Measuring the Predictability of the Shortest Vector Problem Using Linear Regression

2021 Summer STEM Institute

Isha Agarwal

Abstract

The creation of a quantum computer would threaten the security of currently used cryptographic protocols, causing an influx of interest in cryptosystems that are quantum resistant. Lattice-based cryptography is a promising field in this area, but the field has a large gap between practical and theoretical knowledge. In this paper, we propose to bridge this gap by studying specific instances of the shortest vector problem. We implement a machine learning linear regression model to estimate the solutions to particular lattice problems and use its performance as a measure of how difficult these problems are. Performing such an evaluation is a necessary step in implementing lattice-based encryption schemes and ultimately ensuring privacy and security persist in a modern world. In most cases, our model is able to account for over 90% *of the variability in the solutions to lattice problems for dimensions up to* 50*. In particular, our model had an* R^2 *value of* 0.996 *for NTRU lattices which used in the NTRU cryptosystem. We also found that traits such as randomness of numbers in the basis and homogeneity of matrix structure distinguish lattices that are predictable and unpredictable. Our work is crucial in defining a metric to determine which lattices are well-suited for cryptographic protocols, and extending our work to train a model on more features and higher dimensions could more accurately and holistically accomplish this task.*

1. Introduction

Creating secure cryptographic protocols is crucial in a society where data is ubiquitous. Furthermore, as the creation of a quantum computer continues to become more and more realistic, the need for secure encryption schemes becomes even more urgent. In a post-quantum world, or a world in which quantum computers are functional, the most widely-used encryption schemes would become easily decryptable (Shor, 1994). As a result, all digital transactions from emails to bank accounts would be exposed. While the creation of a quantum computer is currently far from reality, increasing understanding of quantum mechanics and improving technology are bringing humanity closer to the possibility, suggesting that one ought to be prepared.

We define a *lattice* to be a regularly shaped grid. The field of *lattice-based cryptography* deals with creating cryptosystems that are based on difficult problems involving lattices. Lattice problems such as finding the shortest vector in a lattice are NP-hard (Ajtai, 1998), meaning they are widely-believed to be impossible to solve in polynomial time, even on a quantum computer. In addition, lattice problems have a high average-case complexity (Ajtai, 1996). This means that any solution to a random instance of the problem would work for the worst-case problem as well, meaning the problem is approximately equally difficult across all cases. Hence, given the potential for highly secure quantum resistant lattice-based cryptographic protocols, gaining a better understanding of lattice problems is crucial to maintaining notions of security and privacy in a modern world.

However, while lattice-based cryptography is a very promising field in theory, it often falls short in practice. Even though there are worst-case theoretical bounds on complexity, experimental results suggest that not all lattices are as complex as these bounds (Gama & Nguyen, 2008). Furthermore, compared to problems currently used in cryptographic protocols, such as factoring and discrete logarithms, very little is understood about the behavior of particular lattice problems (Gama & Nguyen, 2008). Without this understanding, lattice-based quantum resistant cryptographic protocols

Research project conducted during the 2021 Summer STEM Institute. The Summer STEM Institute (SSI) is a six-week virtual summer program where high school students learn how to design and conduct data science and other computational research projects. To view other SSI distinguished projects, please visit www.summersteminstitute.org.

cannot be built and incorporated into society, and everyone's data will remain vulnerable.

Several works explore the implementation of lattice problems to cryptography (Hoffstein et al., 1998; Regev, 2009; Gentry, 2009), while others such as (Ajtai, 1998; 1996) focus more on the theory behind these problems. However, theoretical works tend take on a more general approach, resulting in there not being much understanding about specific instances of lattice problems. Especially since the solutions to random lattices problems can be estimated quite accurately (Ducas, 2018), understanding which lattices yield unpredictable results is essential for the creation of secure cryptographic protocols.

We propose to address the gap between theoretical and practical knowledge of lattice-based cryptography by studying specific instances of lattice problems. In order to study this, we employ a linear regression machine learning model that attempts to predict the solutions to a particular subset of lattice problems. Based on the accuracy rate of the model, we determine how difficult specific instances of the problem that we engineer are.

We aim to use the performance of our linear machine learning model as a metric for determining the difficulty of lattice problems as measured by predictability. Using this metric to predict the solutions to various instances of lattice problems will allow us to determine the traits of lattices that would be well-suited for cryptographic protocols. Determining which lattices are secure is a crucial step forward in terms of implementing lattice-based encryption schemes and ensuring that security and privacy will continue to exist in a post-quantum world.

For most types of lattices, including the lattices used in cryptograhpic protocols, our model is able to explain over 90% of the variability of solutions to lattice problems for small dimensions. Furthermore, we identify randomness and symmetry in structure as traits that distinguish predictable and less predictable lattices.

We begin by giving the reader a brief introduction to concepts related to lattice-based cryptography that are crucial to understanding the rest of this paper in Section 2. In Section 3, we discuss what is currently known about lattice problems and related works. Given the state of the field, we state our research questions in Section 4. Next, we explain how we conducted our experiments and created our machine learning model in Section 5. In Section 6 we present our results and analyze their implications in the context of lattice-based cryptography. Finally, we conclude in Section 7 by summarizing the paper and proposing avenues for future work.

2. Preliminaries

In order to gain a deeper understanding of lattice-based cryptography, we first define some terms and notation that will be used throughout the paper.

Definition 1 (Lattice)**.** A *lattice* is a regularly spaced grid, which we denote as $\mathcal{L}$.

Definition 2 (Linear Independence)**.** We call a set of vectors $\mathbf{v_1}, \mathbf{v_2}, \cdots, \mathbf{v_n}$ *linearly independent* if for some integers $c_1, c_2, \cdots, c_n$, the only solution to the equation $c_1\mathbf{v_1} + c_2\mathbf{v_2} + \cdots + c_n\mathbf{v_n} = 0$ is $c_1 = c_2 = \cdots = c_n = 0$.

Definition 3 (Basis)**.** A *basis* is a set of vectors, $\mathbf{b_1}, \mathbf{b_2}, \cdots, \mathbf{b_n}$, that are linearly independent. We can generate a lattice by taking all possible linear combinations of the basis. We denote a lattice $\mathcal{L}$ with basis $\mathbf{B} = \mathbf{b_1}, \mathbf{b_2} \cdots \mathbf{b_n}$ as $\mathcal{L}(\mathbf{B})$. Furthermore, we can represent $\mathcal{L}(\mathbf{B})$ as the matrix:

$$\begin{pmatrix} \mathbf{b_1} \\ \mathbf{b_2} \\ \vdots \\ \mathbf{b_n} \end{pmatrix}.$$

Note that an infinite amount of bases can generate the same lattice. This concept is the foundation behind some of the algorithms to solve lattice problems.

Using these terms, we are able to define one of the most fundamental hard problems in lattice-based cryptography:

Definition 4 (Shortest Vector Problem)**.** The *shortest vector problem* (SVP) states that given a basis for a lattice, find the shortest non-zero vector in the lattice that can be generated by taking linear combinations of the basis. We denote the shortest vector of lattice $\mathcal{L}$ as $\lambda(\mathcal{L})$.

To better understand SVP, we provide an example. In the example in Figure 1, $\mathbf{B} = \{(19, -6), (31, -11)\}$ and $\lambda(\mathcal{L}) = (2, 3)$. Note that this example is only in 2 dimensions, when in practice dimensions greater than 300 are used.

Next, we introduce another lattice-based problem that reduces to SVP.

Definition 5 (Closest Vector Problem)**.** In the *closest vector problem* (CVP) a lattice $\mathcal{L}(\mathbf{B})$ and a target vector $\mathbf{v}$ are given, and the goal is to find the closest vector to $\mathbf{v}$ in $\mathcal{L}(\mathbf{B})$.

Finally, we introduce a heuristic than can estimate the length of $\lambda(\mathcal{L})$.

Definition 6 (Gaussian Heuristic)**.** The *Gaussian Heuristic* estimates the length of the shortest vector in a random lattice (Ducas, 2018). It is given by the formula:

$$\lambda(\mathcal{L}) \approx \sqrt{\frac{n}{2\pi e}} \cdot \det(\mathcal{L})^{1/n},$$

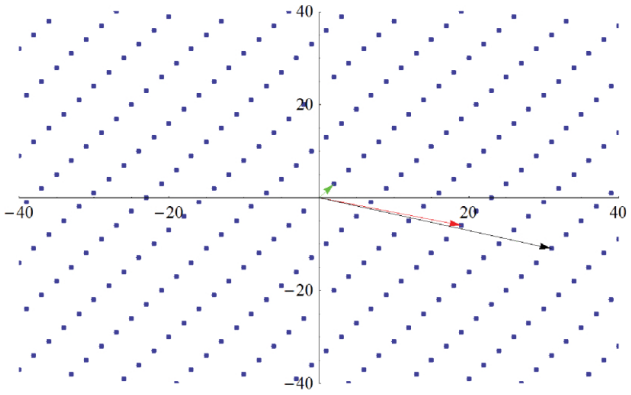


Figure 1: An example of SVP for 2 dimensions from (Chuang et al., 2018). If the basis (shown in red and grey) is $\{(19, -6), (31, -11)\}$, then the shortest vector in the lattice (shown in green) will be $5(19, -6) + 3(31, -11) = (2, 3)$.

where n is the dimension of the lattice and $\det(L)$ is the determinant of the lattice when it is represented as a matrix.

3. Literature Review

Current research in lattice-based cryptography is divided into two large areas: theoretical and implementation. The theoretical side focuses on learning more about and developing algorithms for SVP and other associated problems. Meanwhile, for implementation, researchers use SVP to develop secure encryption schemes.

The field of lattice-based cryptography began with Ajtai proving that solving SVP in the average case is as hard as solving it in the worst-case (Ajtai, 1996). This groundbreaking discovery demonstrated the immense value in creating lattice-based cryptosystems and studying lattice-based problems. Two years later, Ajtai also proved that SVP is NP-Hard (Ajtai, 1998).

There are many existing algorithms for solving the shortest vector problem. These algorithms can be split into two categories: exact and approximate. Each of these algorithms can be measured using time complexity, space complexity, and an *approximation factor*, a. An algorithm with approximation factor a will output a length for the shortest vector in between $\lambda(\mathcal{L})$ and $a\lambda(\mathcal{L})$. Exact algorithms have $a = 1$.

Exact algorithms include the randomized sieve algorithms and the deterministic enumeration algorithms. While these algorithms can produce the exact length of the shortest vector, their exponential time complexity suggests they cannot be used in practice to breach lattice-based cyrptographic protocols. The deterministic enumeration algorithm is a brute-force method that determines a region in the lattice and tests all possible vectors in this space. It was originally created by Kannan, Pohst, and Fincke (Kannan, 1983; Fincke & Pohst, 1985), and the Schnorr–Euchner (Schnorr & Euchner, 1994) variant is implemented in practice. If n is the dimension of the lattice, the time complexity of this algorithm is $2^{O(n \log n)}$. The sieve algorithm was first proposed by Ajtai et al. in 2001 (Ajtai et al., 2001). While its time complexity of $2^{O(n)}$ is theoretically better than that of the enumeration algorithm, in practice it falls short. The sieve algorithm has been explored in numerous papers attempting to increase its efficiency, which are summarized in (Sun et al., 2020).

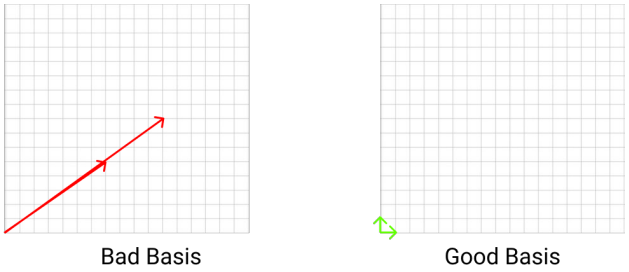


Figure 2: A bad basis (shown in red) versus a good basis (shown in green). Both bases generate all the points in the lattice, but the bad basis has long vectors that are very close together, while the good basis has vectors that are more orthogonal and shorter. When given a good basis, it is much easier to determine the shortest vector in a lattice, whereas with a bad basis it is not immediately obvious what the shortest vector is. This is the idea behind lattice-basis reduction, which turns a bad basis into a good one. Lattice-basis reduction is performed by applying the *Gram-Schmidt Orthogonalization*.

The most well-known approximation algorithms are the LLL (Lenstra et al., 1982) and the BKZ (Schnorr & Euchner, 1994) algorithms. Both algorithms are based on *lattice-basis reduction*, a method of first shortening and orthogonalizing the vectors in the basis in order to find the shortest vector more easily (Figure 2). While both algorithms have polynomial time complexity, the value of a is exponential in terms of n. However, with the BKZ algorithm, one can input a parameter to minimize the value of a at the cost of run time.

Moreover, random lattices can be estimated quite accurately using the Gaussian Heuristic (Ducas, 2018). The existence of such a heuristic implies that some lattices are predictable while others are not.

Creating secure cryptographic protocols based on lattice-problems is another well-researched area in lattice-based cryptography. Exploits in this field as summarized in (Nejatollahi et al., 2019). One of these cryptographic protocols is NTRU, which is equivalent to estimating the solution to CVP using lattice-basis reduction (Hoffstein et al., 1998). Another lattice-based cryptogrpahic protocols is the Learning with Errors (LWE) (Regev, 2009) and its improved

version, the Ring-LWE cryptosystem (Lyubashevsky et al., 2010), both of which are proven to reduce to solving SVP. While NTRU does not have a security reduction proof that suggests it is NP-Hard like Ring-LWE, it is much more efficient and flexible in practice (Nejatollahi et al., 2019).

There is even a lattice-based fully homomorphic encryption scheme (Gentry, 2009). If implemented, a fully homomorphic encryption scheme could allow for data analysis on encrypted data, preventing third-parties from accessing raw user data which threatens security and privacy.

However, while lattice-problems have been widely studied in the general case, there is not much understanding about specific instances of SVP, which could result in interesting conjectures or a validation of SVP having a high average-case complexity. Furthermore, studying which specific instances of SVP yield unpredictable results is vital for the creation of secure cryptographic protocols.

4. Purpose

Thus, in order to supplement the current research by better understanding the patterns or lack thereof in specific SVP problems, we ask the following questions:

1. Can we accurately estimate the length of the shortest vector in a lattice given some features of the lattice?

2. How does the accuracy of our estimates compare across different types of matrices?

Our first research question allows us to generally examine lattice problems through the lens of predictability. The second question extends this analysis to compare different lattices in terms of predictability, which could help define a metric for determining which traits result in unpredictable lattices. Discovering these traits would allow for the creation of lattices that are well-suited to cryptographic protocols.

5. Methodology

In this section we explain how we collected our data and conducted our experiment. We first explain the process of choosing our input data in Section 5.1 and Section 5.2. Then, we walk through the actual process of collecting our data in Section 5.3. Finally, we conclude with explaining how we created the linear regression model and analyzed its outputs in Section 5.4.

5.1. Matrix Selection

In order to select the matrices for our experiment, we attempted to choose a wide variety of matrices to understand how the structure of a matrix plays a role in its predictability.

First, we used random lattices as a baseline. We know that these matrices should be very predictable because of the Gaussian Heuristic.

Next, we chose to use NTRU matrices because of their application to the NTRU cryptosystem. A NTRU matrix is a block matrix comprised of 4 distinct $n \times n$ matrices. The general form of a $2n \times 2n$ NTRU matrix is

$$\begin{pmatrix} 1 & 0 & \dots & 0 & h_0 & h_1 & \dots & h_{n-1} \\ 0 & 1 & \dots & 0 & h_{n-1} & h_0 & \dots & h_{n-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 & h_1 & h_2 & \dots & h_0 \\ 0 & 0 & \dots & 0 & q & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 0 & q & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & 0 & 0 & \dots & q \end{pmatrix},$$

where $q, h_0, h_1, \dots h_{n-1}$ are all chosen randomly.

We also tested a closely related matrix to NTRU called Qary. The main difference between Qary and NTRU is that in a Qary matrix, there are n^2 randomly generated variables in the top right quadrant instead of only n. Furthermore, Qary matrices can have an odd dimension because all the quadrants are not required to be the same size. For dimension n and given input m, the top left quadrant has dimension $(n-m) \times (n-m)$, the top right quadrant has dimension $m \times (n-m)$, the bottom left quadrant has dimension $(n-m) \times m$, and the bottom right quadrant has dimension $m \times m$. We inputted $m = \lfloor \frac{n}{2} \rfloor$.

Furthermore, we tested a Fibonacci version of the NTRU matrices where the initial 2 values are chosen randomly and the rest are the sum of the previous 2 values. We decided to test this Fibonacci-NTRU matrix in order to see if less randomness could be used to yield similar levels of unpredictability. In order to form these matrices, we took all values in this matrix mod 3125 to prevent integer overflow. 3125 or 5^5 was chosen because it is the power of a prime that is sufficiently large to have variation in the data, and the Fibonacci numbers have interesting properties when taken modulo a prime number.

Additionally, we tested a matrices that were a mix of Fibonacci and Qary matrices in order to understand the impact patterned entries had on the performance of the model. To determine this, we tested matrices with k Fibonacci columns and $\lfloor \frac{n}{2} \rfloor - k$ random columns for all values of k between 0 and $\lfloor \frac{n}{2} \rfloor$.

We also tested triangular matrices because of their unique structure. In these matrices, all entries above the main diagonal are 0. Additionally, Figure 13 shows that the heuristic of using $2/3$ the minimum basis vector length seems to be an accurate measure the length of the shortest vector. Thus,

analyzing the predictability of triangular matrices with more features could allow for a more thorough understanding of their behavior. When creating triangular inputs, there is a parameter α that controls the size of the entries in the matrix. We used an input of $\alpha = 0.5$ in order to ensure the program could run efficiently.

Finally, we tested knapsack or intrel matrices because of their irregular behavior due to their unique design of being a $n \times (n+1)$ matrix (see Figure 12). This matrix consists of a $n \times n$ identity matrix and an additional column left of the identity matrix with n randomly generated values.

5.2. Feature Selection

In addition to calculating the length of the shortest vector for each matrix, we also collected the following features:

- The dimension of the matrix
- The length of the minimum vector in the given basis
- The sum of all the entries in the lattice
- The value of the maximum entry in the matrix
- The value of the minimum entry in the matrix
- The average value of all entries in the matrix
- The sum of the entries in the diagonal (starting at the upper left and going to the bottom right) of the matrix
- The length of the diagonal of the matrix (or the diagonal norm)
- The determinant of the matrix [1]
- The Gaussian Heuristic without the constant factors, or $\sqrt{n} \cdot \det(L)$ where n is the dimension of the lattice.

Some of these features, such as the sum of all the entries in the lattice or the length of the diagonal were chosen randomly and do not have any direct correlation to the shortest vector length. We used such random features in order to give the machine learning algorithm a more accurate picture of the input matrix.

However, the Gaussian Heuristic, length of the minimum vector in the given basis, dimension, and determinant of the matrix were all chosen because experimental results suggest relationships exist between the length of the shortest vector and these values for random lattices (see Appendix A.1).

5.3. Data Collection

We tested the different lattices using the fpylll library (development team, 2021). We ran all programs on the cloud using the Sage 9.3 kernel on CoCalc Jupyter Notebook.

While the fpylll library does offer a program for calculating the exact length of the shortest vector, this function only works until dimension 10. The enumeration algorithm for calculating the exact length of the shortest vector was also inefficient for higher dimensions.

Thus, we chose to use the BKZ approximation algorithm to collect our data. We used a high block size of 50 to ensure exact or close to exact results for dimensions up to 50.

For most matrices, we tested up until dimension 51 to ensure there were 50 different dimensions represented in the data. However, for some lattices, we could only go until a dimension of about 40 due to the long computation time required and the program timing out. Table 1 summarizes the highest dimension used for each type of lattice.

For each dimension, we collected 100 different samples of matrices to ensure that there was enough data for the machine learning algorithm to function accurately. If we could only collect data until about dimension 40, we collected 120 samples instead of 100 to reduce the disparity in data set size. The exact number of samples taken for each lattice is shown in Table 1. The random numbers in the matrices had a bit size up to 10.

5.4. Linear Regression Model

Before applying a linear regression model on the data, we analyzed the plots between the shortest vector length and various features to determine if there were polynomial relationships. As a result, we added a few features, including:

- The square root of the sum of all entries
- The length of the diagonal squared
- The determinant raised to the power $\frac{2}{n}$
- The square root of the dimension
- The log of the determinant
- The log of the determinant raised to the power $\frac{2}{n}$

Next, we randomly shuffled the data before splitting it into train, validation, and test sets. This ensured that each set

[1] Note that the determinant is not defined for Knapsack matrices since they are $d \times (d+1)$. Hence, the determinant and related values were not computed for knapsack matrices. However, we can still compute the Gaussian Heuristic using an alternate formation that does not involve the determinant.

Matrix Type	Dimension Range Tested	Number of Samples per Dimension	Total Number of Samples Tested
Random	[2, 51]	100	5000
NTRU	All evens in [4, 44]	250	5250
NTRU Fib	All evens in [2, 40]	250	5000
Qary	[2, 42]	125	5125
Qary Fib	[2,45]	120	5280
Triangular	[2, 47]	120	5520
Knapsack	[2, 51]	100	5000

Table 1: Sample Specifications of Matrices Tested. The first column on the left indicates the type of matrix. The next column indicates the range of integral dimension values that were tested. To the right of this column, the number of samples collected per each dimension is given. Finally, the last column is the total number of samples tested, which is the product of the number of dimensions tested as given by the range in columns 2 and the number of samples per dimension in column 3.

had a mix of dimensions. We used 80% of the entire data set for train data, and 10% for each of the validation and train sets.

Once the data was prepared, we used the scikit-learn library to create a linear regression model and find the coefficients of best fit. To evaluate the model, we also computed the R^2 value.

In order to determine which coefficients had the most impact on the model, we standardized the entire data set and used LASSO regression. We modified the value of alpha until only about 4 features were nonzero. Regularizing the model before determining the most important coefficients allowed us to be more confident that the features with the highest absolute value were related to the length of the shortest vector instead of coincidentally summing to the shortest vector length.

6. Results and Discussion

We split our results into two sections which correspond to our two research questions. In Section 6.1 we look more broadly at the performance of our model in estimating the length of the shortest vector. Then, we take our analysis one step further in Section 6.2 by comparing the results of our model across different types of matrices and determining the traits that differentiate predictable and unpredictable lattices.

6.1. Performance of Model

Table 2 provides an overview of the performance of the linear regression machine learning model. Quite surprisingly, the model could explain at least 90% of the variability for all matrices except for the Fibonacci ones. NTRU matrices were the most predictable, with an R^2 value of approximately 0.996.

While it is initially alarming that the matrices used in cryptographic protocols are the most predictable by the machine learning model, these results do not necessarily indicate that NTRU is not secure. We only tested until dimension 44, when in practice dimensions greater than 300 are used. Thus, an analysis of higher dimensions must be conducted in order to rigorously evaluate the security of the NTRU cryptosystem.

However, the fact that NTRU has the highest R^2 value still suggests that it might not be the best matrix for cryptographic protocols in terms of predictability. Especially since NTRU relies on an approximate version of CVP, a machine learning model that can estimate the length of the shortest vector in an NTRU lattice still poses a significant threat to its security.

In terms of the important features, it is not surprising that features relating to the Gaussian Heuristic, the determinant, the minimum basis vector length, and the dimension are among the most significant features because these have been experimentally proven to have a strong correlation with the length of the shortest vector (see Appendix A.1). However, it is surprising that features relating to the sum of all entries in the matrix, the average entry, and the length of the diagonal are among the most important features.

In summary, our general results show that a machine learning algorithm can quite accurately estimate the length of the shortest vector for certain matrices with small dimensions. This suggests a potential vulnerability in pre-existing cryptosystems, but formally making this claim requires a more thorough analysis of matrices with higher dimensions. Furthermore, we only analyze these cryptographic protocols from the lens of predictability, but this is not the only way to evalute the security of a crytographic protocol.

Matrix Type	R^2	Most Significant Features
Random	0.977	$\det(\mathcal{L})^{2/n}$, Gaussian Heuristic, n
Triangular	0.942	Minimum basis vector length, Gaussian Heuristic, length of diagonal
Knapsack	0.935	Gaussian Heuristic, n, sum, average
NTRU	0.996	Square root of n, $\log(\det(\mathcal{L}))$, average
NTRU Fibonacci	0.734	Length of diagonal, square root of n Gaussian Heuristic, n
Qary	0.970	Gaussian Heuristic, $\det(\mathcal{L})^{2/n}$, $\log(\det(\mathcal{L})^{2/n})$, sum
Qary Fibonacci	0.498	Gaussian Heuristic, average, square root of n, minimum basis vector length

Table 2: Performance of Linear Regression Machine Learning Model. The left column gives the type of matrix, the middle column is the R^2 value to 3 significant digits, and the right column lists the most important features, starting with the most important feature. The most important features have coefficients with the highest absolute value when we used LASSO regression.

6.2. Connections Between Matrix Structure and Predictability

In this subsection, we compare how the performance of our model varies across different types of matrices. We primarily use correlation matrices in order to gain a deeper understanding of the data in Table 2. We begin by looking at random matrices, then the differences between NTRU and Qary matrices, before concluding with taking a closer look at Fibonacci matrices.

6.2.1. Random Matrices

In Figure 3 we can see that most variables have a high pairwise linear correlation. Furthermore, the correlation is most often a positive one. The abundance of correlation for random matrices is to be expected, as with a random lattice most features depend heavily on the dimension. Since the length of the shortest vector has a high correlation with the dimension, it has a high correlation with most other variables as well, with the exception of the maximum, minimum, and average values and the determinant.

6.2.2. NTRU vs Qary

Table 2 shows that Qary matrices have a lower R^2 value than NTRU matrices, especially when comparing the Fibonacci variants of the two. This could have to do with NTRU's unique structuring of the top right quadrant which relies heavily on diagonals. Another possible explanation is that NTRU matrices are more similar to each other since each block is an $\frac{n}{2} \times \frac{n}{2}$ matrix. However, for Qary matrices,

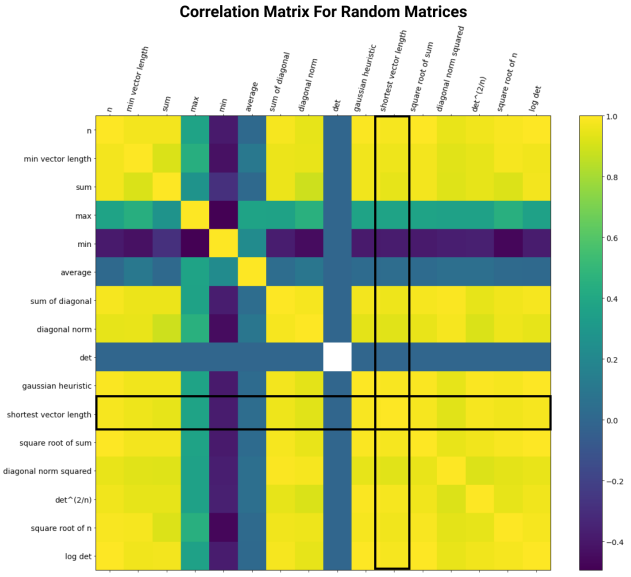
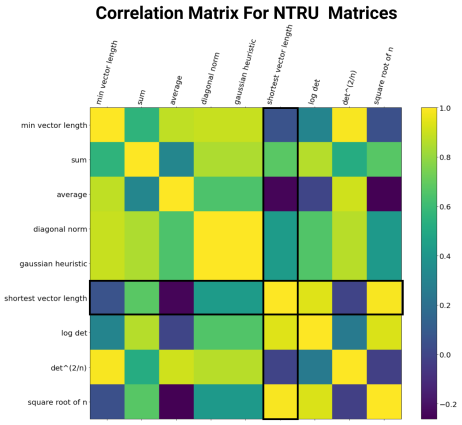
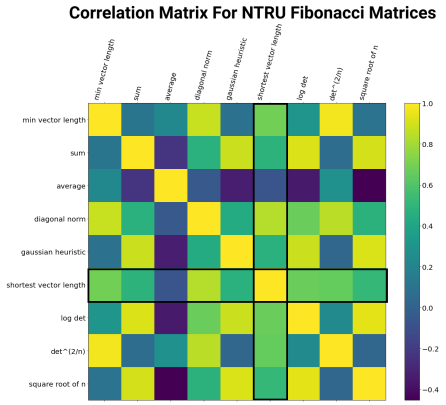


Figure 3: Correlation matrix for random lattices. Each square reflects the correlation coefficient (R) between two variables. As shown in the bar on the right, yellow and generally lighter hues correspond to a higher correlation coefficient, whereas darker hues correspond to a lower correlation coefficient. Dark blue indicates no correlation at all. The column and row corresponding to the shortest vector are highlighted by a black rectangle. The single white square in the matrix occurs due to the large values of the determinant that are unable to be evaluated.

when n is odd, there can be blocks of dimension $\frac{n+1}{2} \times \frac{n-1}{2}$, making these block matrices rectangular instead of square. Having less symmetrical blocks could explain why Qary matrices are less predictable.



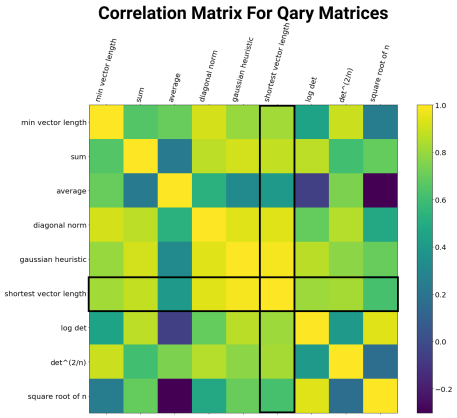
(a) Correlation Matix for NTRU Matrix



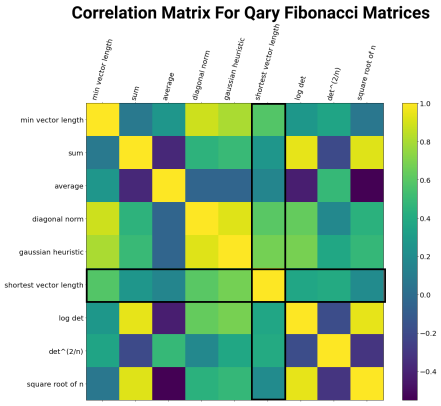
(b) Correlation Matrix for NTRU Fibonacci Matrix

Figure 4: Correlation Matrices for NTRU Matrices. Each square reflects the correlation coefficient (R) between two variables. As shown in the bar on the right, yellow and generally lighter hues correspond to a higher correlation coefficient, whereas darker hues correspond to a lower correlation coefficient. Dark blue indicates no correlation at all. The column and row corresponding to the shortest vector are highlighted by a black rectangle. Note that the pure NTRU matrices appear to have lighter hues overall compared to the NTRU Fibonacci Matrix, especially in the upper left quadrant.

We can see this in Figure 4 and Figure 5. The pure NTRU matrix has a very strong correlation to the square root of the dimension, whereas its Qary counterparts are much more weakly correlated to the dimension. It is likely that the homogeneity of NTRU matrices with respect to dimension plays a role in this high correlation. Furthermore, since dimension is not a significant feature for pure Qary matrices,



(a) Correlation Matix for Qary Matrix



(b) Correlation Matrix for Qary Fibonacci Matrix

Figure 5: Correlation Matrices for Qary Matrices. Each square reflects the correlation coefficient (R) between two variables. As shown in the bar on the right, yellow and generally lighter hues correspond to a higher correlation coefficient, whereas darker hues correspond to a lower correlation coefficient. Dark blue indicates no correlation at all. The column and row corresponding to the shortest vector are highlighted by a black rectangle. Note that the two correlation matrices look nearly identical in terms of patterns, but the correlation matrix on the right has darker hues, indicating less correlation.

the R^2 value for Qary matrices is still close to that of NTRU matrices.

On the other hand, both types of Qary matrices have a stronger correlation to the Gaussian Heuristic than the NTRU matrices. This is likely due to the more randomized nature of Qary matrices.

In conclusion, Qary and NTRU matrices primarily differ in terms of symmetry, which could be a possible explanation for why NTRU matrices have a slightly higher R^2 value than Qary matrices.

6.2.3. Fibonacci Matrices

Another unexpected result in Table 2 is the substantially lower R^2 values of the Fibonacci matrices. A possible explanation for this is the partnered nature of Fibonacci matrices. The Gaussian Heuristic works best for random matrices, and Fibonacci matrices are less random than NTRU and Qary matrices, so the Gaussian Heuristic would be less accurate for Fibonacci matrices.

When we examine the correlation matrices for pure Qary and Qary Fibonacci matrices, we can see that pure Qary matrices have a significantly higher correlation to the Gaussian Heuristic. Since the Gaussian Heuristic is a significant feature for both of these matrices, the Fibonacci model correspondingly has a worse performance.

The correlation between the Gaussian Heuristic and the shortest vector length is about the same for both the pure NTRU and NTRU Fibonacci matrices, with pure NTRU being slightly worse. However, this does not impact the R^2 value of pure NTRU matrices because the Gaussian Heuristic is not a significant feature.

On the other hand, square root of n is a significant feature for both pure NTRU and NTRU Fibonacci matrices, and the shortest vector length has a much stronger correlation to the square root of n for pure NTRU matrices. As a result, the pure NTRU R^2 value is much higher than the NTRU Fibonacci R^2 value.

We can further quantify the impact of randomness of lattices on predictability by analyzing the matrices we tested that are a combination of classic Qary and Qary Fibonacci matrices. In Figure 6 and Figure 7 we can see the generally negative correlation the number of Fibonacci columns has on the R^2 value. The negative correlation supports the claim that more randomness in entries correlates with the shortest vector length being more predictable.

In both graphs, the last peak in the data appears when approximately 0.7 of the columns are Fibonacci numbers. After this last peak, the data sharply decreases in both cases. This suggests that having a clear majority (about 70%) of patterned columns in a matrix significantly reduces the predictability of the length of the shortest vector.

However, we also cannot rule out the possibility that the Fibonacci matrices have a low R^2 value because we did not input the right features into the linear regression model. Conducting an analysis with more features could allow us to determine if this is the case.

A bit tangentially, we can note that the R^2 value in both of these graphs is significantly lower than when matrices across several dimensions were considered. This is likely due to the fact that most features are heavily related to the dimension. Thus, only analyzing one dimension ignores cross-dimensional relationships between features.

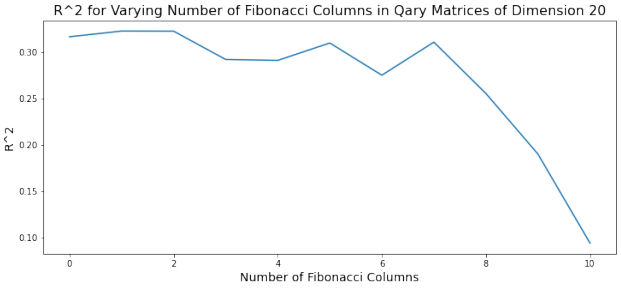


Figure 6: R^2 for Varying Number of Fibonacci Columns in Qary Matrices of Dimension 20. This plot shows how adding more columns with Fibonacci numbers and subsequently reducing the number of columns with random numbers affects the R^2 value. Note that even though the Qary matrix has dimension of 20, the block matrix we are adjusting only has dimension of 10, so the x-axis only goes from 0 to 10. While there is a generally negative correlation, for initial values the R^2 value stays close to 0.3. After the number of Fibonacci columns is greater then 7, the R^2 value plummets downwards.

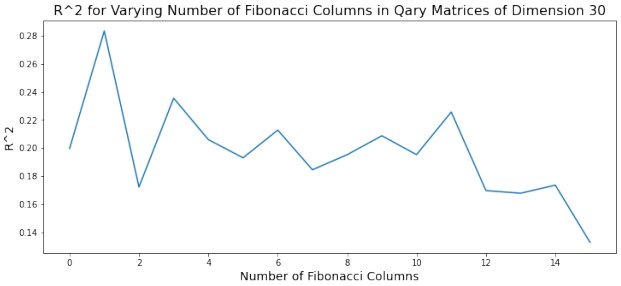


Figure 7: R^2 for Varying Number of Fibonacci Columns in Qary Matrices of Dimension 30. This plot shows how adding more columns with Fibonacci numbers and subsequently reducing the number of columns with random numbers affects the R^2 value. Note that even though the Qary matrix has dimension of 30, the block matrix we are adjusting only has dimension of 15, so the x-axis only goes from 0 to 15. While there is a generally negative correlation, it is not a clean correlation. For initial values the R^2 value behaves erratically but overall decreases slightly. However, once the number of Fibonacci columns is greater then 11, the R^2 value plummets downwards, like with the previous plot.

To summarize, we noticed that there was a significant difference in the percentage of variability of the data explained by the model depending on whether the entries in the matrix were patterned or random. This result shows promise for designing lattices with patterned entries that might be less predictable by a machine learning algorithm.

7. Conclusion and Future Work

In this study, we used a linear regression machine learning model to evaluate the predictability of different types of matrices for small dimensions. Using a list of features of the matrix, we found that in most cases we could explain at least 90% of the variability of the length of the shortest vector in these matrices. In particular, our model had an R^2 value of 0.996 for NTRU matrices that are used in cryptosystems. We also found that matrices with fewer random entries and with a less homogeneous structure, especially as related to the diagonals, were less predictable.

Our work proposes a new metric for determining how predictable the length of the shortest vector in a lattice is. Furthermore, we discover certain traits that could characterize unpredictable instances of lattice problems. Ultimately, these steps are key in refining and gaining a better understanding of lattice-based cryptography in order to incorporate lattice-based quantum resistant cryptosystems into society.

Future work includes refining the linear regression model by adding more features and using it to analyze more matrices, especially those that relate to other cryptographic protocols such as Ring-LWE. Applying deep learning instead of linear regression could also allow for more accurate predictions. Also, including higher dimensions in this analysis would allow for stronger conclusions about whether or not current lattice-based cryptographic protocols are secure.

Another possible avenue for future work is using machine learning to synthesize a matrix embodying the traits of unpredictable matrices. Creating such matrices could allow for the creation of more secure lattice-based cryptographic protocols.

8. Acknowledgements

I would like to thank my mentor, Jacob Imola, without whom this project would not have been possible. Specifically, I want to thank Jacob for exposing me to the field of lattice-based cryptography and helping me think critically about my results. In addition, I would like to thank Summer Stem Institute for this amazing opportunity.

References

Ajtai, M. Generating hard instances of lattice problems (extended abstract). In *Proceedings of the twenty-eighth annual ACM symposium on Theory of Computing*, STOC '96, pp. 99–108, New York, NY, USA, July 1996. Association for Computing Machinery. ISBN 978-0-89791-785-8. doi: 10.1145/237814.237838.

Ajtai, M. The shortest vector problem in L_2 is *NP*-hard for randomized reductions (extended abstract). In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, STOC '98, pp. 10–19, New York, NY, USA, May 1998. Association for Computing Machinery. ISBN 978-0-89791-962-3. doi: 10.1145/276698.276705.

Ajtai, M., Kumar, R., and Sivakumar, D. A sieve algorithm for the shortest lattice vector problem. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pp. 601–610, 2001.

Chuang, Y.-L., Fan, C.-I., and Tseng, Y.-F. An Efficient Algorithm for the Shortest Vector Problem. *IEEE Access*, 6:61478–61487, 2018. ISSN 2169-3536. doi: 10.1109/ACCESS.2018.2876401. Conference Name: IEEE Access.

development team, T. F. fpylll, a Python wraper for the fplll lattice reduction library, Version: 0.5.6. 2021.

Ducas, L. Shortest vector from lattice sieving: a few dimensions for free. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 125–145. Springer, 2018.

Fincke, U. and Pohst, M. Improved methods for calculating vectors of short length in a lattice, including a complexity analysis. *Mathematics of computation*, 44(170):463–471, 1985.

Gama, N. and Nguyen, P. Q. Predicting lattice reduction. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 31–51. Springer, 2008.

Gentry, C. Fully homomorphic encryption using ideal lattices. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pp. 169–178, 2009.

Hoffstein, J., Pipher, J., and Silverman, J. H. Ntru: A ring-based public key cryptosystem. In *International Algorithmic Number Theory Symposium*, pp. 267–288. Springer, 1998.

Kannan, R. Improved algorithms for integer programming and related lattice problems. In *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, pp. 193–206, 1983.

Lenstra, A. K., Lenstra, H. W., and Lovász, L. Factoring polynomials with rational coefficients. *Mathematische annalen*, 261(ARTICLE):515–534, 1982.

Lyubashevsky, V., Peikert, C., and Regev, O. On ideal lattices and learning with errors over rings. In *Annual international conference on the theory and applications of cryptographic techniques*, pp. 1–23. Springer, 2010.

Nejatollahi, H., Dutt, N., Ray, S., Regazzoni, F., Banerjee, I., and Cammarota, R. Post-quantum lattice-based cryptography implementations: A survey. *ACM Computing Surveys (CSUR)*, 51(6):1–41, 2019.

Regev, O. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):1–40, 2009.

Schnorr, C.-P. and Euchner, M. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Mathematical programming*, 66(1):181–199, 1994.

Shor, P. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pp. 124–134, Santa Fe, NM, USA, 1994. IEEE Comput. Soc. Press. ISBN 978-0-8186-6580-6. doi: 10.1109/SFCS.1994.365700.

Sun, Z., Gu, C., and Zheng, Y. A review of sieve algorithms in solving the shortest lattice vector problem. *IEEE Access*, 8:190475–190486, 2020.

A. Appendix

A.1. Supplemental Figures

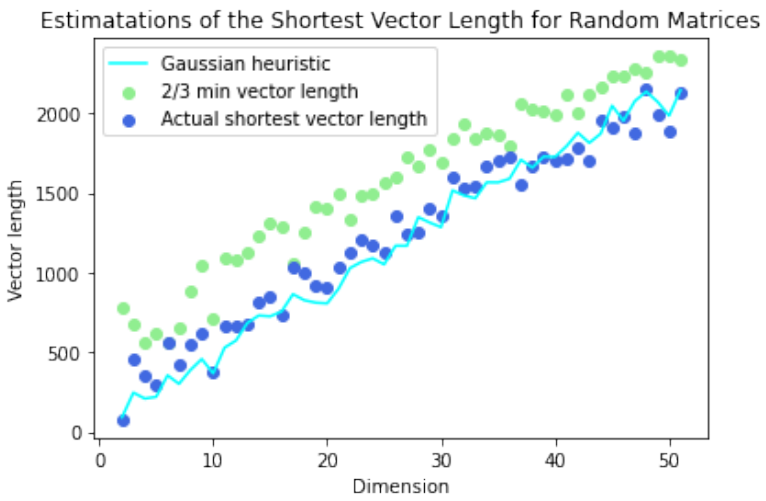


Figure 8: Estimations of the Shortest Vector Length for Random Matrices. The light green dots indicate a heuristic we experimentally determined, which is $2/3$ times the length of the minimum vector in the basis. The light blue line shows the Gaussian Heuristic, which much more accurately predicts the length of the shortest vector.

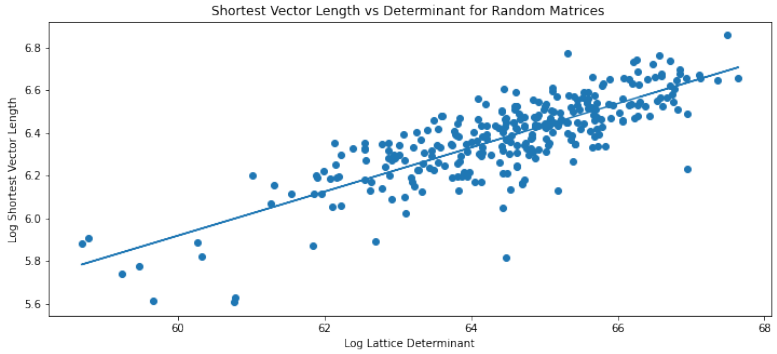


Figure 9: The length of the shortest vector compared to the determinant for random matrices. The line through the scatter plot indicates the line of best fit. Notice that we took the logarithm of the shortest vector length and determinant in order to find a linear correlation between the two variables. This indicates that there is a power law relationship between the length of the shortest vector and the determinant.

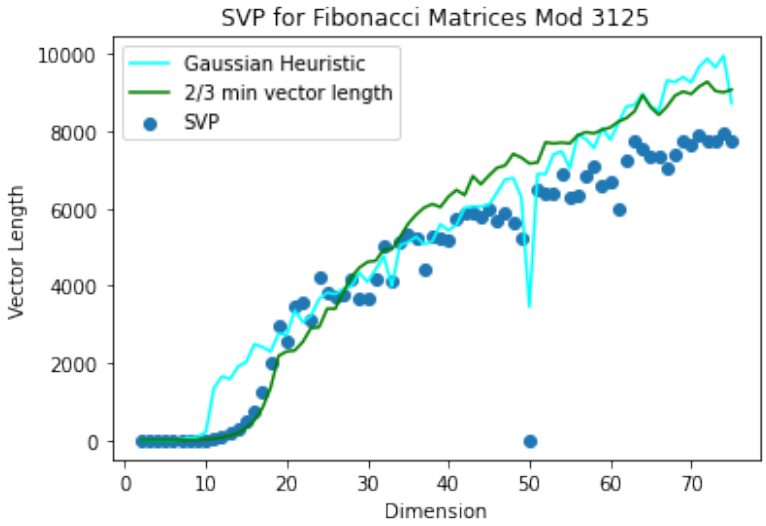


Figure 10: Estimations of the Shortest Vector Length for Fibonacci Matrices. For these matrices, we took the regular Fibonacci numbers modulo 3125 and arranged them by rows instead of columns like we do in the paper. Notice that the Gaussian Heuristic is not as accurate for these matrices as it was for random matrices, especially with higher dimensions. There are also several instances where $2/3$ times the minimum basis vector length more closely estimates the length of the shortest vector.

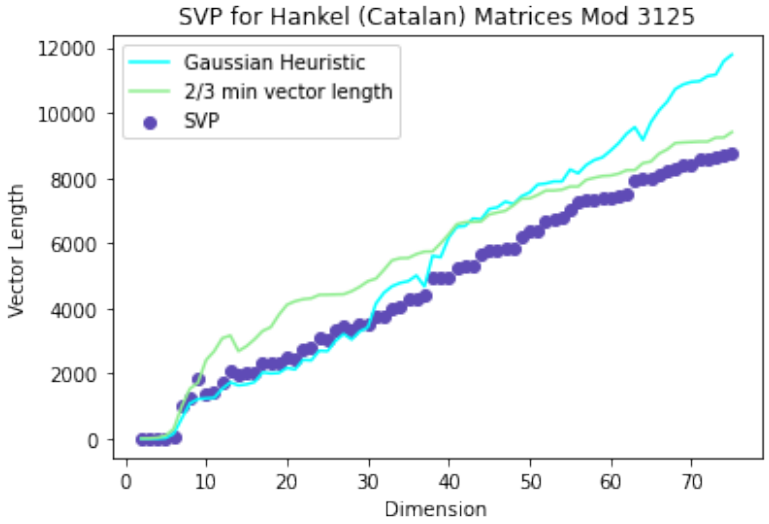


Figure 11: Estimations of the Shortest Vector Length for Catalan Matrices. For these matrices, we took the regular Catalan numbers modulo 3125 and arranged them by rows. Notice that the Gaussian Heuristic is not as accurate for these matrices as it was for random matrices or Fibonacci matrices, especially with higher dimensions. There are also several instances where $2/3$ times the minimum basis vector length more closely estimates the length of the shortest vector. Past dimension 40, $2/3$ times the minimum basis vector length appears to be the better heuristic.

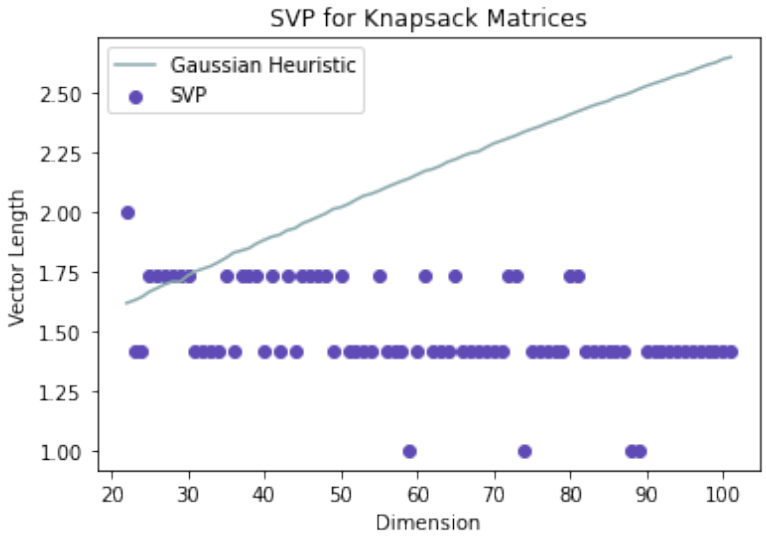


Figure 12: Shortest Vector Length for Knapsack Matrices. Compared to the other plots, these matrices are the most irregular and unable to be predicted by the Gaussian Heuristic. While the other plots had a generally positive correlation, the length of the shortest vector for Knapsack matrices appears to not grow with the dimension but rather remain stagnant.

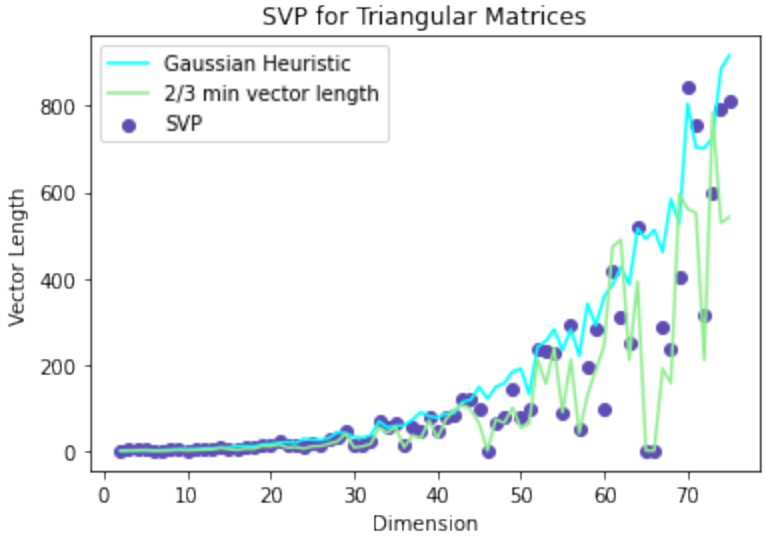


Figure 13: Estimations of the Shortest Vector Length for Triangular Matrices. Note how closely the heuristic of $2/3$ the minimum basis vector length matches the length of the actual shortest vector in the lattice. This likely has to do with the large number of zeros present in Triangular Matrices. Furthermore, note that while the Gaussian Heuristic is able to generally follow the trend of the shortest vector length, it is not as close-fitted as $2/3$ the minimum basis vector length.