Improved Privacy-Preserving Training using fixed-Hessian Minimisation

Tabitha Ogilvie, Rachel Player, Joe Rowell

Information Security Group

WAHC 2020
Contributions

➢ A new method for homomorphic Logistic Regression training using CKKS

➢ A comparison of 3 different Logistic Regression training methods at 128-bit security

➢ An improved method for homomorphically calculating $\frac{1}{x}$

➢ Three methods for homomorphic Ridge Regression training
data

model
encrypted data

encrypted model
Homomorphic Encryption

A scheme $E$ is fully homomorphic if, given a function $f$ and an encryption of some data $E(x)$, we can generate $E(f(x))$ without decrypting.
Logistic Regression
➢ A binary classifier based on the sigmoid function
➢ No closed form solution
➢ Cost function given by

\[ J(\beta) = - \sum_{i=1}^{n} \log(1 + \exp(-y_i \beta^T x_i)) \]

Ridge Regression
➢ Linear regression with L2 regularisation
➢ Closed form solution
➢ Cost function given by

\[ J(\beta) = \frac{1}{2} \left( \lambda \sum_{i=1}^{d} \beta_i^2 + \sum_{i=1}^{n} (y_i - \beta^T x_i)^2 \right) \]
Prior Work
Main Comparisons

Kim18a achieves Logistic Regression training using:

- “Feature wise” encoding
- 80-bit security
- Gradient Descent

\[
\beta^{(k+1)} = \beta^{(k)} - \alpha \nabla J(\beta^{(k)})
\]

Kim18b achieves Logistic Regression training using:

- “database” encoding
- 80-bit security
- Nesterov’s Accelerated Gradient Descent

\[
\begin{align*}
\beta^{(k+1)} &= \nu^{(k)} - \alpha_k \cdot \nabla J(\nu^{(k)}) \\
\nu^{(k+1)} &= (1 - \gamma_k) \cdot \beta^{(k+1)} + \gamma_k \cdot \beta^{(k)}
\end{align*}
\]


Hessian-Based Minimisation

\[ \beta^{(k+1)} = \beta^{(k)} - H(\beta^{(k)})^{-1} \nabla J(\beta^{(k)}) \]

➢ \( H(\beta^{(k)}) \) depends on the current value of the parameters
➢ We need to invert a matrix

Böhning and Lindsay: replace the full Hessian with a \textit{fixed} symmetric positive definite matrix \( \tilde{H} \) such that \( \tilde{H} \geq H \):

➢ Monotonic
➢ Guaranteed convergence to a minimum (if \( J \) is bounded below)
➢ Linear rate of convergence

Fixed Hessian Logistic Regression Training

\[ H(\beta^{(k)}) \] \[ \frac{1}{4} X^T X \] \[ H \]

BL88 \quad BV18: row sums


Update the BV fixed-Hessian approach, adding the following features:

- Use the CKKS encoding to **parallelise** the calculation of the entries of $\bar{H}$, consuming only 1 level
- Use a data pre-processing step similar to 18a to reduce the number of levels per parameter update to 3
- Replace the linear Taylor approximation with a linear Chebyshev approximation
- To invert the entries $\frac{1}{H_{kk}}$, we first use the linear approximation from Schulte et al., and then iterate Newton Raphson 3 times, reducing the relative error

## Implementation & Comparison

<table>
<thead>
<tr>
<th>Descent</th>
<th>Enc</th>
<th>log $\Delta$</th>
<th>deg $g$</th>
<th>$\mu$</th>
<th>Time</th>
<th>AUC</th>
<th>Acc. (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GD</td>
<td>F</td>
<td>27</td>
<td>3</td>
<td>8</td>
<td>102.18s</td>
<td>0.89</td>
<td>82.53</td>
</tr>
<tr>
<td></td>
<td>F</td>
<td>28</td>
<td>3</td>
<td>7</td>
<td>59.12s</td>
<td>0.93</td>
<td>83.65</td>
</tr>
<tr>
<td></td>
<td>F</td>
<td>31</td>
<td>3</td>
<td>7</td>
<td>58.62s</td>
<td>0.95</td>
<td>80.46</td>
</tr>
<tr>
<td>NAD</td>
<td>D</td>
<td>30</td>
<td>3</td>
<td>5</td>
<td>42.51s</td>
<td>0.96</td>
<td>88.90</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>31</td>
<td>3</td>
<td>5</td>
<td>42.57s</td>
<td>0.94</td>
<td>88.74</td>
</tr>
<tr>
<td>FH (Ours)</td>
<td>F</td>
<td>39</td>
<td>1</td>
<td>5</td>
<td>45.70s</td>
<td>0.94</td>
<td>88.50</td>
</tr>
<tr>
<td></td>
<td>F</td>
<td>40</td>
<td>1</td>
<td>4</td>
<td>27.18s</td>
<td>0.92</td>
<td>88.26</td>
</tr>
<tr>
<td></td>
<td>F</td>
<td>45</td>
<td>1</td>
<td>4</td>
<td>30.98s</td>
<td>0.92</td>
<td>88.26</td>
</tr>
</tbody>
</table>
Why Use a Fixed-Hessian Method?

➢ With the Fixed-Hessian method, no step size needs to be chosen
➢ Adopting “real world” methods for choosing the step size could compromise security

\[
\beta^{(k+1)} = \beta^{(k)} - \alpha \nabla J(\beta^{(k)})
\]

\[
\begin{align*}
\beta^{(k+1)} &= v^{(k)} - \alpha_k \cdot \nabla J(v^{(k)}) \\
\nu^{(k+1)} &= (1 - \gamma_k) \cdot \beta^{(k+1)} + \gamma_k \cdot \beta^{(k)}
\end{align*}
\]
Ridge Regression

- Use the same three minimisation techniques
- Use the BV diagonalization method to approximate the Hessian
- Use a feature-by-feature encoding to parallelise over the size of the dataset
- Use a server-side pre-processing step to reduce computation per iteration
## Implementation & Comparison

<table>
<thead>
<tr>
<th>Descent</th>
<th>log $\Delta$</th>
<th>$\mu$</th>
<th>Time</th>
<th>$r^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>GD</td>
<td>40</td>
<td>9</td>
<td>207.27s</td>
<td>0.4165</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>13</td>
<td>450.53s</td>
<td>0.4058</td>
</tr>
<tr>
<td>NAD</td>
<td>40</td>
<td>6</td>
<td>127.71s</td>
<td>0.4054</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>9</td>
<td>415.54s</td>
<td>0.4566</td>
</tr>
<tr>
<td>FH</td>
<td>40</td>
<td>6</td>
<td>137.68s</td>
<td>0.3206</td>
</tr>
</tbody>
</table>
Further Work

➢ Ridge Regression closed form solution

➢ CKKS precision
   ➢ Setting the precision factor
   ➢ Precision loss and iterative descent methods
   ➢ Bootstrapping
Thank You!