CS754 Project Report

Arpon Basu
Shashwat Garg

Spring 2022

Contents

Introduction

This is a report of our project which involved implementing and extending the paper "**Enhancing Sparsity by Reweighted l_1 Minimization**", in which we investigate how a simple extension of the l_1 norm minimization principle gives us significantly better results than the vanilla l_1 algorithm alone.
In this report, we shall explain how we (re)implemented all the results already demonstrated in the paper, as well as extended them further using some insights of our own.
This is how our report will be organized: We shall first explain briefly what the new algorithm proposes to do, and then we shall explore it's various applications in the field of compressive sensing and also explain our extensions to the algorithm.

1 A Brief Description of the Algorithm

The fundamental problem in sparse recovery is to recover a sparse vector $\boldsymbol{x} \in \mathbb{R}^n$ from an undersampled measurement $\boldsymbol{y} = \boldsymbol{\Phi x}$, where $\boldsymbol{\Phi} \in \mathbb{R}^{m \times n}$ is the sensing matrix, such that $m < n$.
Now, the basic premise of l_1 minimization says that the solution of optimization

$$\boldsymbol{x^* := \arg\min_{y=\Phi x} \|x\|_1}$$

will for sparse enough $\boldsymbol{x}$ and large enough m act as a *suitable proxy* for the l_0 norm of $\boldsymbol{x}$, which counts the number of non-zero entries of $\boldsymbol{x}$. Indeed, we declare that our vector $\boldsymbol{x}$ has been recovered successfully only if $\|\boldsymbol{x} - \boldsymbol{x}^*\|_\infty < \boldsymbol{\delta}$ for some small threshold value δ.
The proposed algorithm takes this relation between l_0 and l_1 norms one step further: Note that for any vector $\boldsymbol{v}$, $\boldsymbol{W\|v\|_1 = \|v\|_0}$, where $\boldsymbol{W := \mathbf{diag}(\frac{1}{|v_1|}, \frac{1}{|v_2|}, ..., \frac{1}{|v_n|})}$ (assuming all entries of $\boldsymbol{v}$ are non-zero). Thus, we do exactly this: After obtaining the preliminary estimate of $\boldsymbol{x}$ from the l_1 minimization algorithm, we *iteratively* improve upon it by optimizing, in the subsequent iterations, not $\|\boldsymbol{x}\|_1$, but $\|\boldsymbol{Wx}\|_1$, where at each iteration, $\boldsymbol{W}$ is modified according to the $\boldsymbol{x}$-estimate obtained in the previous iteration, where $\boldsymbol{W}$ is a diagonal matrix whose diagonal entries are (approximately) inversely proportional to the corresponding entries of $\boldsymbol{x}$, so as to simulate the l_0 norm with the l_1 norm.
Finally, note that one can't directly substitute $\boldsymbol{w_{ii} = \frac{1}{|x_i|}}$ for the obvious reason that x_i may be 0. Thus, instead of directly taking the reciprocal of the absolute values of the entries of $\boldsymbol{x}$, we assume the mathematical relation

$$\boldsymbol{w_{ii} = \frac{1}{|x_i| + \varepsilon}}$$

where ε is a small constant to avoid division by zero. However, far from being a trifling numerical factor, as we shall see below, ε is an important hyperparameter of this algorithm.

2 Exploring the Algorithm

2.1 Role of the hyperparameter ε

As mentioned earlier, the hyperparameter ε has an important effect in the performance of the algorithm: Indeed, for very large values of ε we are effectively ignoring the values of $\boldsymbol{x}$, thus effectively making $\boldsymbol{W \approx \frac{1}{\varepsilon} I}$, which defeats our purpose of reweighting the $\boldsymbol{x}$ vector for the next iteration. Too small εs on the other hand tend to make the algorithm unstable (as the algorithm is perturbed by transient misleading entries of $\boldsymbol{x}$), thus once again reducing performance.
As witnessed by the graphs below, an optimum value of ε seems to be somewhere between 0.1 and 1.

2.2 Our Innovation: Introduction of New Cost Functions

As mentioned in the paper too, the reweighting relation $w_{ii} = \frac{1}{|x_i|+\varepsilon}$ can be viewed as the derivative of the cost function

$$f(x) := \sum_{i=1}^{n} \log(|x_i| + \varepsilon)$$

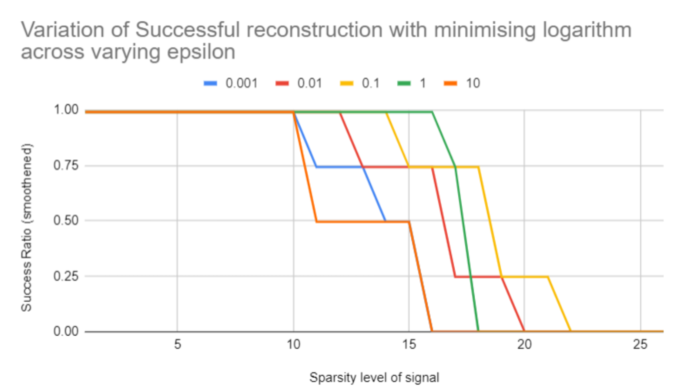
Inspired by this, we introduce some more common cost functions from statistical analysis and machine learning ourselves, which include

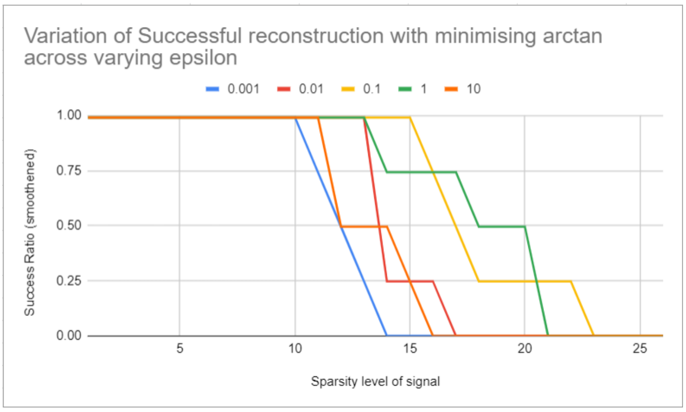
$$f(x) := \sum_{i=1}^{n} \arctan(|x_i/\varepsilon|) \implies w_{ii} = \frac{1}{x_i^2 + \varepsilon^2}$$

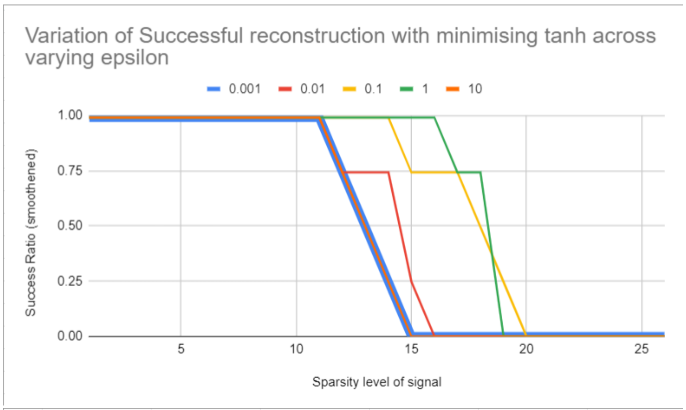
$$f(x) := \sum_{i=1}^{n} \tanh(|x_i/\varepsilon|) \implies w_{ii} = \operatorname{sech}^2(x_i/\varepsilon)$$

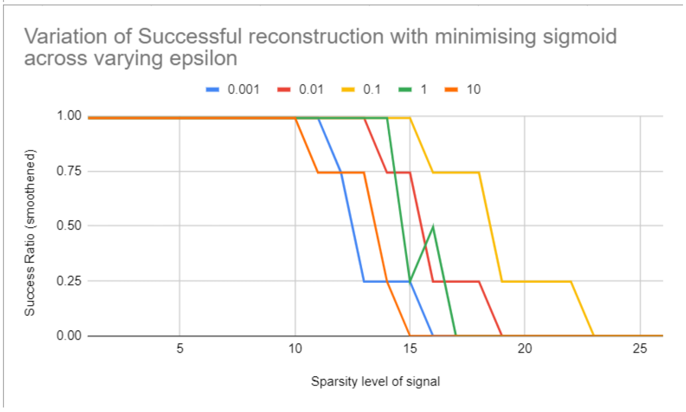
$$f(x) := \sum_{i=1}^{n} \operatorname{sigmoid}(|x_i/\varepsilon|) \implies w_{ii} = \sigma(x_i)(1 - \sigma(x_i))$$

The performance for these is included below.



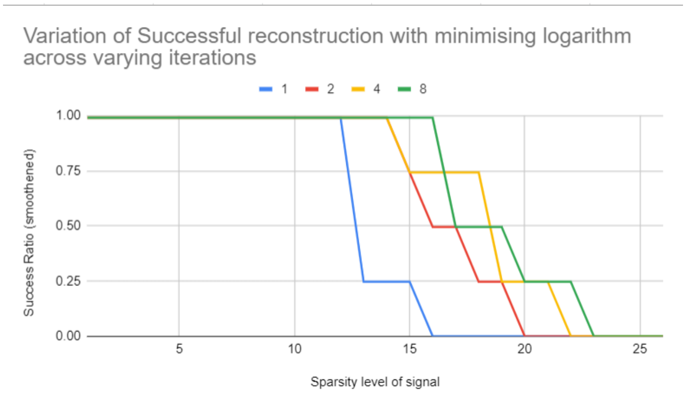


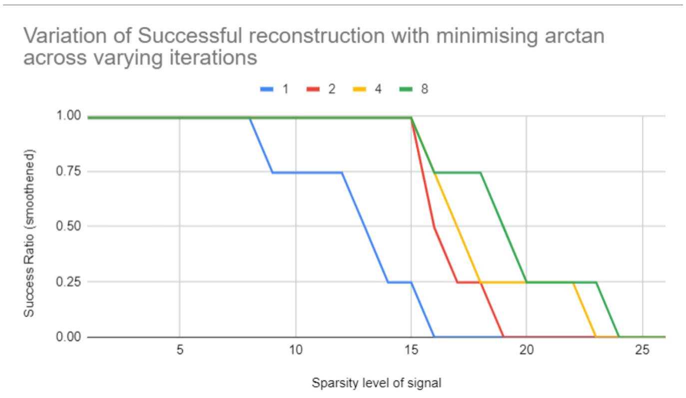


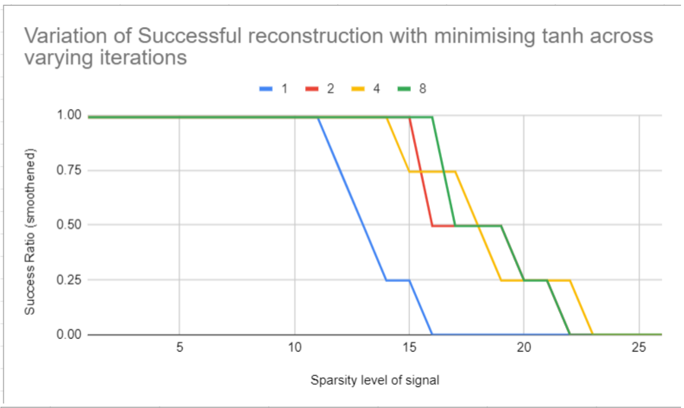


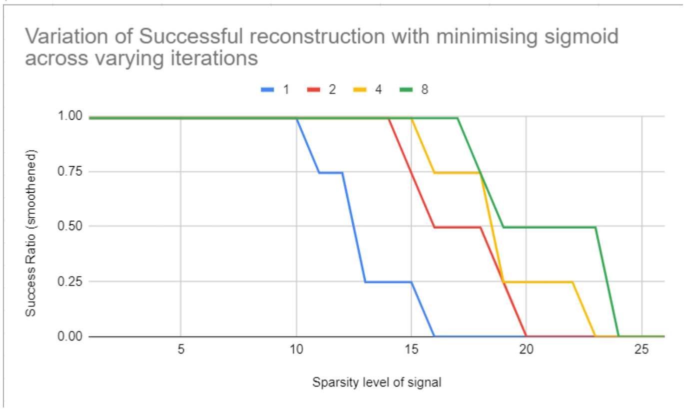
2.3 Effect of Number of Reweighting Iterations

It's obvious that the number of reweighting iterations will improve accuracy. However, our numerical experiments reveal that if convergence occurs, then it usually does so within 2-3 iterations itself, in most cases.



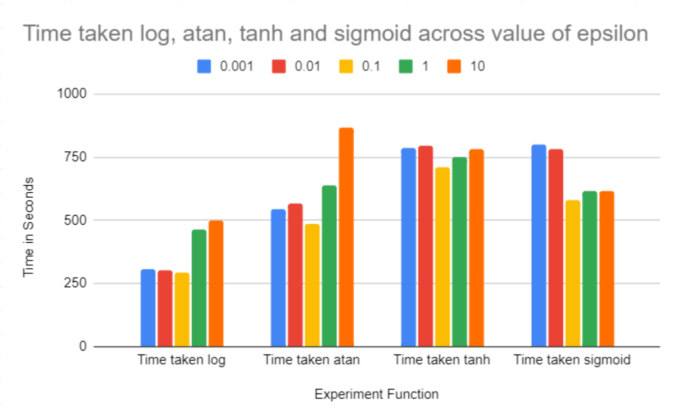






2.4 Comparison of Times Taken

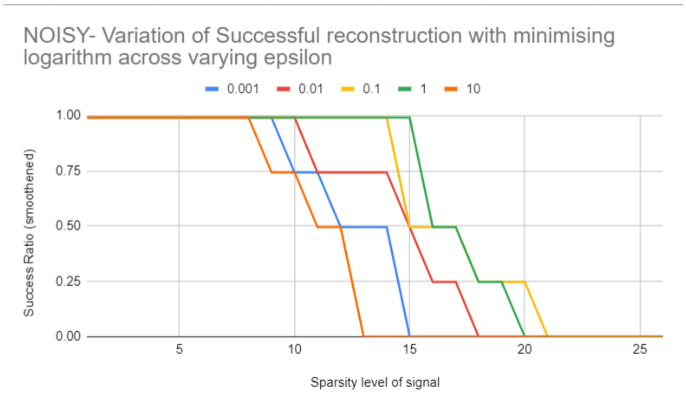
As discussed above, a change in the value of ϵ affects performance a lot. Somewhat surprisingly, it turns out that the value of ε for which the best recovery probability (the probability of recovery is defined as the proportion of vectors recovered successfully when the same instance of the algorithm, with the same hyperparameters is run multiple times) is obtained, is also the most economical in terms of time taken for execution.

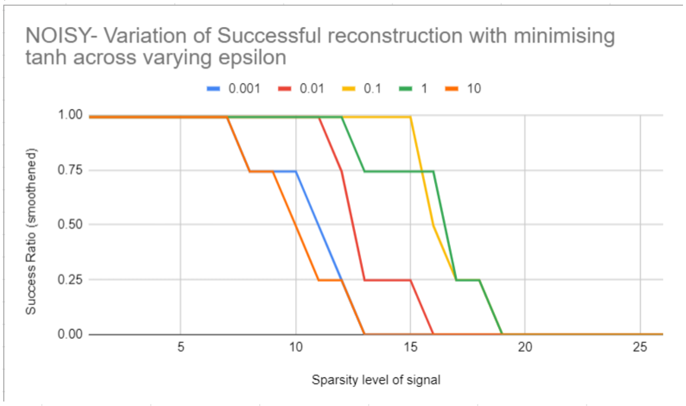


3 Robustness of the Algorithm vis-a-vis Noise

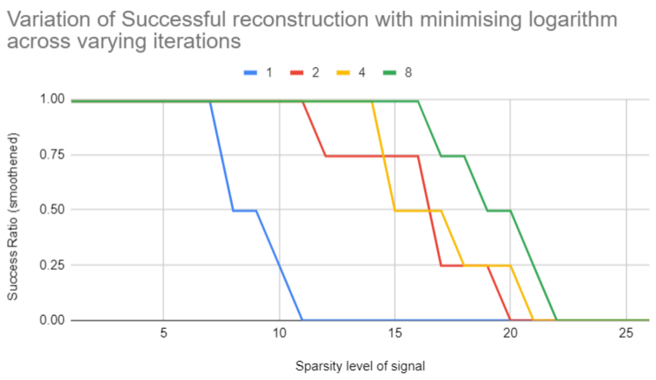
We observe that more or less same results are obtained even when we consider noisy input, ie:- our relation is now $\boldsymbol{y} = \boldsymbol{\Phi x} + \boldsymbol{\eta}$, where $\boldsymbol{\eta}$ is random Gaussian white noise. While carrying out our numerical experiments though, we observe that we have to relax our threshold for successful recovery somewhat to obtain similar results (we took $\delta = 10^{-3}$ for our noiseless case. For the noisy case we had to relax it to $\delta = 10^{-2}$).

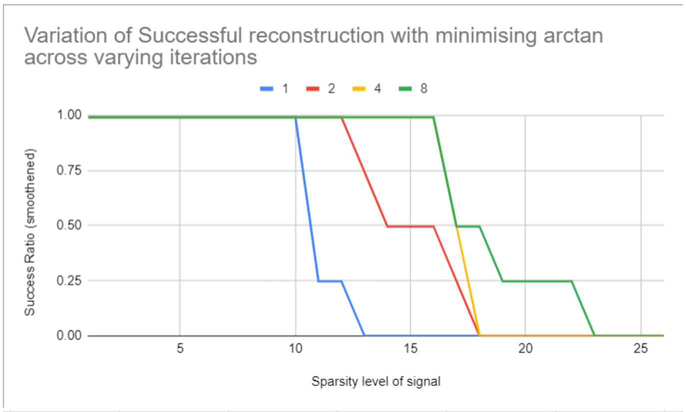
3.1 Different Cost Functions

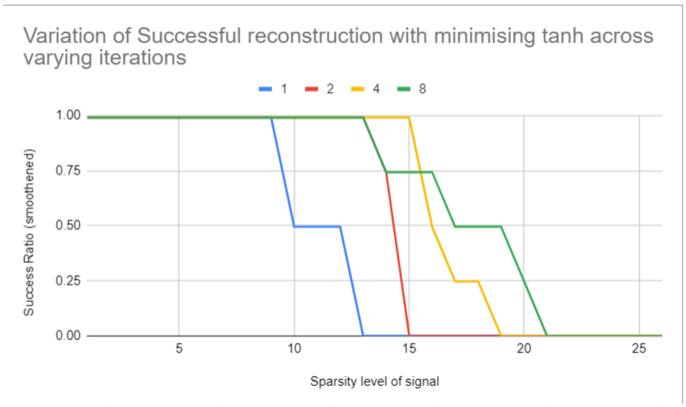


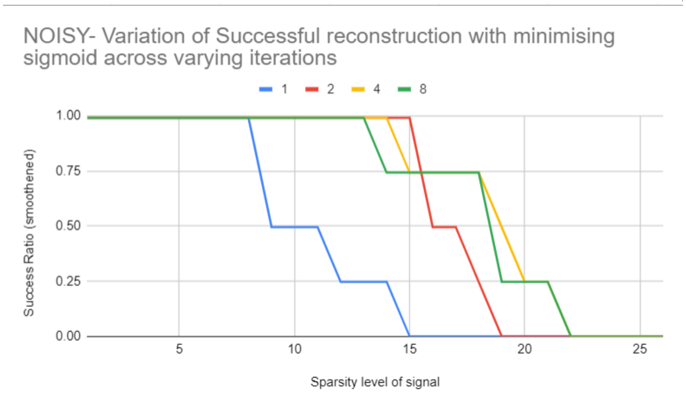


3.2 Effect of Number of Iterations

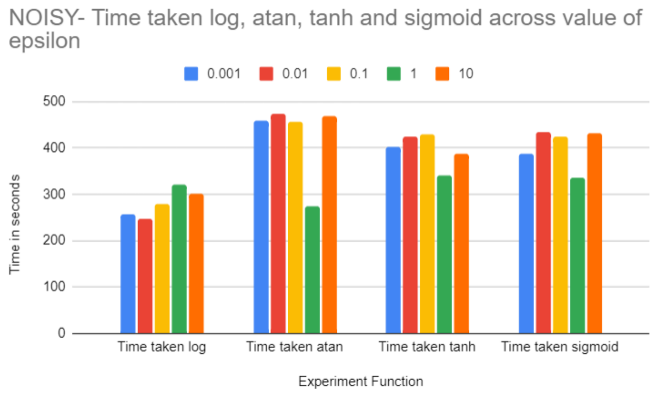


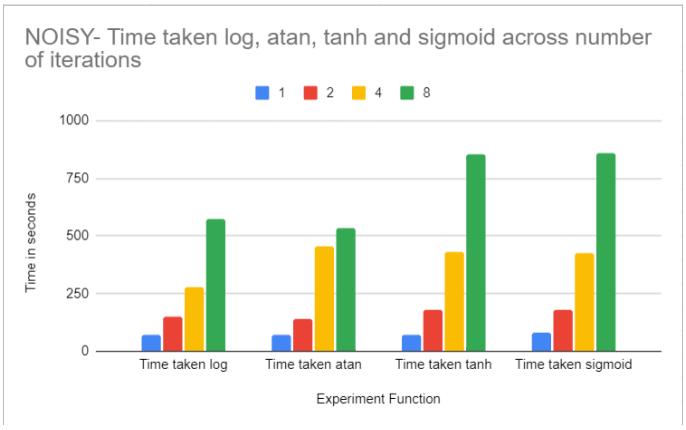






3.3 Comparison of Times Taken in Noisy Case

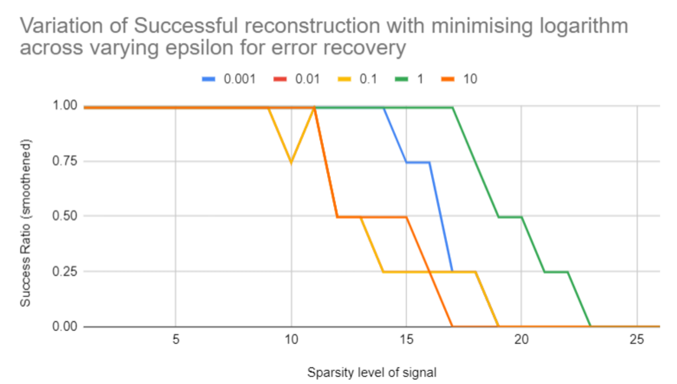




4 Weeding out Errors: How this algorithm helps in error correcting Codes

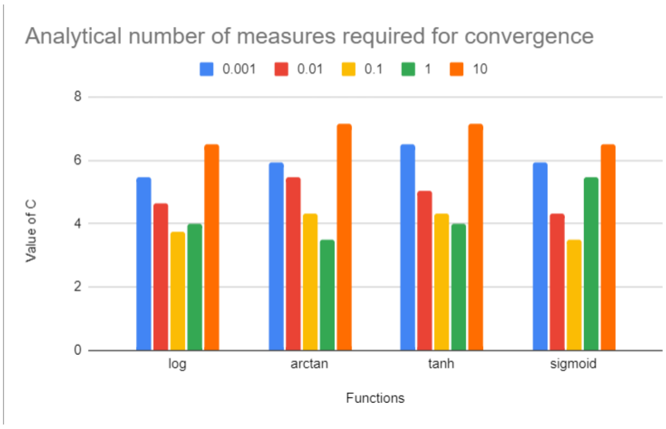
If $m > n$ in Φ, then we enter a field called error correcting codes, in which the sensing matrix is deliberately imbued with some redundancy so as to capture error. Here, instead of minimizing $\|\boldsymbol{Wx}\|_1$, we minimize $\|\boldsymbol{W(y - \Phi x)}\|_1$, because here $\boldsymbol{x}$ isn't the sparse quantity, $\boldsymbol{e := y - \Phi x}$ is.

The results are more or less in line with the results obtained above.



5 Calculating the exact value of the point sparse recovery breaks down

As one may observe from the graphs above, the probability of successful recovery *behaves like a phase transition, ie:- after being fairly constant at 1, it suddenly takes a sharp plunge and plummets to 0 within a short interval, displaying interesting transient phenomena in the midst.* Indeed, as we identified, **the midpoint of the "transient" region can be taken to be the point of theoretical guarantees, which say that we need m to be atleast $Ck\log(\frac{n}{k})$ for recovery to happen**. This data gives us a numerical way of calculating the constant C, as given below.



6 Conclusion

In conclusion, we observe that both concave and convex functions perform similarly in terms of the ability to converge. There is rapid increase in the convergence rate in the second iteration of the weighted L1 norm process itself and further iterations offer only marginal improvement.

This also indicates how the weighted L1 always provides an improvement over the classic L1 norm minimization. Since we observe the same behaviour in case of all functions, it seems that the improvement factor is also robust to choice of the function.

There seems to be a sweet-spot for the hyperparameter where the recovery rate is maximum. Also, this value might differ for the cases with and without noise, as we have already seen above.

In conclusion, an iterative approach using several different tools allowed us to get a better intuition of the behaviour of the algorithms.

There are several future extensions of this project. One can try non-linear weighting strategies to better fit the value of the vectors. We can also try to prove the accuracy and performance metrics theoretically and obtain some new results. We hope to touch on some of these aspects in the future.

Thank you for reading.