Multiparty Linear Regression via SPDZ

Valerie Chen Valerio Pastro Mariana Raykova

June 14, 2017

1 Preliminaries

Valerio: stuff

2 Experiments

The experiments were run on the SPDZ framework, using a 128 bit and 256 bit prime field respectively. In the 128 bit prime field, the bits were allotted with 28 bits after the decimal place to match the 32 bit example in the original paper and 60 bits after the decimal for the original 64 bit case. Compared original paper, we needed more space to the left of the decimal for comparison purposes and operations using the fixed point type. However, we believe this is still a fair comparison because we are not working with more precision after the decimal place and if anything using more bits should actually slow down our calculations. For each generated matrix of varying dimensions, the features were split evenly between two parties. We then wanted to see if linear decomposition algorithms like LDLT, Cholesky, and CGD (with 5, 10, 15, and 20 iterations) were able to get comparable if not better results.

To better fine tune the CGD and Cholesky algorithms to SPDZ, we ran a few experiments to test hypotheses for

First, we ran the two algorithms two gauge time and error rates for matrices of size 10, 20, 50, 100, 200, and 500, with 200000 features each. Secondly, we ran on a size 20 matrices with condition numbers varying from 1 to 10. Both parts were completed for both bit sizes to evaluate performance differences. To model after the original experiment, we ran iterations of 5, 10, 15, and 20 for the CGD algorithm but added on as an extension of 25 iterations to see if necessary (i.e. the errors were not comparable for all sizes or condition numbers), there was still room for improvement.

3 Result

Comparing square root for Cholesky. In SPDZ, we hypothesized that division would be time consuming, however the behavior does not seem to differ much both in terms of run time and error... Babylonian vs Newton method differ in how many divisions are used in the algorithm. From the results we can see that using different operations don't change the final result. This might be because the way that division is implemented in SPDZ..

For CGD, we decide to test if L2 norm vs Linfinity would make a difference. One utilizes comparisons and the other addition and square root.

Averaged results Averaged column 3 over 30 samples with n = 100000 (avg cn 1.77?)

Valerie: compare ranges of error between them???

N = 100000

N = 1000

Valerie: talk about outliers in these cases, skews results

As expected, utilizing more bits gives much better accuracy. In both cases, the error results are comparable to the garbled circuits paper. However, we did find that the original experiment truncated the generated data at 12 decimal places, however we modified it so that we could work with the full representation of the number to get a better understanding of how accurate our algorithm is.

The run times for SPDZ are always comparable if not much faster. In terms of run-time, we observe that both Cholesky and LDLT were faster for smaller dimensions when $d \leq 100$ and CGD is faster for larger dimensions when $d > 100$. This is consistent with the findings in the original experiment. For dimensions 100 and smaller, when Cholesky and LDLT are faster, they comparative differences are not that big (LDLT with 1.6 seconds compared to CGD 20 with 60.89 seconds) when compared to the largest differences between the larger sizes (LDLT with 1747 seconds and CGD5 with 97 seconds). It is also important to note that the time differences between 32 and 64 bits are not that large- only blank percent more compared to the garbled circuits result where doubling the number of bits roughly increased time by an entire magnitude of 10.

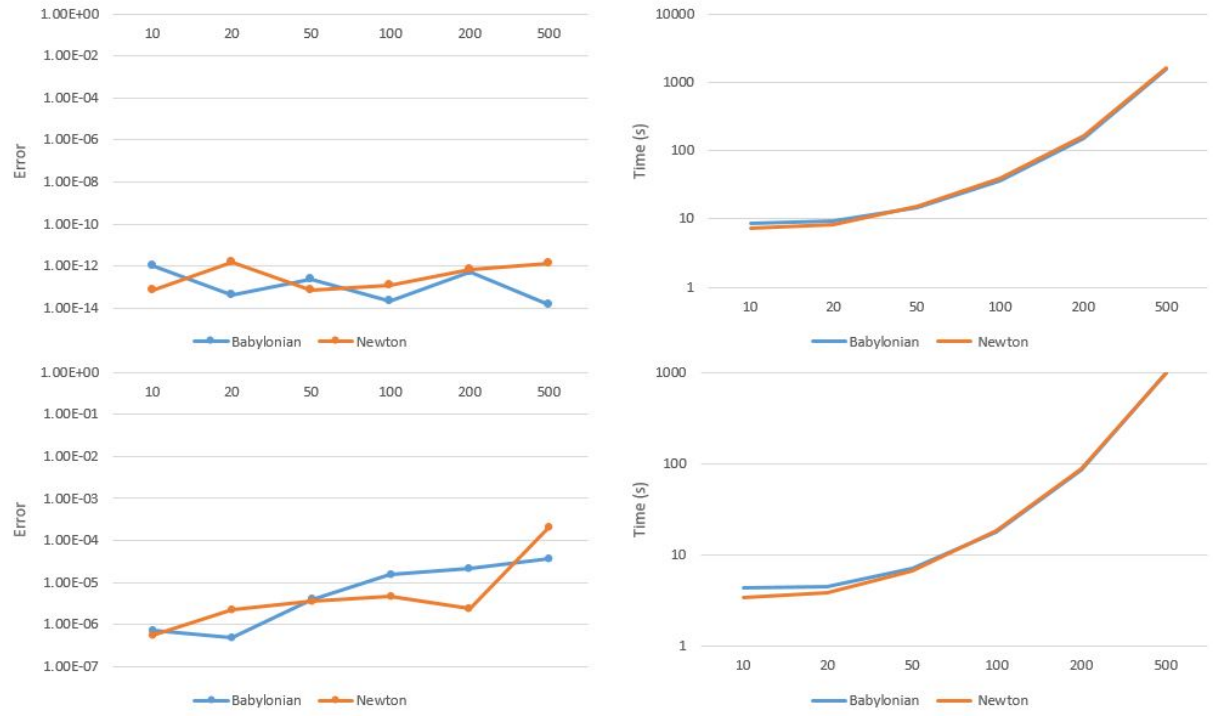

Figure 1: (Top) 64 bit (Bottom) 32 bit results

Condition number plot

Valerie: insert table comparison here??

In the condition number experiment, for CGD iterations of 5, 10, and 15, error increases as condition number increases. For example, in CGD 10 a small condition number may have an error of $10e-9$ yet a larger condition number could have an error as large as $10e-3$. However, in the CGD 20 case, the condition number does not seem to have as much of effect as the range of errors is much less., with the error varying only form $10e-11$ to $10e-14$ keeping in mind these are much smaller increments. **Valerie:** why?? Add some explanation about how condition number affects errors in a matrix For the 60 bit precision case, we chose to run more iterations because in the CGD 20 line, there is a noticeable upward slope in the scatter plot, indicating that still as condition numbers increased, the errors were increasing as well. Indeed, 25 iterations showed some improvement as we can see a flatter trend in the points.

- increasing d and increasing the condition number will result in higher errors, so you need more iterations of CGD to get comparable results. - for high condition numbers, Cholesky outperforms CGD (i.e., gets more accurate results with similar running time). - for high values of d, but low condition numbers, CGD outperforms Cholesky in both running time and accuracy (that's basically the right column in Figure 6 in the PETS paper) Regarding the correlation between n, d, and the condition number, increasing d and keeping n constant will result in a higher expected condition number. (confirmed with Phillipp)

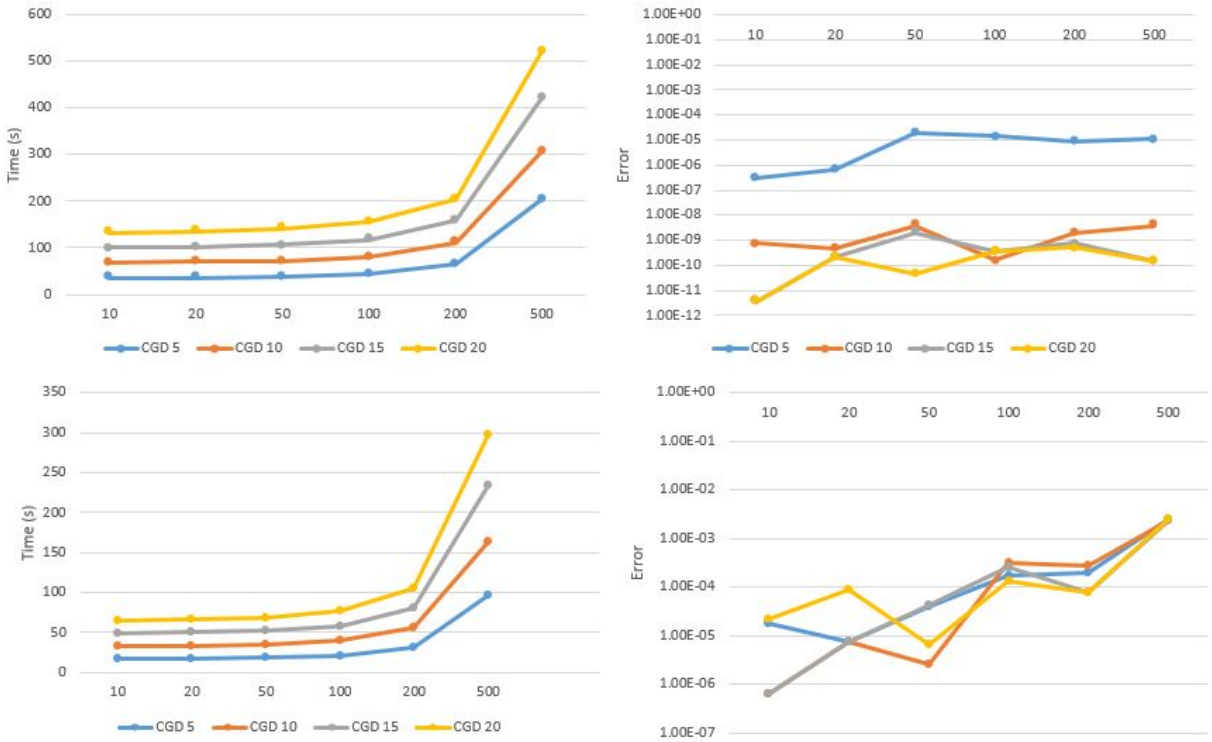

Figure 2: (Top) 64 bit (Bottom) 32 bit results

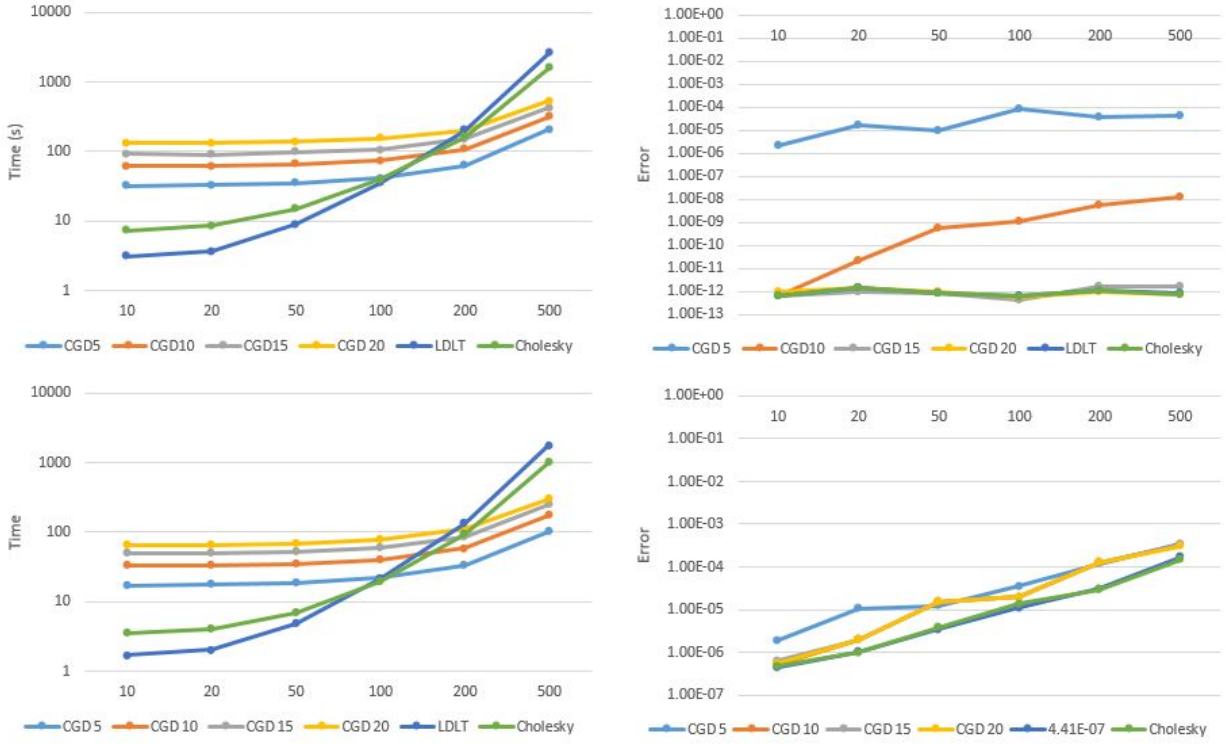

Figure 3: (Left) Comparison between different methods of solving linear systems: running time in seconds for Cholesky, LDLT, and CGD (with 5, 10, 15, and 20 iterations) as a function of input dimension. (Middle) Accuracy of CGD as a function of the input on the condition number of the input matrix A with $d = 20$. (Right) Accuracy of Cholesky, LDLT, and CGD, as a function of the input dimensionality d. (Top) Fixed-point numbers with the $b = 80$ bits, with 60 in the fractional part. (Bottom) $b = 41$, with 28 in the fractional part

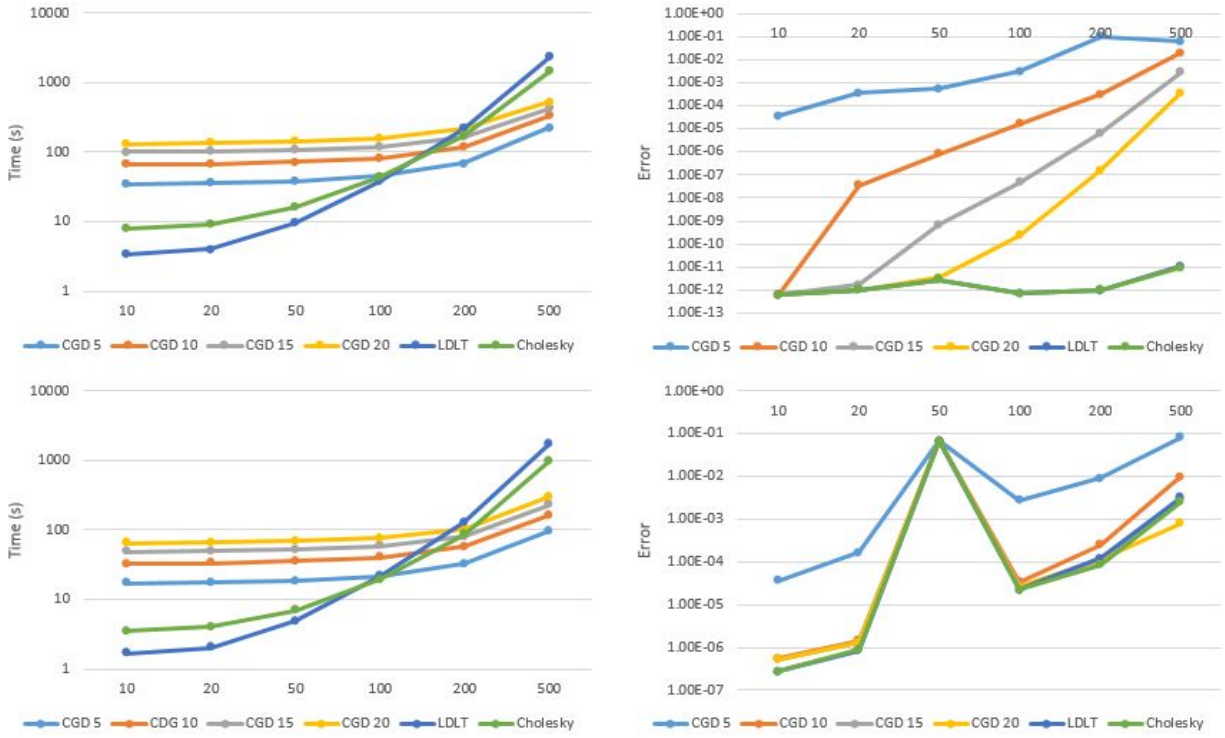

Figure 4: (Left) Comparison between different methods of solving linear systems: running time in seconds for Cholesky, LDLT, and CGD (with 5, 10, 15, and 20 iterations) as a function of input dimension. (Middle) Accuracy of CGD as a function of the input on the condition number of the input matrix A with $d = 20$. (Right) Accuracy of Cholesky, LDLT, and CGD, as a function of the input dimensionality d. (Top) Fixed-point numbers with the $b = 80$ bits, with 60 in the fractional part. (Bottom) $b = 41$, with 28 in the fractional part

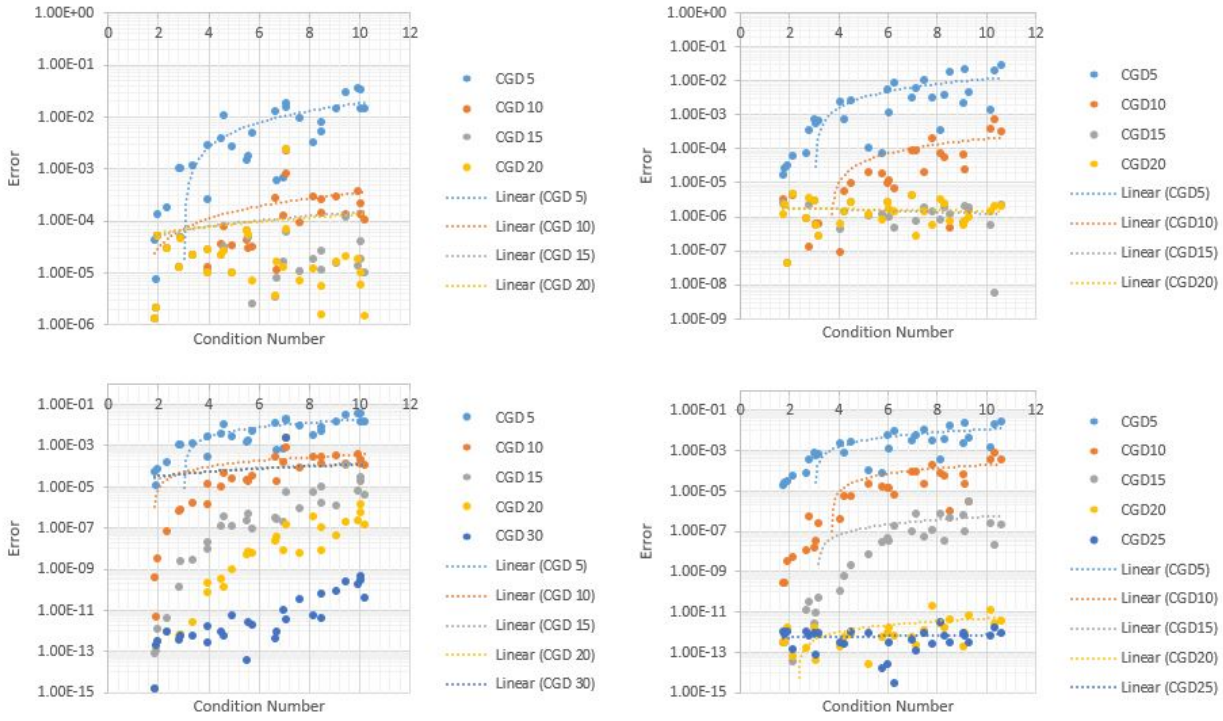

Figure 5: (Left) Comparison between different methods of solving linear systems: running time in seconds for Cholesky, LDLT, and CGD (with 5, 10, 15, and 20 iterations) as a function of input dimension. (Middle) Accuracy of CGD as a function of the input on the condition number of the input matrix A with $d = 20$. (Right) Accuracy of Cholesky, LDLT, and CGD, as a function of the input dimensionality d. (Top) Fixed-point numbers with the $b = 80$ bits, with 60 in the fractional part. (Bottom) $b = 41$, with 28 in the fractional part