Privacy-Preserving Ridge Regression Without Garbled Circuits

Marc Joye

NXP Semiconductors, San Jose, USA
marc.joye@nxp.com

Abstract. Ridge regression is an algorithm that takes as input a large number of data points and finds the best-fit linear curve through these points. It is a building block for many machine-learning operations. This report presents a system for privacy-preserving ridge regression. The system outputs the best-fit curve in the clear, but exposes no other information about the input data.
This problem was elegantly addressed by Nikolaenko *et al.* (S&P 2013). They suggest an approach that combines homomorphic encryption and Yao garbled circuits. The solution presented in this report only involves homomorphic encryption. This improves the performance as Yao circuits were the main bottleneck in the previous solution.

Keywords: Privacy; Homomorphic encryption; Ridge regression; Machine learning.

1 Learning a Model

We briefly review ridge regression, the algorithm that an analyst executes to learn the model, $\vec{\beta}$. All results discussed below are classic, and can be found in most statistics and machine learning textbooks (*e.g.*, [2]).

1.1 Linear Regression

Given a set of n input variables $\vec{x}_i \in \mathbb{R}^d$, and a set of output variables $y_i \in \mathbb{R}$, the problem of learning a function $f : \mathbb{R}^d \to \mathbb{R}$ such that $y_i \simeq f(\vec{x}_i)$ is known as *regression*. For example, the input variables could be a person's age, weight, body mass index, *etc.*, while the output can be their likelihood to contract a disease.

Learning such a function from real data has many interesting applications, that make regression ubiquitous in data mining, statistics, and machine learning. On one hand, the function itself can be used for *prediction*, *i.e.*, to predict the output value y of a new input $\vec{x} \in \mathbb{R}^d$. On the other hand, the structure of f can aid in identifying how different inputs affect the output —establishing, *e.g.*, that weight, rather than age, is more strongly correlated to a disease.

Linear regression is based on the premise that f is well approximated by a linear map,[1] *i.e.*,

$$y_i \simeq \vec{\beta}^\intercal \, \vec{x}_i \,, \qquad i \in \{1, \ldots, n\}$$

for some $\vec{\beta} \in \mathbb{R}^d$. Linear regression is one of the most widely used methods for inference and statistical analysis in the sciences. In addition, it is a fundamental building block for several more advanced methods in statistical analysis and machine learning, such as kernel methods. For example, learning a function that is a polynomial of degree 2 reduces to linear regression over $x_{ik}\, x_{ik'}$, for $1 \le k, k' \le d$; the same principle can be generalized to learn any function spanned by a finite set of basis functions.

As mentioned above, beyond its obvious uses for prediction, the vector $\vec{\beta} = (\beta_k)_{k=1,\ldots,d}$ is interesting as it reveals how y depends on the input variables. In particular, the sign of a coefficient β_k indicates either positive or negative correlation to the output, while the magnitude captures the relative importance. To ensure these coefficients are comparable, but also for numerical stability, the inputs $\vec{x}_i$ are rescaled to the same, finite domain (*e.g.*, $[-1, 1]$).

[1] Affine maps can also be captured through linear regression, by appending a 1 to each of the inputs.

1.2 Computing the Coefficients

To compute the vector $\vec{\beta} \in \mathbb{R}^d$, the latter is fit to the data by minimizing the following quadratic function over $\mathbb{R}^d$:

$$F(\vec{\beta}) = \sum_{i=1}^{n} (y_i - \vec{\beta}^\intercal \, \vec{x}_i)^2 + \lambda \, \|\vec{\beta}\|_2^2 \quad . \tag{1}$$

The procedure of minimizing Eq. (1) is called *ridge regression*; the objective $F(\vec{\beta})$ incorporates a penalty term $\lambda \, \|\beta\|_2^2$, which favors parsimonious solutions. Intuitively, for $\lambda = 0$, minimizing Eq. (1) corresponds to solving a simple least-squares problem. For positive $\lambda > 0$, the term $\lambda \, \|\vec{\beta}\|_2^2$ penalizes solutions with high norm: between two solutions that fit the data equally, one with fewer large coefficients is preferable. Recalling that the coefficients of $\vec{\beta}$ are indicators of how input affects output, this acts as a form of "Occam's razor": simpler solutions, with few large coefficients, are preferable. Indeed, a $\lambda > 0$ gives in practice better predictions over new inputs than the least-squares based solution.

Let $\vec{y} \in \mathbb{R}^n$ be the vector of outputs and $X \in \mathbb{R}^{n \times d}$ be a matrix comprising the input vectors, one in each row: $\vec{y} = (y_i)_{i=1,\ldots,n}$ and $X = (\vec{x}_i{}^\intercal)_{i=1,\ldots,n}$; *i.e.*,

$$\vec{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \quad \text{and} \quad X = \begin{pmatrix} x_{11} & x_{12} & \ldots & x_{1d} \\ x_{21} & x_{22} & \ldots & x_{2d} \\ \vdots & \vdots & & \vdots \\ x_{n1} & x_{n2} & \ldots & x_{nd} \end{pmatrix} \quad .$$

The minimizer of Eq. (1) can be computed by solving the linear system

$$A\vec{\beta} = \vec{b} \tag{2}$$

where

$$A = X^\intercal X + \lambda I \in \mathbb{R}^{d \times d} \quad \text{and} \quad \vec{b} = X^\intercal \vec{y} \in \mathbb{R}^d \quad .$$

For $\lambda > 0$, the matrix A is symmetric positive definite, and an efficient solution can be found using the Cholesky decomposition.

Credits and Priority dates. The techniques presented in this report were mostly developed in 2014 while the author was with Technicolor, USA. Technicolor filed a patent application in August 2015.

2 Privacy-Preserving Ridge Regression

As described in the previous section, given a large number of points in high dimension, the regression algorithm produces a best-fit linear curve through these points. Our goal is to perform this computation without exposing any other information about user data.

We target the same system as the one considered in [3]; see Fig. 1. Users send their encrypted data to a party called the *Evaluator* who runs the learning algorithm. At certain points the Evaluator may interact with a *Crypto Service Provider* (CSP), who is trusted not to collude with the Evaluator. The final outcome for the Evaluator is the predictive model in the clear.

The solution offered in [3] is a hybrid method to privacy-preserving ridge regression. It uses both homomorphic encryption (*e.g.*, Paillier's scheme [4]) and Yao garbled circuits [7]. The goal of the paper is to build a solution that only makes use of homomorphic encryption.

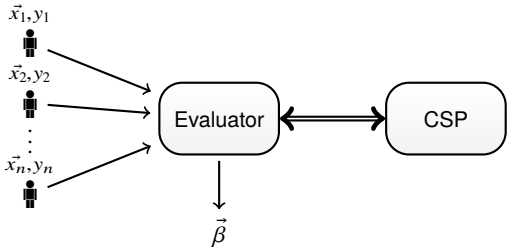

Fig. 1. The parties in the system. The Evaluator learns $\vec{\beta}$, the model describing the best linear curve fit to the data $(\vec{x}_i, y_i), i = 1, \ldots, n$, without seeing the data in the clear.

2.1 Architecture and Entities

The system is designed for many users to contribute data to a central server called the *Evaluator*. The Evaluator performs *regression* over the contributed data and produces a *model*, which can later be used for prediction or recommendation tasks. More specifically, each user $i = 1, \ldots, n$ has a private record comprising two variables $\vec{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$, and the Evaluator wishes to compute a $\vec{\beta} \in \mathbb{R}^d$ —the model— such that $y_i \simeq \vec{\beta}^\intercal \vec{x}_i$.

The goal is to ensure that the Evaluator learns nothing about the user's records beyond what is revealed by $\vec{\beta}$, the final result of the regression algorithm. To initialize the system we will need a third party, which we call a "Crypto Service Provider", that does most of its work offline.

More precisely, the parties in the system are the following (shown in Fig. 1):

- Users: each user i has private data $(\vec{x}_i, y_i)$ that it sends encrypted to the Evaluator.
- Evaluator: runs a regression algorithm on the encrypted data and obtains the learned model $\vec{\beta}$ in the clear.
- Crypto Service Provider (CSP): initializes the system by giving setup parameters to the users and the Evaluator. The CSP does most of its work offline before users contribute data to the Evaluator. In our most efficient design, the CSP is also needed for a short one-round online step when the Evaluator computes the model.

2.2 Proposed Solution

Following [3], we adopt a two-phase approach to solve in a privacy-preserving fashion the linear system given by Eq. (2); namely

$$A\vec{\beta} = \vec{b}$$

where $A = X^\intercal X + \lambda I$ and $\vec{b} = X^\intercal \vec{y}$, with $X = (\vec{x}_i{}^\intercal)_{i=1,\ldots,n}$ and $\vec{y} = (y_i)_{i=1,\ldots,n}$.

The authors of [3] astutely observe that the matrix A and vector $\vec{b}$ can be computed in an iterative fashion as follows. Recalling that each pair $(\vec{x}_i, y_i)$ is held by a distinct user, each user i can locally compute the matrix $A_i = \vec{x}_i \, \vec{x}_i{}^\intercal$ and the vector $\vec{b}_i = y_i \, \vec{x}_i$. It is then easily verified that summing these partial contributions yields:

$$A = \sum_{i=1}^{n} A_i + \lambda I \quad \text{and} \quad \vec{b} = \sum_{i=1}^{n} \vec{b}_i \ . \tag{3}$$

Equation (3) importantly shows that A and $\vec{b}$ are the result of a series of additions. The Evaluator's regression task can therefore be separated into two subtasks: (a) collecting the A_i's and $\vec{b}_i$'s to construct matrix A and vector $\vec{b}$, and (b) using these to obtain β by solving the linear system (2).

Let

$$\mathfrak{E}_{\mathsf{pk}} : (A_i; \vec{b}_i) \in \mathcal{M} \mapsto \mathfrak{c}_i = \mathfrak{E}_{\mathsf{pk}}(A_i; \vec{b}_i)$$

be a semantically secure encryption scheme indexed by a public key pk that takes as input a pair $(A_i; \vec{b}_i)$ in the message space $\mathcal{M}$ and returns the encryption $\mathfrak{c}_i$ of $(A_i; \vec{b}_i)$ under pk. Then it must hold for any pk and any two pairs $(A_i; \vec{b}_i)$, $(A_j; \vec{b}_j)$, that

$$\mathfrak{E}_{\mathsf{pk}}(A_i; \vec{b}_i) \otimes \mathfrak{E}_{\mathsf{pk}}(A_j; \vec{b}_j) = \mathfrak{E}_{\mathsf{pk}}(A_i + A_j; \vec{b}_i + \vec{b}_j)$$

for some public binary operator $\otimes$.

Such an encryption scheme can be constructed from any semantically secure additively homomorphic encryption scheme by encrypting component-wise the entries of A_i and $\vec{b}_i$. We suppose that $\mathcal{M} = \mathbb{Z}/N\mathbb{Z}$ is the message space associated to this additively homomorphic encryption scheme. This setting covers all practical known schemes, including Paillier's scheme [4].

Phase 1.

1.1 Each user i ($1 \leq i \leq n$) contributes locally and computes $A_i = \vec{x}_i \, \vec{x}_i{}^{\intercal}$ and $\vec{b}_i = y_i \, \vec{x}_i$. The values of A_i and $\vec{b}_i$ are encrypted under the CSP's public key pk_{csp},

$$\mathfrak{c}_i = \mathfrak{E}_{\mathsf{pk}_{\text{csp}}}(A_i; \vec{b}_i),$$

and sent to the Evaluator.

1.2 The Evaluator computes $\mathfrak{c}_\lambda = \mathfrak{E}_{\mathsf{pk}_{\text{csp}}}(\lambda I; \vec{0})$. It then aggregates it to all the ciphertexts $\mathfrak{c}_i$ and obtains the encryted matrix A and the encrypted vector $\vec{b}$; namely,

$$\begin{aligned} \mathfrak{c} &:= \left(\bigotimes_{i=1}^{n} \mathfrak{c}_i \right) \otimes \mathfrak{c}_\lambda = \mathfrak{E}_{\mathsf{pk}_{\text{csp}}} \left(\sum_{i=1}^{n} A_i + \lambda I; \sum_{i=1}^{n} \vec{b}_i \right) \\ &= \mathfrak{E}_{\mathsf{pk}_{\text{csp}}}(A; \vec{b}) \end{aligned}$$

The challenge now is to solve Eq. (2), with the help of the CSP, without revealing (to the Evaluator or the CSP) any additional information other than $\vec{\beta}$.

Phase 2.

2.1 The Evaluator chooses uniformly at random two matrices $R \in \mathsf{GL}(d, \mathbb{Z}/N\mathbb{Z})$ and $S \in \mathsf{GL}(d, \mathbb{Z}/N\mathbb{Z})$ and computes

$$\mathfrak{c}^\star = \mathfrak{E}_{\mathsf{pk}_{\text{csp}}}(RAS; R\vec{b}) \ .$$

The Evaluator forwards $\mathfrak{c}^\star$ to the CSP.

2.2 The CSP uses her secret key to decrypt $\mathfrak{c}^\star$ and gets $A^\star = RAS$ and $\vec{b^\star} = R\,\vec{b}$. The CSP then solves for $\vec{\beta}'$ the system

$$A^\star \vec{\beta}' = \vec{b^\star}$$

and sends $\vec{\beta}'$ to the Evaluator.

2.3 The Evaluator finally recovers $\vec{\beta}$ as $\vec{\beta} = S\vec{\beta}'$.

Remark 1. Step 1 (in Phase 2) requires to compute matrix multiplications in the encrypted domain. This can be done by observing that the components of matrices R and S are known to the Evaluator. For example, to evaluate element (u, v) of $\mathfrak{E}_{\mathsf{pk}_{\mathrm{csp}}}(RA)$, the Evaluator computes

$$\prod_{1 \le j \le d} \mathfrak{E}_{\mathsf{pk}_{\mathrm{csp}}}(A_{j,v})^{R_{u,j}}$$

which is equal to $\mathfrak{E}_{\mathsf{pk}_{\mathrm{csp}}}(\sum_{1 \le j \le d} R_{u,j} A_{j,v})$, as desired.

3 Implementation

The resolution of the linear system given by Eq. (2) requires to accurately represent real numbers in a binary form. As in [3], we consider a fixed-point number representation. So, by scaling both sides of Eq. (2), we can without loss of generality assume that the entries of matrix A and of vector $\vec{b}$ have integer entries represented with a fixed number of bits, say κ bits.

In order to apply the rational reconstruction to vector $\vec{\beta}$ (as the outcome of Step 2.3), seen as a vector in $\mathbb{Q}^d$, we need to bound the numerator and denominator of its entries. According to [6,1], letting B_1 and B_2 respectively denote the bounds on the numerator and on the denominator, one must have $2B_1B_2 \le 2^\lambda$, where λ is the bit-size of N —remember that A_i and $\vec{b_i}$ are encrypted component-wise as elements of $\mathbb{Z}/N\mathbb{Z}$.

We can write

$$\vec{\beta} = A^{-1}\,\vec{b} = \frac{1}{\det(A)}\,\mathrm{adj}(A)\,\vec{b}$$

where $\mathrm{adj}(A)$ is the adjucate of A (*i.e.*, the transpose of its cofactor matrix).

We recall that, for a matrix $M = (m_{i,j}) \in \mathbb{R}^{d\times d}$, Hadamard's inequality tells that $|\det(M)| \le \prod_{j=1}^{d} \|\vec{M}_j\|_2$ where the $\vec{M}_j$'s represent the columns of M. As a corollary, since $\|\vec{M}_j\|_2 \le \sqrt{d}\,\|\vec{M}_j\|_\infty$, we get $|\det(M)| \le (\sqrt{d}\,\mu)^d$ with $\mu = \max_{i,j} |m_{i,j}|$. Furthermore, when M is positive definite, the bound can be tightened to $\det(M) \le \prod_{j=1}^{d} m_{j,j} \le \mu^d$.

Since matrix A is positive definite, we have

$$B_2 \le \det(A) \le (2^\kappa)^d \quad .$$

For the numerator, Hadamard's inequality yields

$$B_1 \le \|\,\mathrm{adj}(A)\,\vec{b}\|_\infty \le d \cdot \left(\left(\sqrt{(d-1)}\,2^\kappa\right)^{d-1} \cdot 2^\kappa\right) \le d^{(d+1)/2}\,(2^\kappa)^d \quad .$$

The condition $2B_1B_2 \le 2^\lambda$ is therefore fulfilled when

$$\lambda > 2\kappa d + \tfrac{d+1}{2} \log_2 d \quad .$$

For example, using Paillier's scheme, a valid set of parameters is $|N|_2 = 1024$, $d = 20$ and $\kappa = 24$. This setting was numerically verified on a synthetic dataset with a set of routines developed with GP/Pari [5].

References

1. Pierre-Alain Fouque, Jacques Stern, and Jan-Geert Wackers. Cryptocomputing with rationals. In *Financial Cryptography*, volume 2357 of *Lecture Notes in Computer Science*, pages 136–146. Springer, 2002.
2. Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer Series in Statistics. Springer, 2nd edition, 2009.
3. Valeria Nikolaenko, Udi Weinsberg, Stratis Ioannidis, Marc Joye, Dan Boneh, and Nina Taft. Privacy-preserving ridge regression on hundred of millions of records. In *2013 IEEE Symposium on Security & Privacy (S&P 2013)*, pages 334–348. IEEE Computer Society, 2013.
4. Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 1999.
5. The PARI Group, Bordeaux. *PARI/GP version* `2.7.3`, 2015. Available from `http://pari.math.u-bordeaux.fr/`.
6. Paul S. Wand, M. J. T. Guy, and J. H. Davenport. *p*-adic reconstruction of rational numbers. *ACM SIGSAM Bulletin*, 16(2):2–3, 1982.
7. Andrew Chi-Chih Yao. How to generate and exchange secrets. In *IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 162–167. IEEE Computer Society, 1986.