Principal Component Analysis over encrypted data using Homomorphic Encryption

Hilder Vitor Lima Pereira and Professor Diego F. Aranha

July, 05 2016

Summary

Machine Learning as a Service

Amazon Machine Learning

Google Prediction API

Machine Learning as a Service

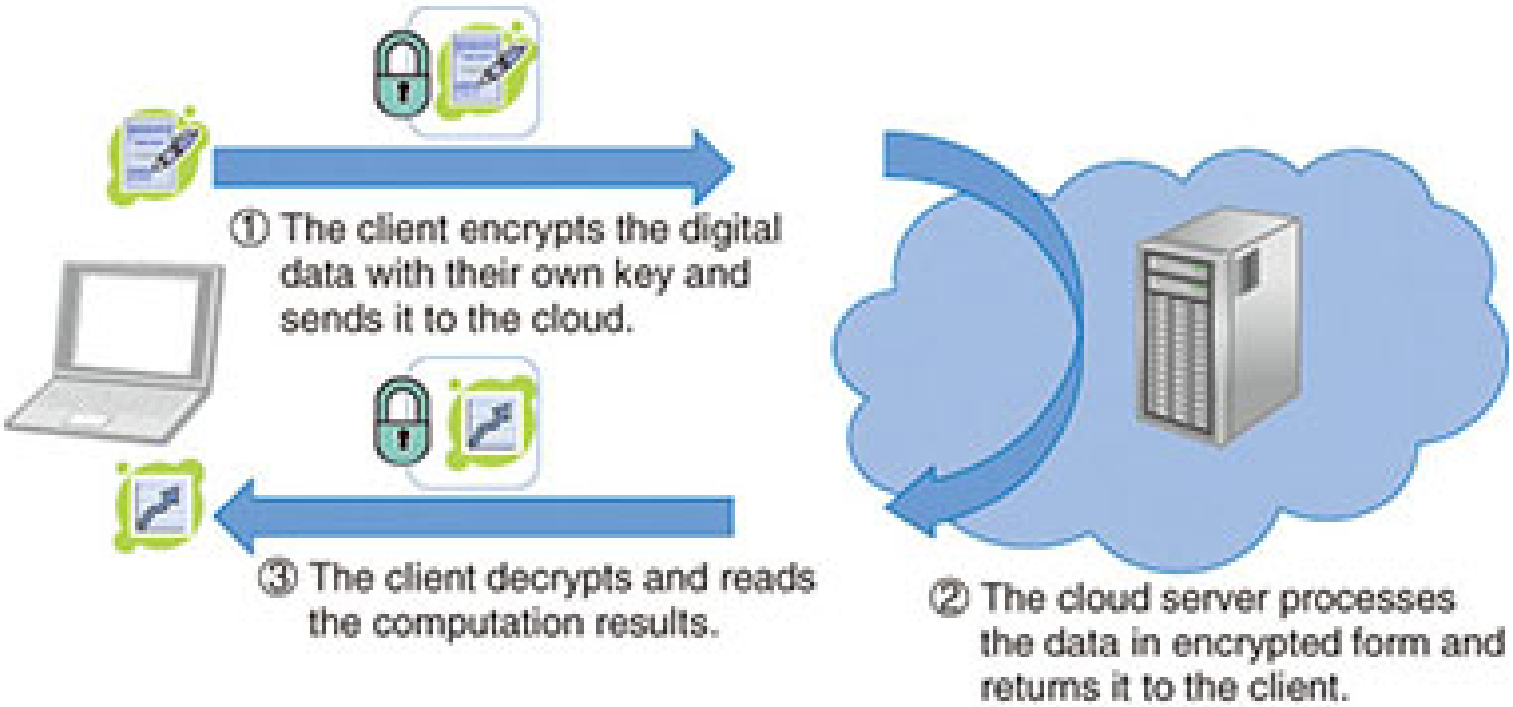
Amazon Machine Learning

Google Prediction API

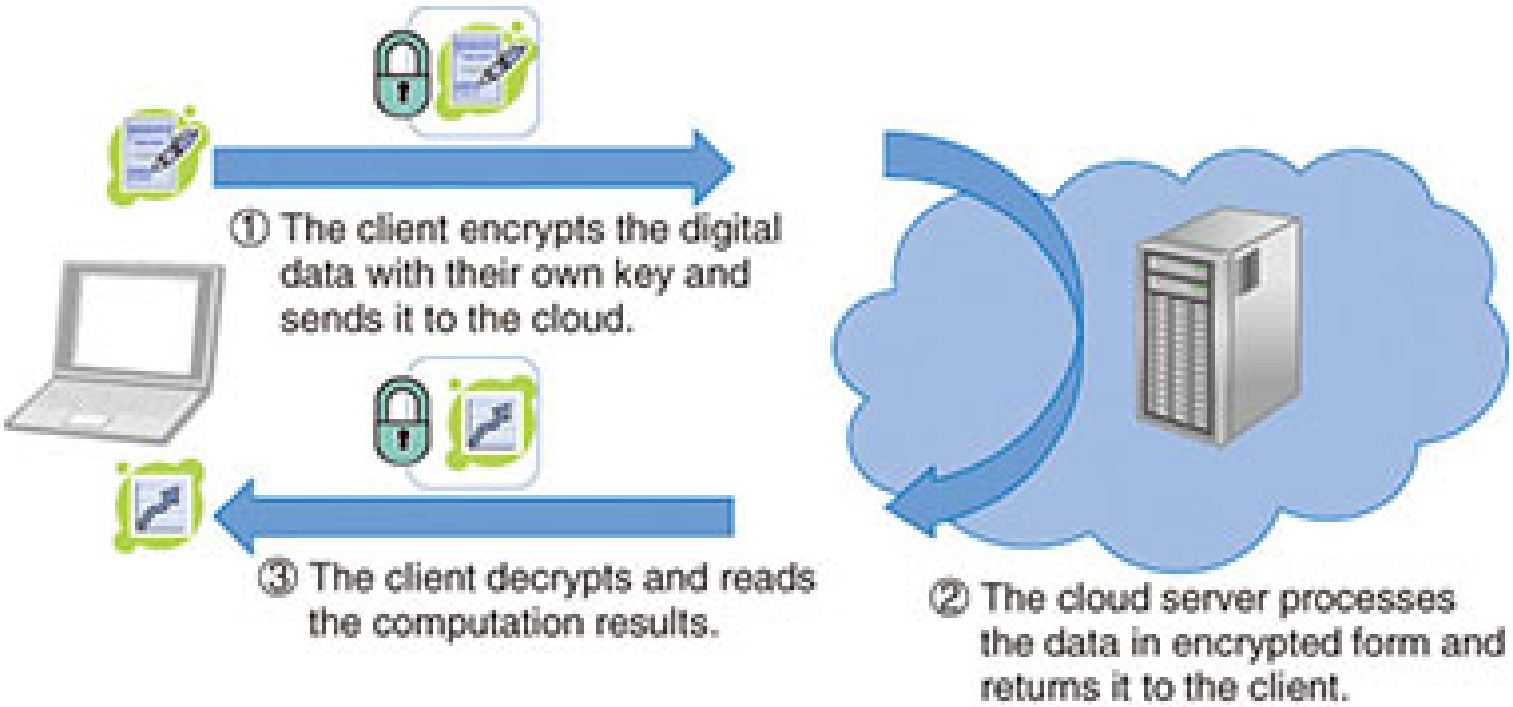
Clients have to trust in the cloud...

UNICAMP

Computing over Encrypted Data



Computing over Encrypted Data



Fully Homomorphic Encryption schemes are very inefficient...

UNICAMP

Some solutions

- Binary classifiers: Linear Means ans Fisher's Linear Discriminant [GLN12]
- Naïve Bayes and decision trees (multi-party scenario, testing phase) [BPTG14]
- Neural networks (testing phase) [DBLN16]

Principal Component Analysis

What is that?

- Project a dataset into a vector space with smaller dimension maintaining the variability of the data [Jol02]
- Vectors in the basis of this new vector space are called PCs (Principal Components)
- Dominant eigenvectors of covariance matrix

Principal Component Analysis

Applications

- Reduce the dimensionality of the data (efficiency and accuracy)
- Principal Component Regression
- Outlier detection
- Image compression (Hotelling transform)
- Face recognition [TP91]

Principal Component Analysis

Finding the PCs

- Decompositions like SVD [Shl05]
- Iterative like NIPALS [GK86]

Principal Component Analysis

Finding the PCs

- Decompositions like SVD [Shl05]
- Iterative like NIPALS [GK86]

Several divisions, comparisons and even sorting the eigenvectors (homomorphic sorting is very expensive!).

Principal Component Analysis

Finding the PCs

- Decompositions like SVD [Shl05]
- Iterative like NIPALS [GK86]

Several divisions, comparisons and even sorting the eigenvectors (homomorphic sorting is very expensive!).

We need an algorithm without divisions and that finds PCs in the right order.

YASHE cryptosystem

Things to remember

- Plaintext space: $R_t = \mathbb{Z}_t[x]/\langle \Phi_d(x) \rangle$
- Ciphertext space: $R_q = \mathbb{Z}_q[x]/\langle \Phi_d(x) \rangle$
- Implicit parameter L: multiplicative depth
- Big values for t and L imply in big values of d and q (poor efficiency)
- Cheap multiplications between ciphertexts and plaintexts available

YASHE cryptosystem

HE. paramsGen(λ) Given the security parameter λ, choose integers w, t, and q such that $w > 1$, $1 < t < q$ and q is prime. Chose a positive integer d and calculate $\Phi_d(x)$. Fix the two distributions χ_{key} and χ_{err}. Output $\mathfrak{P} = (d, q, t, \chi_{\text{key}}, \chi_{\text{err}}, w)$.

YASHE cryptosystem

HE. paramsGen(λ) Given the security parameter λ, choose integers w, t, and q such that $w > 1$, $1 < t < q$ and q is prime. Chose a positive integer d and calculate $\Phi_d(x)$. Fix the two distributions χ_{key} and χ_{err}. Output $\mathfrak{P} = (d, q, t, \chi_{\text{key}}, \chi_{\text{err}}, w)$.

HE. keyGen($\mathfrak{P}$) Sample f' from χ_{key} and let $f = tf' + 1 \in R_q$. If f is not invertible in R_q, pick a new f' and recalculate f. Compute the inverse f^{-1} of f in R_q and set $h = tgf^{-1} \in R_q$. Sample the two vectors e and s from ${\chi_{\text{err}}}^{\ell_{w,q}}$ and compute $\gamma = \text{powersOf}(f) + e + hs \in R_q^{\ell_{w,q}}$. Output $(\text{pk}, \text{sk}, \text{evk}) = (h, f, \gamma)$.

YASHE cryptosystem

HE. paramsGen(λ) Given the security parameter λ, choose integers w, t, and q such that $w > 1$, $1 < t < q$ and q is prime. Chose a positive integer d and calculate $\Phi_d(x)$. Fix the two distributions χ_{key} and χ_{err}. Output $\mathfrak{P} = (d, q, t, \chi_{\text{key}}, \chi_{\text{err}}, w)$.

HE. keyGen($\mathfrak{P}$) Sample f' from χ_{key} and let $f = tf' + 1 \in R_q$. If f is not invertible in R_q, pick a new f' and recalculate f. Compute the inverse f^{-1} of f in R_q and set $h = tgf^{-1} \in R_q$. Sample the two vectors e and s from ${\chi_{\text{err}}}^{\ell_{w,q}}$ and compute $\gamma = \text{powersOf}(f) + e + hs \in R_q^{\ell_{w,q}}$. Output $(\text{pk}, \text{sk}, \text{evk}) = (h, f, \gamma)$.

HE. enc(pk, m) Take a plaintext $m \in R_t$, sample s and e from χ_{err} and output the ciphertext $c = \left\lfloor \frac{q}{t} \right\rfloor m + e + \text{pk}\, s \in R_q$.

YASHE cryptosystem

HE. paramsGen(λ) Given the security parameter λ, choose integers w, t, and q such that $w > 1$, $1 < t < q$ and q is prime. Chose a positive integer d and calculate $\Phi_d(x)$. Fix the two distributions χ_{key} and χ_{err}. Output $\mathfrak{P} = (d, q, t, \chi_{\text{key}}, \chi_{\text{err}}, w)$.

HE. keyGen($\mathfrak{P}$) Sample f' from χ_{key} and let $f = tf' + 1 \in R_q$. If f is not invertible in R_q, pick a new f' and recalculate f. Compute the inverse f^{-1} of f in R_q and set $h = tgf^{-1} \in R_q$. Sample the two vectors e and s from ${\chi_{\text{err}}}^{\ell_{w,q}}$ and compute $\gamma = \text{powersOf}(f) + e + hs \in R_q^{\ell_{w,q}}$. Output $(\text{pk}, \text{sk}, \text{evk}) = (h, f, \gamma)$.

HE. enc(pk, m) Take a plaintext $m \in R_t$, sample s and e from χ_{err} and output the ciphertext $c = \left\lfloor \frac{q}{t} \right\rfloor m + e + \text{pk}\, s \in R_q$.

HE. dec(sk, c) Compute $z = \text{sk} \cdot c \in R_q$ and output $m = \left\lfloor \frac{t}{q} \cdot z \right\rceil \in R_t$.

YASHE cryptosystem

HE. paramsGen(λ) Given the security parameter λ, choose integers w, t, and q such that $w > 1$, $1 < t < q$ and q is prime. Chose a positive integer d and calculate $\Phi_d(x)$. Fix the two distributions χ_{key} and χ_{err}. Output $\mathfrak{P} = (d, q, t, \chi_{\text{key}}, \chi_{\text{err}}, w)$.

HE. keyGen($\mathfrak{P}$) Sample f' from χ_{key} and let $f = tf' + 1 \in R_q$. If f is not invertible in R_q, pick a new f' and recalculate f. Compute the inverse f^{-1} of f in R_q and set $h = tgf^{-1} \in R_q$. Sample the two vectors e and s from ${\chi_{\text{err}}}^{\ell_{w,q}}$ and compute $\gamma = \text{powersOf}(f) + e + hs \in R_q^{\ell_{w,q}}$. Output $(\text{pk}, \text{sk}, \text{evk}) = (h, f, \gamma)$.

HE. enc(pk, m) Take a plaintext $m \in R_t$, sample s and e from χ_{err} and output the ciphertext $c = \left\lfloor \frac{q}{t} \right\rfloor m + e + \text{pk}\, s \in R_q$.

HE. dec(sk, c) Compute $z = \text{sk} \cdot c \in R_q$ and output $m = \left\lfloor \frac{t}{q} \cdot z \right\rceil \in R_t$.

HE. add(c_1, c_2) Output $c_{\text{add}} = c_1 + c_2 \in R_q$.

UNICAMP

YASHE cryptosystem

HE. paramsGen(λ) Given the security parameter λ, choose integers w, t, and q such that $w > 1$, $1 < t < q$ and q is prime. Chose a positive integer d and calculate $\Phi_d(x)$. Fix the two distributions χ_{key} and χ_{err}. Output $\mathfrak{P} = (d, q, t, \chi_{\text{key}}, \chi_{\text{err}}, w)$.

HE. keyGen($\mathfrak{P}$) Sample f' from χ_{key} and let $f = tf' + 1 \in R_q$. If f is not invertible in R_q, pick a new f' and recalculate f. Compute the inverse f^{-1} of f in R_q and set $h = tgf^{-1} \in R_q$. Sample the two vectors e and s from ${\chi_{\text{err}}}^{\ell_{w,q}}$ and compute $\gamma = \text{powersOf}(f) + e + hs \in R_q^{\ell_{w,q}}$. Output $(\text{pk}, \text{sk}, \text{evk}) = (h, f, \gamma)$.

HE. enc(pk, m) Take a plaintext $m \in R_t$, sample s and e from χ_{err} and output the ciphertext $c = \left\lfloor \frac{q}{t} \right\rfloor m + e + \text{pk}\, s \in R_q$.

HE. dec(sk, c) Compute $z = \text{sk} \cdot c \in R_q$ and output $m = \left\lfloor \frac{t}{q} \cdot z \right\rceil \in R_t$.

HE. add(c_1, c_2) Output $c_{\text{add}} = c_1 + c_2 \in R_q$.

HE. prod(c_1, c_2) Compute $c_3 = \frac{t}{q} c_1 c_2 \in R_q$ and output $c_{\text{mult}} = \text{keySwitch}(c_3, \text{evk})$.

YASHE cryptosystem

Things to remember

- Plaintext space: $R_t = \mathbb{Z}_t[x]/\langle \Phi_d(x) \rangle$
- Ciphertext space: $R_q = \mathbb{Z}_q[x]/\langle \Phi_d(x) \rangle$
- Implicit parameter L: multiplicative depth
- Big values for t and L imply in big values of d and q (poor efficiency)
- Cheap multiplications between ciphertexts and plaintexts available

Encoding double values allows us to divide ciphertexts by plaintexts.

To ensure correctness of decoding, we have to chose big values of t

Our strategy

How do we evaluate the PCA homomorphicaly?

- Calculate the covariance matrix
- Find the dominant eigenvector (first PC)
- Shift the eigenvectors of covariance matrix
- Find the dominant eigenvector (second PC)
- Shift the eigenvectors
- Find the dominant eigenvector (third PC)
- ...

Covariance

Algorithm 1 COVARIANCE

Input: $X \in \mathbb{R}^{n \times p}$, dominant eigenvector v

for $j = 1$ **to** p **do**
    $\mu = 0$
    for $i = 1$ **to** p **do**
        $\mu = \mu + X[i][j]$
    end for
    $\mu = \mu / n$
    for $i = 1$ **to** p **do**
        $X[i][j] = X[i][j] - \mu$
    end for
end for
$C = X^T * X$ ▷ Multiplication between ciphertexts.
$C = C/(n-1)$
return C

First Principal Component

Algorithm 2 POWER METHOD

Input: $C \in \mathbb{R}^{p \times p}$

$u =$ random p-dimensional vector
for $k = 1$ **to** $\lceil \log_2(S) \rceil$ **do**
    $C = C * C$ ▷ multiplication between ciphertexts
    $u = u/\theta_k$
end for
return $C * u$

Multiplicative depth: $\lceil \log_2(S) \rceil$

Further PCs

Shifting the eigenvectors

- The second dominant eigenvector becomes the first one
- The third dominant eigenvector becomes the second one
- So on...

If we had a way to shift the eigenvectors, we could iterate applying the Power Method and the Shift procedure.

UNICAMP

Further PCs

We can use the Spectral Theorem to write a symmetric matrix C as sum involving its eigenvalues and the its normalized eigenvectors:

$$C = \sum_{i=1}^{p} \lambda_i v_i v_i^T$$

So, the following algorithm works:

Algorithm 3 (PLAINTEXT) SHIFT

Input: $C \in \mathbb{R}^{p \times p}$, dominant eigenvector v

$u = \frac{v}{||v||}$
$B = C - Cuu^T$
return B

Further PCs

Normalizing the eigenvector would require divisions...

Further PCs

Normalizing the eigenvector would require divisions...

But multiplying a matrix by a positive constant does not change its eigenvectors

Further PCs

Normalizing the eigenvector would require divisions...

But multiplying a matrix by a positive constant does not change its eigenvectors

and $||v||^2$ is the inner product $v^T v$, which can be computed using only products and additions...

Further PCs

Instead of returning

$$B = C - C\frac{v}{||v||}\frac{v^T}{||v||}$$

we return

$$||v||^2 B = ||v||^2(C - C\frac{v}{||v||}\frac{v^T}{||v||}) = ||v||^2 C - Cvv^T$$

Algorithm 4 (HOMOMORPHIC) SHIFT

Input: $C \in \mathbb{R}^{p\times p}$, $v \in \mathbb{R}^p$

$\alpha =$ INNER PRODUCT (v, v)
$B' = \alpha C - Cvv^T$
return B'

Homomorphic PCA

Algorithm 5 PCA

Input: $X \in \mathbb{R}^{n \times p}$, new dimension K

$C = \text{COVARIANCE}(X)$
$pcs = \emptyset$
for $i = 1$ **to** K **do**
    $pc_i = \text{POWER METHOD}(C)$
    $C = \text{HOMOMORPHIC SHIFT}(C, pc_i)$
    $pcs = \{pc_i\} \cup pcs$
end for
return pcs

Multiplicative depth: $2K - 1 + K \cdot \lceil \log_2(S) \rceil$ (linear in K).

Interactive Homomorphic PCA

Algorithm 6 Interactive PCA

Input: $X \in \mathbb{R}^{n \times p}$, new dimension K

$C = \text{covariance}(X)$
$pcs = \emptyset$
for $i = 1$ **to** K **do**
    $pc_i = \text{POWER METHOD}(C)$
    $pc_i = \text{REENCRYPT}(pc_i)$
    $C = \text{HOMOMORPHIC SHIFT}(C, pc_i)$
    $pcs = \{pc_i\} \cup pcs$
end for
return pcs

Multiplicative depth: $2(K-1) + 1 + \lceil log_2(S) \rceil$.

We can use smaller values for t.

Experimental Results

Angle between the expected PC and the PC found.

L_∞: infinity norm of the difference between the expected PC and the PC found.

Table: Accuracies for the first and the second PC.

	ANGLE	L_∞
FIRST PC	0.0029	0.0012633
SECOND PC	0.0076	0.0066135

Experimental Results

Table: Execution times for data sets with *N* samples of *P* variables.

Number of samples	P = 5	P = 10	P = 15
15	4h29min	5h50min	8h33min
25	4h55min	6h25min	8h52min
50	5h42min	6h58min	9h54min

Conclusions

- Proposed solution has multiplicative depth linear in the number of PCs.
- PCs are found in the right order (no need for sorting).
- The accuracy is good.
- Execution times are still prohibitive.
- We need more efficient encoding techniques.

Future Works

- Search for new encoding techniques.
- Use other homomorphic encryption schemes.
- Use CRT to split the big value of t into several small values.

Bibliography

BPTG14 R. Bost, R. A. Popa, S. Tu, and S. Goldwasser, **Machine learning classification over encrypted data**, Network and Distributed System Security Symposium - NDSS 2015.

DBLN16 N. Dowlin, R. Bachrach, K. Laine, K. Lauter, and M. Naehrig, **CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy**, Cryptology ePrint Archive, 2016.

GK86 P. Geladi and B. R. Kowalski, **Partial Least Squares Regression: A Tutorial**, Analytica Chimica Acta, 1986.

GLN12 T. Graepel, K. Lauter, and M. Naehrig, **ML Confidential: Machine Learning on Encrypted Data**, International Conference on Information Security and Cryptology – ICISC 2012.

Jol02 I. T. Jolliffe, **Principal Component Analysis**, Springer Series in Statistics, 2002.

LN14 T. Lepoint, and M. Naehrig, **A comparison of the homomorphic encryption schemes FV and YASHE**, AFRICACRYPT 2014.

Shl05 Jonathon Shlens, **A tutorial on Principal Component Analysis**, Systems Neurobiology Laboratory, 2005.

Tib14 M. Tibouchi, **Fully Homomorphic Encryption over the Integers: From Theory to Practice**, NTT Technical Review, 2014.

TP91 M. Turk and A. Pentland, **Eigenfaces for Recognition**, Journal of Cognitive Neuroscience, 1991.

UNICAMP

Questions

UNICAMP