Secure Multiparty Regression: Are we ready to keep data privacy promise?

Giuseppe Bruno[1]

[1]Bank of Italy

UNECE/Eurostat Work Session on Statistical Data Confidentiality.
Poznań, Poland, 1-3 December 2021

The views expressed in the presentation are the author's only and do not imply those of his Institution.

Outline

Why merging data sets from different providers?

Social scientists and researchers often need to perform statistical analyses across large, independently-owned datasets.

- it is often difficult to achieve these datasets;
- there is not a unique way to safely compute statistics on private data;
- Collaborative analyses involving private data are often limited by ethical and regulatory constraints;
- trade-offs between collective benefits and individual harms produced by processing sensitive data;

Why do we want to privately compute functions of arguments belonging to different people.

A Secure Multi-Party Computation (MPC) framework assumes a set of parties wishing to compute a joint function of their private inputs without revealing anything but the output.

- Protecting the privacy of individuals and firms is of paramount relevance in our societies.
- To obtain more accurate models, it is common that organizations cooperate to build joint training datasets;
- many examples relates economic, medical, judicial and tax records;

Why do we want to privately compute functions of arguments belonging to different people.

A Secure Multi-Party Computation (MPC) framework assumes a set of parties wishing to compute a joint function of their private inputs without revealing anything but the output.

- Protecting the privacy of individuals and firms is of paramount relevance in our societies.
- To obtain more accurate models, it is common that organizations cooperate to build joint training datasets;
- many examples relates economic, medical, judicial and tax records;

Why do we want to privately compute functions of arguments belonging to different people.

A Secure Multi-Party Computation (MPC) framework assumes a set of parties wishing to compute a joint function of their private inputs without revealing anything but the output.

- Protecting the privacy of individuals and firms is of paramount relevance in our societies.
- To obtain more accurate models, it is common that organizations cooperate to build joint training datasets;
- many examples relates economic, medical, judicial and tax records;

Why do we want to privately compute functions of arguments belonging to different people.

A Secure Multi-Party Computation (MPC) framework assumes a set of parties wishing to compute a joint function of their private inputs without revealing anything but the output.

- Protecting the privacy of individuals and firms is of paramount relevance in our societies.
- To obtain more accurate models, it is common that organizations cooperate to build joint training datasets;
- many examples relates economic, medical, judicial and tax records;

Why do we want to privately compute functions of arguments belonging to different people.

A Secure Multi-Party Computation (MPC) framework assumes a set of parties wishing to compute a joint function of their private inputs without revealing anything but the output.

- Protecting the privacy of individuals and firms is of paramount relevance in our societies.
- To obtain more accurate models, it is common that organizations cooperate to build joint training datasets;
- many examples relates economic, medical, judicial and tax records;

Envisaged social benefit

Possible social benefits from sharing otherwise private databases:

- Different hospitals could improve their medical analytics for better healthcare delivery.
- State tax authority would like to check banking relationships with suspect tax evader.
- National law enforcement bodies of different countries would like to compare their respective database of suspected terrorists.

Envisaged social benefit

Possible social benefits from sharing otherwise private databases:

- Different hospitals could improve their medical analytics for better healthcare delivery.
- State tax authority would like to check banking relationships with suspect tax evader.
- National law enforcement bodies of different countries would like to compare their respective database of suspected terrorists.

Envisaged social benefit

Possible social benefits from sharing otherwise private databases:

- Different hospitals could improve their medical analytics for better healthcare delivery.
- State tax authority would like to check banking relationships with suspect tax evader.
- National law enforcement bodies of different countries would like to compare their respective database of suspected terrorists.

Oblivious computing

Obliv-C is an extension of the C programming language that provides data-oblivious programming constructs which allow to embed secure computation protocols

- Obliv-C provides a framework to write suitable protocols for secure Multiparty computation;
- given two parties owning private data x and y;
- joint functionality $\mathbf{f}(x, y) = (f_1(x, y), f_2(x, y))$;
- the first party receives only $f_1(x, y)$ while the second receives only $f_2(x, y)$;

Oblivious computing

Obliv-C is an extension of the C programming language that provides data-oblivious programming constructs which allow to embed secure computation protocols

- Obliv-C provides a framework to write suitable protocols for secure Multiparty computation;
- given two parties owning private data x and y;
- joint functionality $\mathbf{f}(x, y) = (f_1(x, y), f_2(x, y))$;
- the first party receives only $f_1(x, y)$ while the second receives only $f_2(x, y)$;

Oblivious computing

Obliv-C is an extension of the C programming language that provides data-oblivious programming constructs which allow to embed secure computation protocols

- Obliv-C provides a framework to write suitable protocols for secure Multiparty computation;
- given two parties owning private data x and y;
- joint functionality $\mathbf{f}(x, y) = (f_1(x, y), f_2(x, y))$;
- the first party receives only $f_1(x, y)$ while the second receives only $f_2(x, y)$;

Oblivious computing

Obliv-C is an extension of the C programming language that provides data-oblivious programming constructs which allow to embed secure computation protocols

- Obliv-C provides a framework to write suitable protocols for secure Multiparty computation;
- given two parties owning private data x and y;
- joint functionality $\mathbf{f}(x, y) = (f_1(x, y), f_2(x, y))$;
- the first party receives only $f_1(x, y)$ while the second receives only $f_2(x, y)$;

Enabling oblivious computation with conditionals

An oblivious conditional statement requires code adjustments for execution:

$$\textbf{obliv if}(x > y)x = y;$$

The oblivious nature of x and y prevents the knowledge of the truth value $x > y$ logical condition.
How can we achieve obliviousness with conditional statements?

Enabling oblivious computation with conditionals

An oblivious conditional statement requires code adjustments for execution:

$$\textbf{obliv if}(x > y)x = y;$$

The oblivious nature of x and y prevents the knowledge of the truth value $x > y$ logical condition.
How can we achieve obliviousness with conditional statements?

Oblivious computation with conditionals

The conditional statements have to be converted using assignment statement adjustments for execution.
Here we see a possible avenue for a suitable transformation:

$$\begin{aligned} \mathbf{cond} &= (\mathbf{x} > \mathbf{y}); \\ \mathbf{x} &= \mathbf{x} + \mathbf{cond} \cdot (\mathbf{y} - \mathbf{x}); \end{aligned} \tag{1}$$

With this transformation we have removed any control flow dependency

Oblivious computation with conditionals

The conditional statements have to be converted using assignment statement adjustments for execution.
Here we see a possible avenue for a suitable transformation:

$$\begin{aligned} \mathbf{cond} &= (\mathbf{x} > \mathbf{y}); \\ \mathbf{x} &= \mathbf{x} + \mathbf{cond} \cdot (\mathbf{y} - \mathbf{x}); \end{aligned} \tag{1}$$

With this transformation we have removed any control flow dependency

Oblivious computation with conditionals

The conditional statements have to be converted using assignment statement adjustments for execution.
Here we see a possible avenue for a suitable transformation:

$$\begin{aligned} \mathbf{cond} &= (\mathbf{x} > \mathbf{y}); \\ \mathbf{x} &= \mathbf{x} + \mathbf{cond} \cdot (\mathbf{y} - \mathbf{x}); \end{aligned} \tag{1}$$

With this transformation we have removed any control flow dependency

The first linear model

In these examples we have employed linear multivariate models. All the data have been synthetically generated using different random distribution.

$$\begin{aligned} \mathbf{x_1} &\leftarrow \mathbf{rexp(n)} \\ \mathbf{x_2} &\leftarrow \mathbf{rnorm(n, mean = 0, sd = 1)} \\ \mathbf{x_3} &\leftarrow \mathbf{rnorm(n, mean = 3, sd = 1.5)} \end{aligned} \tag{2}$$

$$y_v = 1.5 + 0.5 \cdot x_1 + 0.2 \cdot x_2 + 0.1 \cdot x_3 + \varepsilon$$

Here ε is an added noise with gaussian distribution $\hat{\varepsilon} = 0$ and $\sigma_\varepsilon = 1$.

The first linear model

In these examples we have employed linear multivariate models. All the data have been synthetically generated using different random distribution.

$$\begin{aligned} \mathbf{x_1} &\leftarrow \mathbf{rexp(n)} \\ \mathbf{x_2} &\leftarrow \mathbf{rnorm(n, mean = 0, sd = 1)} \\ \mathbf{x_3} &\leftarrow \mathbf{rnorm(n, mean = 3, sd = 1.5)} \end{aligned} \tag{2}$$

$$y_v = 1.5 + 0.5 \cdot x_1 + 0.2 \cdot x_2 + 0.1 \cdot x_3 + \varepsilon$$

Here ε is an added noise with gaussian distribution $\hat{\varepsilon} = 0$ and $\sigma_\varepsilon = 1$.

The first linear model

In these examples we have employed linear multivariate models. All the data have been synthetically generated using different random distribution.

$$\begin{aligned} \mathbf{x_1} &\leftarrow \mathbf{rexp(n)} \\ \mathbf{x_2} &\leftarrow \mathbf{rnorm(n, mean = 0, sd = 1)} \\ \mathbf{x_3} &\leftarrow \mathbf{rnorm(n, mean = 3, sd = 1.5)} \end{aligned} \tag{2}$$

$$y_v = 1.5 + 0.5 \cdot x_1 + 0.2 \cdot x_2 + 0.1 \cdot x_3 + \varepsilon$$

Here ε is an added noise with gaussian distribution $\hat{\varepsilon} = 0$ and $\sigma_\varepsilon = 1$.

The first linear model

In these examples we have employed linear multivariate models. All the data have been synthetically generated using different random distribution.

$$\begin{aligned} \mathbf{x_1} &\leftarrow \mathbf{rexp(n)} \\ \mathbf{x_2} &\leftarrow \mathbf{rnorm(n, mean = 0, sd = 1)} \\ \mathbf{x_3} &\leftarrow \mathbf{rnorm(n, mean = 3, sd = 1.5)} \end{aligned} \tag{2}$$

$$y_v = 1.5 + 0.5 \cdot x_1 + 0.2 \cdot x_2 + 0.1 \cdot x_3 + \varepsilon$$

Here ε is an added noise with gaussian distribution $\hat{\varepsilon} = 0$ and $\sigma_\varepsilon = 1$.

The second linear model

This second model requires the estimation of a ten parameters multivariate models.

This example allows us to check whether some nonlinear effect pops out when increasing the number of regressors.

$$\begin{aligned} y_z &= 0.7 + 0.5 \cdot x_1 + 0.2 \cdot x_2 + 0.1 \cdot x_3 + 0.9 \cdot x_4 + 0.1 \cdot x_5 + \\ &\quad -1.2 \cdot x_6 - 0.25 \cdot x_7 + 0.8 \cdot x_8 + 1.5 \cdot x_9 + \varepsilon \end{aligned}$$

The exogenous variables are randomly generated as in the case of the first model.

The second linear model

This second model requires the estimation of a ten parameters multivariate models.
This example allows us to check whether some nonlinear effect pops out when increasing the number of regressors.

$$\begin{aligned} y_z &= 0.7 + 0.5 \cdot x_1 + 0.2 \cdot x_2 + 0.1 \cdot x_3 + 0.9 \cdot x_4 + 0.1 \cdot x_5 + \\ &\quad -1.2 \cdot x_6 - 0.25 \cdot x_7 + 0.8 \cdot x_8 + 1.5 \cdot x_9 + \varepsilon \end{aligned}$$

The exogenous variables are randomly generated as in the case of the first model.

The second linear model

This second model requires the estimation of a ten parameters multivariate models.
This example allows us to check whether some nonlinear effect pops out when increasing the number of regressors.

$$\begin{aligned} y_z &= 0.7 + 0.5 \cdot x_1 + 0.2 \cdot x_2 + 0.1 \cdot x_3 + 0.9 \cdot x_4 + 0.1 \cdot x_5 + \\ &\quad -1.2 \cdot x_6 - 0.25 \cdot x_7 + 0.8 \cdot x_8 + 1.5 \cdot x_9 + \varepsilon \end{aligned}$$

The exogenous variables are randomly generated as in the case of the first model.

Perfomance comparisons

We have estimated the two models with four datasets having a growing size.
We employed 10^3, 10^4, 10^5, 10^6 observations and we measured the execution time for the oblivious regressions. The following table shows the timing for the first model:

Δt	number of observations			
	10^3	10^4	10^5	10^6
user	$22.3''$	$3'35''$	$36'35''$	$358'26''$
system	$5.9''$	$1'1''$	$9'29''$	$83'10''$

timing results for the private regression model with four independent variables.

Perfomance comparisons

We have estimated the two models with four datasets having a growing size.
We employed 10^3, 10^4, 10^5, 10^6 observations and we measured the execution time for the oblivious regressions. The following table shows the timing for the first model:

Δt	**number of observations**			
	10^3	10^4	10^5	10^6
user	$22.3''$	$3'35''$	$36'35''$	$358'26''$
system	$5.9''$	$1'1''$	$9'29''$	$83'10''$

timing results for the private regression model with four independent variables.

Perfomance comparisons: second model

Also for the second model we employed the same set of number of observations. The following table shows the timing for the second model:

Δt	number of observations			
	10^3	10^4	10^5	10^6
user	2′34″	26′24″	196′24″	2066′26″
system	30.3″	4′58″	45′22″	536′2″
performances on the AWS Platinum 8124M				
user	1′19″	13′14″	133′34″	1299′42″
system	13.7″	2′20″	29′30″	295′52″

timing results for the private regression model with ten independent variables.

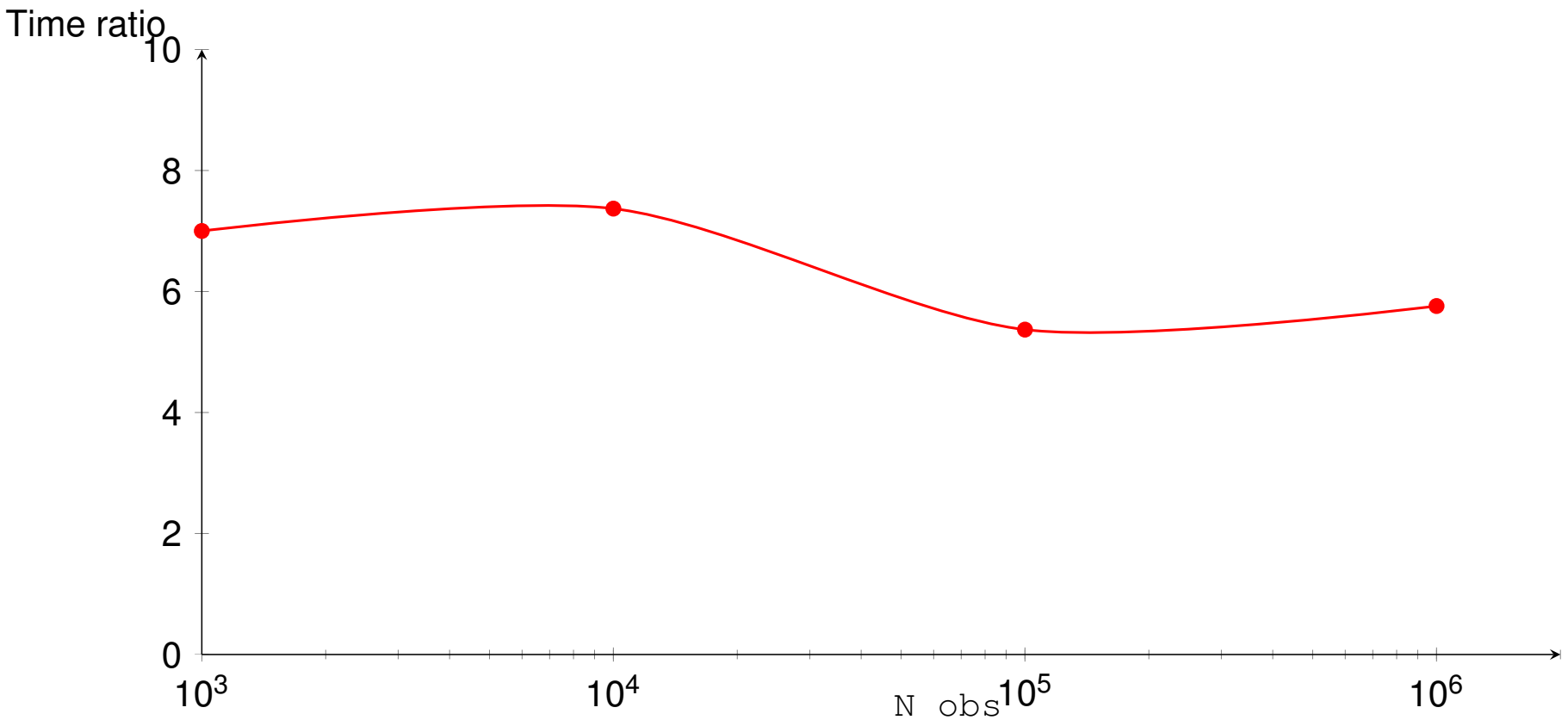
Perfomance comparisons: second model

Also for the second model we employed the same set of number of observations. The following table shows the timing for the second model:

Δt	**number of observations**			
	10^3	10^4	10^5	10^6
user	$2'34''$	$26'24''$	$196'24''$	$2066'26''$
system	$30.3''$	$4'58''$	$45'22''$	$536'2''$
performances on the AWS Platinum 8124M				
user	$1'19''$	$13'14''$	$133'34''$	$1299'42''$
system	$13.7''$	$2'20''$	$29'30''$	$295'52''$

timing results for the private regression model with ten independent variables.

Performance behaviour according to the dataset size



Estimation time ratio between the model with ten and four explanatory variables.

Accuracy comparison between Obliv-C and R lm

Coefficients	R-estimate	Obliv-C
Intercept	0.7007754	0.4890862
x1	0.5013639	0.4854234
x2	0.3983556	0.4025301
x3	0.1010488	0.0941359
x4	0.8995275	0.8804412
x5	0.0994992	0.05695939
x6	1.1998318	1.230985
x7	0.2498331	0.2374886
x8	0.8000125	0.8254486
x9	1.5004663	1.563147

Coefficient estimates with the R package and Obliv-C.

Concluding Remarks

- we have shown a possible avenue for carrying out a linear multivariate regression by extending the Obliv-C framework;
- the overhead caused by oblivious regressions grows approximately linearly up to 1 million of observations;
- estimation time grow more than quadratically with the number of explanatory variables;
- it seems compelling to introduce the double precision oblivious data type;
- code parallelization is a mandatory requirement to achieve a reasonable performances with huge micro datasets;

Concluding Remarks

- we have shown a possible avenue for carrying out a linear multivariate regression by extending the Obliv-C framework;
- the overhead caused by oblivious regressions grows approximately linearly up to 1 million of observations;
- estimation time grow more than quadratically with the number of explanatory variables;
- it seems compelling to introduce the double precision oblivious data type;
- code parallelization is a mandatory requirement to achieve a reasonable performances with huge micro datasets;

Concluding Remarks

- we have shown a possible avenue for carrying out a linear multivariate regression by extending the Obliv-C framework;
- the overhead caused by oblivious regressions grows approximately linearly up to 1 million of observations;
- estimation time grow more than quadratically with the number of explanatory variables;
- it seems compelling to introduce the double precision oblivious data type;
- code parallelization is a mandatory requirement to achieve a reasonable performances with huge micro datasets;

Concluding Remarks

- we have shown a possible avenue for carrying out a linear multivariate regression by extending the Obliv-C framework;
- the overhead caused by oblivious regressions grows approximately linearly up to 1 million of observations;
- estimation time grow more than quadratically with the number of explanatory variables;
- it seems compelling to introduce the double precision oblivious data type;
- code parallelization is a mandatory requirement to achieve a reasonable performances with huge micro datasets;

Concluding Remarks

- we have shown a possible avenue for carrying out a linear multivariate regression by extending the Obliv-C framework;
- the overhead caused by oblivious regressions grows approximately linearly up to 1 million of observations;
- estimation time grow more than quadratically with the number of explanatory variables;
- it seems compelling to introduce the double precision oblivious data type;
- code parallelization is a mandatory requirement to achieve a reasonable performances with huge micro datasets;

For Further Reading


S. Zahur and D. Evans.
Obliv-C: A Language for Extensible Data-Oblivious Computation.
International Conference on Machine Learning, 2005.


O. Goldreich, S. Micali and A. Wigderson.
How to Play any Mental Game or a Complete theorem for protocols with honest majority.
ACM Symposium on the Theory of Computing, 2003.


A. C. C. Yao.
How to Generate and Exchange Secrets.
IEEE Symposium on Foundation of Computer Science, 1986.

Thank you very much for your attention.

Dziękuję bardzo za uwagę.

Merci beaucoup pour votre attention.

Questions?