On the Use of a Machine Learning (ML) Model to Improve the Time Complexity of Finding the Inverse of a Square Matrix With Gauss-Jordan Elimination

Anay Aggarwal

Stoller MS

There are multiple ways to calculate the inverse of a square matrix. The first is simply guessing and checking matrices, which is very inefficient. The second involves cofactors, adjugates, and determinants. This is also quite inefficient, but it can be done systemically. By hand, the method that is regarded as the most efficient is the Gauss-Jordan elimination method. However, the Gauss-Jordan method is difficult to implement in code because it requires intuition. Luckily, we have Machine Learning for that. Inverses of matrices can be used to solve systems of equations. Matrix inversion also plays an important role in Computer Graphics and cryptography.

Guessing and Checking, while being a valid way to invert a matrix, is extremely inefficient. In fact, it is impossible to measure it's time complexity without using other methods. You may ask how to do the "checking" part. The inverse of a matrix $\mathbb{A}$ is defined as the matrix $\mathbb{A}^{-1}$ such that the following holds:

$$\mathbb{A} \cdot \mathbb{A}^{-1} = I,$$

where I is the identity matrix (Weisstein & Stover, n.d.). The $n \times n$ identity matrix is defined as the following:

$$I_{i,j} = 0 \text{ if } i \neq j$$

$$I_{i,j} = 1 \text{ if } i = j$$

For example, $I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$.

Let's attempt to find the time complexity given a set S such that $\mathbb{A}_{i,j}^{-1} \in S \forall \mathbb{A}_{i,j}^{-1} \in \mathbb{A}^{-1}$. Without Loss of Generality, suppose that $\mathbb{A}$ is an $n \times n$ matrix. Notice that we will loop through each of the elements in $\mathbb{A}^{-1}$ (n^2 elements) $|S|$ times ($|S|$ denotes the cardinality, number of elements, of the set S). Thus the complexity for the guessing is $O(n^2|S|)$. To check, we must multiply the two matrices. As of October 2020, the matrix multiplication algorithm with the best time complexity runs in $O(n^{2.3728596})$ time (multiplying two $n \times n$ matrices), according to Alman & Williams, 2020. Multiplying the complexities, we get a final runtime of $O(n^{4.3728596}|S|)$. Notice that often, $S \not\subseteq \mathbb{Z}^+$ or even $S \not\subseteq \mathbb{Z}$ altogether, so this is not very feasible.

The second method to invert a matrix involves a few new concepts. It utilizes the fact that

$$\mathbb{A}^{-1} = \frac{1}{\det(\mathbb{A})} \text{adj}(\mathbb{A}),$$

where $\det(\mathbb{A})$ and $\text{adj}(\mathbb{A})$ are the determinant and adjugate of $\mathbb{A}$, respectively. To find the determinant, we use minors. For each $\mathbb{A}_{i,j}$, we can systematically pop out all $\mathbb{A}_{i,k}$ and $\mathbb{A}_{j,k}$ for $1 \leq k \leq n$ ($\mathbb{A}$ is an $n \times n$ matrix). We then calculate the determinant of the remaining matrix, and assign that to $\mathbb{A}_{i,j}$, according to Math Is Fun, n.d. In more formal terms,

$$\mathbb{B}_{i,j} = \det(\mathbb{A}_{x,y}), x \neq i, y \neq j,$$

where $\mathbb{B}$ is now the matrix of minors. To convert this to the matrix of cofactors ($\mathbb{C}$), for each i, j with $i + j = 0 \pmod 2$, we assign $\mathbb{C}_{i,j}$ to $\mathbb{B}_{i,j}$, and for each i, j with $i + j = 1 \pmod 2$, we assign $\mathbb{C}_{i,j}$ to $-\mathbb{B}_{i,j}$ ($i, j \in \mathbb{Z}$, of course). Alternatively, we could simply multiply each element in $\mathbb{B}_{i,j}$ by $(-1)^{i+j}$ to get $\mathbb{C}$, according to Rusczyk & Lehoczky, 2017. "Now 'Transpose' all elements of the previous matrix... in other words swap their positions over the diagonal

(the diagonal stays the same)" (Math is Fun, n.d.). The result is the adjugate matrix. Now choose $q \in \{1,2,3,...,n\}$ (it doesn't matter which q, you choose, the result will be the same). We have

$$\det(\mathbb{A}) = \sum_{r=1}^{n} \left(\text{adj}(\mathbb{A})_{q,r} \cdot \mathbb{A}_{q,r}\right).$$

The determinant of $\mathbb{A}$ can also be written as $\underline{\mathbb{A}}$, as per Rusczyk & Lehoczky, 2017. It is important to note that $\mathbb{A}^{-1}$ does not exist if $\underline{\mathbb{A}} = 0$. We can see that the runtime in this method is at least bounded, be it still large.

The method that we will be focusing on in this project is commonly known as Gauss-Jordan elimination. It is the most common way to invert a matrix by hand. "To apply Gauss-Jordan elimination, operate on a matrix

$$[\mathbb{A}\ \mathbb{I}] = \begin{bmatrix} a_{11} & ... & a_{1n} \\ a_{21} & ... & a_{2n} \\ \vdots & \ddots & \vdots \\ a_{n1} & ... & a_{nn} \end{bmatrix} \begin{bmatrix} 1 & 0 & ... & 0 \\ 0 & 1 & ... & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & ... & 1 \end{bmatrix},$$

where $\mathbb{I}$ is the identity matrix, and use gaussian elimination to obtain a matrix of the form

$$\begin{bmatrix} 1 & 0 & ... & 0 \\ 0 & 1 & ... & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & ... & 1 \end{bmatrix} \begin{bmatrix} b_{11} & ... & b_{1n} \\ b_{21} & ... & b_{2n} \\ \vdots & \ddots & \vdots \\ b_{n1} & ... & b_{nn} \end{bmatrix}.$$

The matrix

$$\mathbb{B} = \begin{bmatrix} b_{11} & ... & b_{1n} \\ b_{21} & ... & b_{2n} \\ \vdots & \ddots & \vdots \\ b_{n1} & ... & b_{nn} \end{bmatrix}$$

is then the matrix inverse of $\mathbb{A}$. " (Weisstein, Gauss-Jordan Elimination, n.d.). But what exactly is Gaussian elimination? It is a method of solving matrix equations of the form $\mathbf{A}x = \mathbf{b}$, according to Weisstein, Gaussian Elimination, n.d. It performs a series of row operations on a matrix. For example, given

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

we can perform an operation such as $R2 \cdot 3 + R1 \cdot 2 \to R1$. First off, notice that $R1, R2, R3$ are the first, second, and third rows of the matrix, respectively. What this does is multiply the vector $\vec{R2} = \begin{bmatrix} 4 & 5 & 6 \end{bmatrix}$, second row, by the scalar 3. It then multiplies the vector $\vec{R1} = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$ by the scalar 2. It adds the two results, and pops the result into the first row of the matrix. This is $\begin{bmatrix} 12 & 15 & 18 \end{bmatrix} + \begin{bmatrix} 2 & 4 & 6 \end{bmatrix} = \begin{bmatrix} 14 & 19 & 24 \end{bmatrix}$. Thus the new matrix is $\begin{bmatrix} 14 & 19 & 24 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$.

We must do the same to the second matrix. This yields $\begin{bmatrix} 2 & 3 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$. So we went from

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

to

$$\begin{bmatrix} 14 & 19 & 24 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \begin{bmatrix} 2 & 3 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

in one "move". Our end goal is the identity matrix on the left hand side, so this "move" was pretty useless. We can do any move of the form $\mathbf{v}_1 \cdot c_1 + \mathbf{v}_2 \cdot c_2 \to \mathbf{v}_3$, for constant c_1, c_2 (need not be in $\mathbb{Z}^+$), and vectors $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ all in matrix $\mathbb{A}$.

Building a model for this process will be difficult, since it Gauss-Jordan elimination is not systematic. First, a program that will perform row operations (e.g. $R1 \cdot 1 + R3 \cdot 4 \to R3$) will be written. Then, systematic code for Gauss-Jordan elimination will be added in order to generate a dataset of the form

$$\mathbb{A}, m_1, m_2, m_3, ..., m_k,$$

where $\mathbb{A}$ is a matrix and $m_1, m_2, ..., m_k$ are the "moves" applied to find the inverse of $\mathbb{A}$. We will include over 10,000 matrices. The native language will be C++. According to Chintamaneni, 2015, given that $\mathbb{A}$ is $n \times n$, $\max(k) = \frac{n(n+1)}{2} + \frac{2n^3+3n^2-5n}{3} = \frac{4n^3+9n^2-7n}{6}$, so this is possible. Finally, we will train a ML model with the generated dataset, and ideally, it will learn how to make the set of flawless moves to find the invert any $\mathbb{A}$. The result should have a faster runtime than the plain Gauss-Jordan model does ($O(n^3)$).

Finding the inverse of a matrix has many applications. It's biggest use is in 3D graphics, specifically rotations. According to Wikipedia, 2021, "The inverse of a rotation matrix is its transpose, which is also a rotation matrix". The matrix that rotates a point (x, y) with angle θ about the $x-$axis with respect to the origin is $\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$, making the altered points $(x', y') = (x\cos\theta - y\sin\theta, x\sin\theta + y\cos\theta)$. Inverting this, we get another type of rotation matrix.

Matrices are also useful in cryptography. Matrices can be used to encrypt a message, meaning that we need the inverse of the matrix to decrypt the message. The quicker we find the inverse, the shorter it will take to crack. First, we split into groups of two letters (i.e. MATH RULES $\to$ MA, TH, -R, UL, ES). Then, use $A = 1, B = 2, C = 3, D = 4, ..., Z = 26, - = 27$, to convert each pair into a 2×1 matrix (MA is $\begin{bmatrix} 13 \\ 1 \end{bmatrix}$). Finally, we multiply each of the resulting matrices by the key matrix $\mathbb{A}$. This can be anything you want, as long as it's 2×2, according to Sekhon, et.al, 2021. If $\mathbb{A}$ is $n \times n$, then we would partition MATH RULES into sets of size n. To decode a message, the steps are: "

1. Take the string of coded numbers and multiply it by the inverse of the matrix that was used to encode the message.

2. Associate the numbers with their corresponding letters." (Sekhon, et.al, 2021).

Also, according to Sekchon, et.al, 2021, "This method, known as the Hill Algorithm, was created by Lester Hill, a mathematics professor who taught at several US colleges and also was involved with military encryption. "

In conclusion, inverting a matrix fast is important, because it is essential to 3D graphics, and useful in the field of cryptography. We can invert a matrix in multiply ways, including guessing and checking, using the determinant-adjugate formula, and Gauss-Jordan elimination. Gauss-Jordan elimination is the most efficient, but it cannot be easily done systematically (without losing some runtime), because it requires intuition. The goal of this project is to create a machine learning model to speed up the time complexity of Gauss-Jordan elimination.

References

1. Weisstein, E. W., & Stover, C. (n.d.). Matrix Inverse.

   Retrieved January 05, 2021, from https://mathworld.wolfram.com/MatrixInverse.html

2. Alman, J., & Williams, V. V. (2020, October 13). A Refined Laser Method and Faster Matrix Multiplication.

   Retrieved January 05, 2021, from https://arxiv.org/pdf/2010.05846.pdf

3. Inverse of a Matrix using Minors, Cofactors and Adjugate. (n.d.). Retrieved January 05, 2021, from

   https://www.mathsisfun.com/algebra/matrix-inverse-minors-cofactors-adjugate.html

4. Rusczyk, R., & Lehoczky, S. (2017). The Art of Problem Solving: Volume 2 and Beyond. Alpine, CA: AoPS.

5. Weisstein, E. W. (n.d.). Gauss-Jordan Elimination. Retrieved January 05, 2021, from

   https://mathworld.wolfram.com/Gauss-JordanElimination.html

6. Weisstein, E. W. (n.d.). Gaussian Elimination. Retrieved January 06, 2021, from

   https://mathworld.wolfram.com/GaussianElimination.html

7. Chintamaneni, K. (2015, September 23). Operations Required in Matrix Elimination. Retrieved January 06,

   2021, from http://web.mit.edu/18.06/www/Fall15/Matrices.pdf

8. Rotation matrix. (2021, January 01). Retrieved January 06, 2021, from https://en.wikipedia.org/wiki/Rotation_matrix

9. Sekhon, R., & Bloom, R. (2021, January 02). 2.5: Application of Matrices in Cryptography. Retrieved January

   07, 2021, from

   https://math.libretexts.org/Bookshelves/Applied_Mathematics/Book%3A_Applied_Finite_Mathematics

   _(Sekhon_and_Bloom)/02%3A_Matrices/2.05%3A_Application_of_Matrices_in_Cryptography