CS 5110: Advanced Data Privacy

Multi-column Synthetic Data Creation

Jared Krogsrud

Overview:

This project is intended to attempt to create an entire synthetic dataset from any given data set. Synthetic data is often useful when one wants to allow a user to make their own queries on a data set. This allows one to bypass the need to release aggregated and differentially private data that can often come at quite an expense to the privacy budget. My attempt at doing this should work for datasets made of categorical data, it will have trouble creating synthetic data when one or more of the columns have numerical data that has a large variety of possibilities. I used my method, described below, to create a synthetic data set for the adult dataset of equal length. The primary strategy behind my approach is to create a Bayesian network to analyze the most likely causal relationships between the columns. For this, I used the **bnlearn** python library. Using this graph, I pruned it until I had something resembling a tree or a forest in the case of multiple root nodes. From the edges of the graph, I created marginals and used a similar method from class to populate all columns of the new synthetic data.

The Process:

First, several of the columns were dropped from the dataset: Name, DOB, SSN, Zip, fnlwgt. The first four were dropped as they are typically identifying information, the last I dropped because it appeared to be identifying or numerical.

After dropping the unwanted columns I fit a directed acyclic graph (DAG) in an attempt to approximate the best causal relations between the columns of the data. The easiest way to do this was to use bnlearn's structure_learning class. Using this DAG I wanted to trim edges further, particularly because some nodes had several edges pointing towards them. To use the methodology we used in class we'd need to trim excess edges. I created a function called prune_edges which essentially did this by sorting the nodes of the DAG in a topological order, and then looping through the list and cutting any edges in which the "to node" already had an edge previously pointing to it in the topological ordering.

With the trimmed DAG I created a dictionary of marginals for every edge that appeared in the graph. This is done in the create_marginals function, within the body of this function it counts how many marginals need to be made and assigns a fraction of the privacy budget to each. This is the only time the Laplace mechanism is called.

Now that we have all of our marginals, I take the highest priority edge in the DAG and create the first two columns of the data. For every subsequent column (or columns in a multiple root DAG) I add data to the dataset by checking if the "from column" of the edge already appears in the dataset, and if so filter the marginal for these two columns and sample from it as we did in our homework. In the rare case that we reach an edge in which neither of the columns has been created I generate both and merge them with the growing dataframe.

Results:

I tested my resulting synthetic data against the original dataset in three ways:

1) Comparing the counts of every value occurring in each column with the proportion that they appear in the synthetic data.

The results here started promising, most of the error was between 0 and 10% average error. The worst cases occurred for numerical data: Capital Gain and Capital Loss. I'm assuming that's because these columns are numerical data and were not grouped in a pre-processing step.

2) Calculating the average percent error of the proportion of all pairs of values from two columns in the original data set and the synthetic data.

   The percent error here ended up being quite large. The average percent error was 422.3% but this was caused by a very large outlier. The median percent error over all pairs of columns ended up being 61%. I examined what column comparisons did best and worst. The best (under 10 average error) were ('Education', 'Education-Num'), ('Education', 'Target'), ('Education-Num', 'Target') and ('Marital Status', 'Sex'). Unsurprisingly, three of these occurred as edges in the DAG. The rest occurred in the range from 10 to 50% average percent error.

3) Calculating the average percent error of the proportion of all triplets of values from three columns in the original data set and the synthetic data.

   The average error did not increase overwhelmingly as I expected from the results of comparing two columns. The median average error landed around 80% for all 3 column comparisons and at best we got 7% average error which occurred at examining the occurrences of triplets in the (Education, Education-Num, Target) columns. The worst were generally built on top of the pairs that performed the worst in part 2.

Conclusion:

While I completed my intended goal to create an entirely synthetic dataset, the amount of error that would occur while examining correlation in the synthetic data would render it somewhat useless. Were I to continue my work on this there are a few things I would attempt. The first is to pre-process the data so that numerical data is put in ranges, this might require my implementation to have the user submit a list of data types to the algorithm beforehand. The second major change is I would attempt to find a way to group parts of the DAG into clusters of high inference and create synthetic data for the clusters individually instead of the long strings of inference I attempted in my solution. I believe the error compounded every step away from the root node which caused the overall massive average error.