. The resulting synthetic dataset provides a privacy-preserving alternative to the original data while maintaining approximate statistical relationships between variables. This approach demonstrates how DP marginals can be used to generate realistic synthetic data for downstream modeling tasks.