Collected Papers 2022 - A Special Issue On Distance Measures and Machine Learning

Mikko I. Malinen
mikko.i.malinen@gmail.com

1st January, 2023

Contents

Preface

This article collection "Collected Papers 2022" contains five articles written in 2022 by Mikko I. Malinen, a Computer Scientist from Joensuu, Finland. The themes of the articles are Distance Measures and Machine Learning. See the connections of the articles to each other in Figure 1.
If you want to cite an article in this collection, you may do it like this:

[1] M.I. Malinen, "The Name of The Article Comes Here", in Collected Papers 2022 - A Special Issue on Distance Measures And Machine Learning, Mikko I. Malinen, Joensuu, Finland, 2023

Joensuu, Finland 22nd November, 2022

Dr. Mikko I. Malinen, Ph.D.

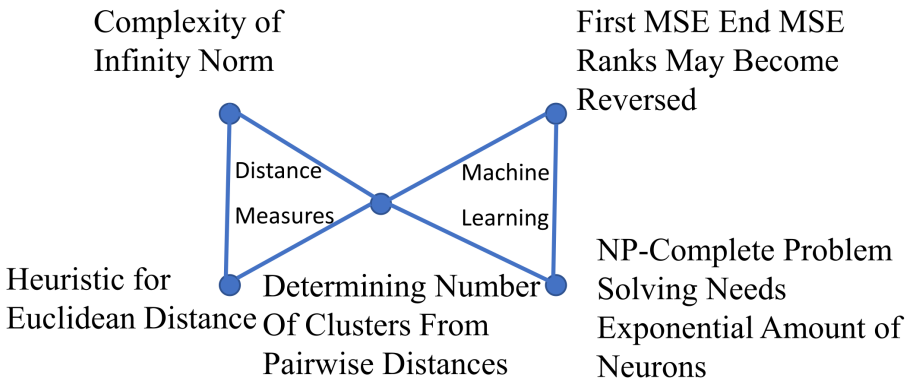

Figure 1: Connections From Articles to Each Other

Chapter 1

A Heuristic for the Euclidean Distance Metric

1.1 Introduction

Distance calculations are important in clustering, pattern recognition and many other fields where points in space are involved. In this article we do research how good the Euclidean distance measure, Manhattan distance measure and the Infinite norm are as heuristics of each other and how effective these heuristics are in speeding up calculations.
In Figure 1.1 we show calculation of three distance measures in two-dimensional space.

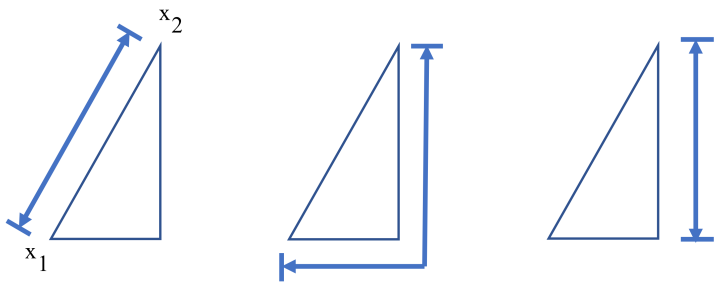

Figure 1.1: Calculation of Euclidean Distance (left), Manhattan Distance (City-Block Distance, 1-norm) (middle) and Infinity Norm (right).

1.2 Efficiency

We base the following calculations to assumptions that

-addition takes one clock cycle
-substraction takes one clock cycle
-multiplication takes four clock cycles
-rising to power of two takes four clock cycles
-taking absolute value takes one clock cycle

on a processor. These are values that are true for example in 8051 microcontroller.

Euclidean distance is calculated as

$$d_{Eucl} = \sqrt{(x_{11} - x_{21})^2 + (x_{12} - x_{22})^2 + ... + (x_{1d} - x_{2d})^2},$$

whre the first subindex denotes point number and the second subindex denotes the feature (coordinate) number. Time to calculate Euclidean distance takes so

$$\begin{aligned}(1+4) \cdot d + (d-1) + T(sqrt) \\ = 6d - 1 + T(sqrt)\end{aligned}$$

clock cycles.

Manhattan distance is calculated as

$$d_{Manh} = \mid x_{11} - x_{21} \mid + \mid x_{12} - x_{22} \mid + ... + \mid x_{1d} - x_{2d} \mid .$$

Time to calclulate Manhattan distance is thus

$$(1+1) \cdot d + d - 1 = 3d - 1.$$

We see that Manhattan distance is more that twice faster to calculate than the Euclidean distance. We did the calculation also for the Infinity norm, and saw that it was faster to calculate than the Euclidean distance, but on average slower to calculate than the Manhattan distance. Therefore we left the Infinity norm out of consideration of using it as a heuristic for Euclidean distance.

1.3 Accuracy

In experiments we used g2 datasets [1] that have two Gaussian clusters with varying dimensionality and varying cluster overlap. An example of a g2 dataset is in Figure 1.2. In Figure 1.3 we see the correlation of Euclidean distances and Manhattan distances as a function of dataset dimensionality. We see that the correlation is high, near one, for all the tested dimensionalities. In Figure 1.4 we see the same correlation as a function of separation of clusters in datasets. We see that the correlation is high for all the tested datasets. In Figure 1.5 we show also a scatter diagram of Euclidean vs. Manhattan distance across datasets with varying dimensionality. The ovals form from a single dataset. We see that the dependency between Euclidean and Manhattan distances is is almost linear.

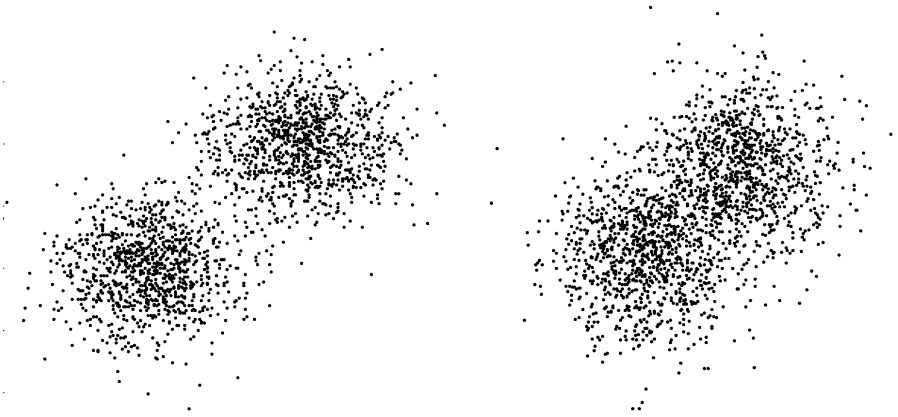

Figure 1.2: Examples of two-dimensional g2 sets. Overlap 30 (left) and 50 (right).

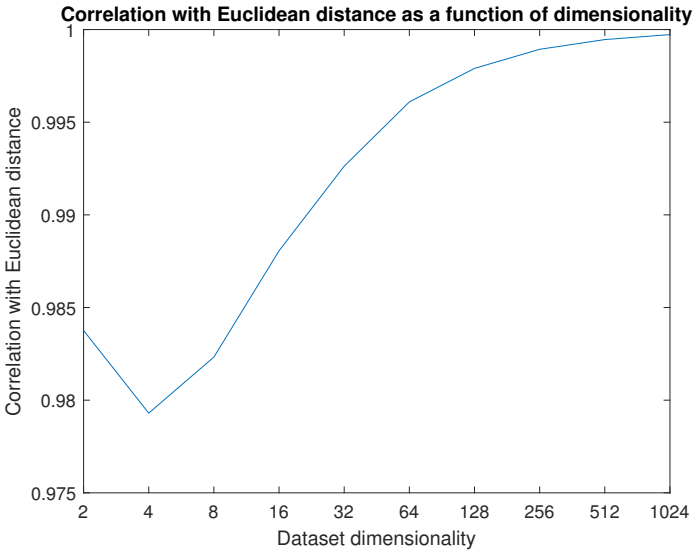

Figure 1.3: Correlation of Euclidean and Manhattan distances as a function of dataset dimensionality.

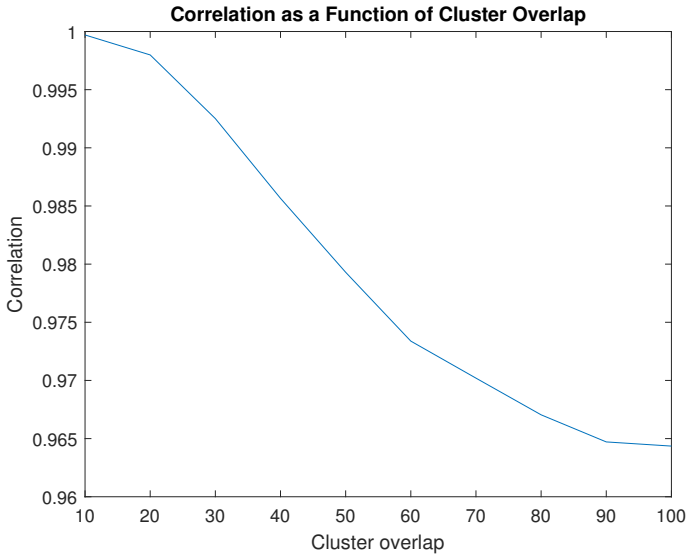

Figure 1.4: Correlation of Euclidean and Manhattan distances as a function of cluster overlaps in datasets.

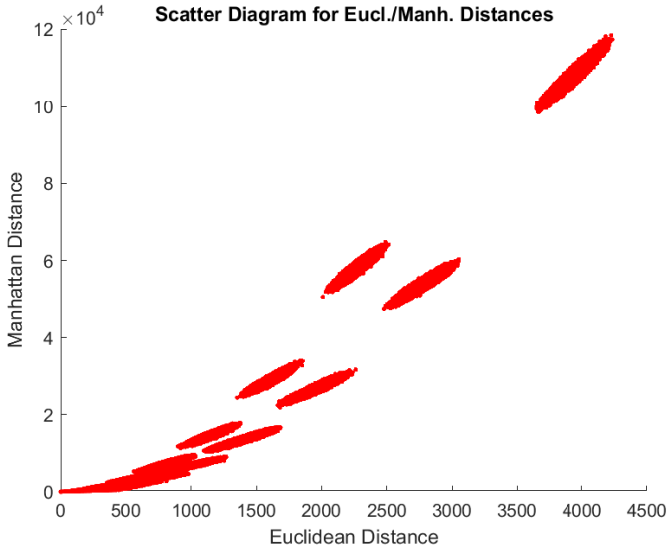

Figure 1.5: Scatter diagram of Euclidean vs. Manhattan distances.

1.4 Conclusions

Our experiments show that Manhattan distance can be used as a heuristic for Euclidean distance in terms of efficiency and accuracy. As future work we suggest trying clustering with this heuristic and seeing how much results differ in comparison with using the Euclidean distance.

References

[1] Dataset page in Machine Learning Unit in University of Eastern Finland. http://cs.uef.fi/sipu/datasets/

Chapter 2

Complexity of Infinity Norm

2.1 Introduction

In Figure 2.1 we show how some distance measures are calculated in 2 dimensions. In this article we derive time complexities of the infinity norm. Infinity norm is calculated in d dimensions as

$$d_\infty = \max(\mid x_{11} - x_{21} \mid, \mid x_{12} - x_{22} \mid, ..., \mid x_{1d} - x_{2d} \mid), \tag{2.1}$$

where the first subindex denotes the point number and the second subindex denotes the feature number. As an algorithm we can write it

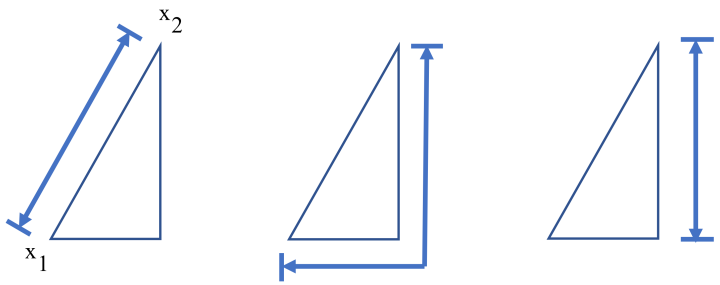

Figure 2.1: Calculation of Different Distance Measures. Euclidean distance (left), Manhattan distance (middle) and Infinity norm (right)

```
max_j = 0                    // 1 clock cycle
for j = 1 to d
    d_j = abs(x_1j − x_2j)   // 2 clock cycles
    if d_j > max_j           // 1 clock cycle
        max_j = d_j          // 1 clock cycle
    endif
endfor
output max_j
```

Algorithm 1: Calculation of Infinite norm

2.2 Time Complexities

Following the clock cycle counts in Algorithm 1 we calculate several time complexities. The worst-case time complexity, when the if-block needs to be executed in every for-loop iteration, is

$$T_w = 1 + d \cdot (2 + 1 + 1) = 4d + 1. \tag{2.2}$$

The best-case time complexity, when the if-block needs to be executed only once, is

$$T_b = 1 + d \cdot (2 + 1) + 1 = 3d + 2. \tag{2.3}$$

Applying the "several points" case result from [1], we infer that the average number of visits to the if-block is

$$1 + \frac{1}{2} + \frac{1}{3} + ... + \frac{1}{d} \tag{2.4}$$

when the dataset is random in a hypercube-shaped space, and applying a modified example from [2], p. 531 we infer that

$$1 + \int_2^{d+1} \frac{dx}{x} < 1 + \frac{1}{2} + \frac{1}{3} + ... + \frac{1}{d} < 1 + \int_1^d \frac{dx}{x} \tag{2.5}$$

We now write the average time complexity:

$$T_a = 1 + d \cdot (2 + 1) + (1 + \frac{1}{2} + \frac{1}{3} + ... + \frac{1}{d}) \tag{2.6}$$

and a bound for it

$$1 + 3d + 1 + \int_2^{d+1} \frac{dx}{x} < T_a < 1 + 3d + 1 + \int_1^d \frac{dx}{x}, \tag{2.7}$$

that is

$$3d + 2 + \ln \frac{d+1}{2} < T_a < 3d + 2 + \ln d. \tag{2.8}$$

2.3 Conclusions

We derived time complexities for the infinite norm.

References

[1] M. I. Malinen, "Means of Several Random Variables", in Collected Papers 2021 - A Special Issue on Statistics and Probability, Mikko I. Malinen, 2022

[2] R. A. Adams, Calculus: A Complete Course, 4th ed., Addison Wesley, 1999

Chapter 3

Determining Number of Clusters From Pairwise Distances

3.1 Introduction

Determining the number of clusters is an important operation in clustering. There are several classical methods to determine the number of clusters, often based on validity indices [1]. In this article we present a modern method, in which pairwise distances are taken and a deep neural network is used to determine the number of clusters. In Figure 3.1 we see a dataset with highly overlapping clusters.

3.2 Proposed Method

In this method pairwise distances are taken, and a histogram is created from them to obtain a 1000-length array. The contents of the array are scaled so that the maximum is aleways the same. The array is fed into a neural network to obtain the number of clusters. Data for training and testing of the network: 18020 datasets consisting of variably but highly overlapping Gaussian clusters. n=100..1000 (number of points), k=1..10 (number of clusters), d=2..3 (dimensionality), n_i = random size (sizes of individual clusters). We divide the datasets into training set (14420) and test set (3600). The neural network consisted of the following layers:

Input layer(size 1000) followed by fully connected layers of sizes 1000, 1000, 800, 500, 300, 100, 10, followed by softmax layer(10) and classification layer(10).

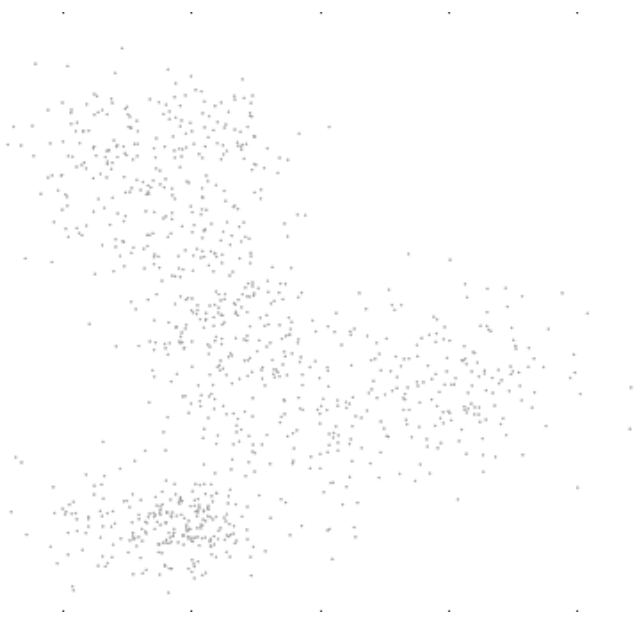

Figure 3.1: A 2-d dataset with overlapping 8 clusters

3.3 Results

Percentage of times the number of clusters was determined correctly:

GAP	28%
Proposed	24%
Calinski-Harabasz	16%
Davies-Bouldin	14%
Silhouette	13%
Randomly guessed	10%

3.4 Conclusion and Future Work

Works quite at the level of best classical methods. Later increase the diversity of datasets.

References

[1] G. Gan, C. Ma and J. Wu, "Data Clustering Theory, Algorithms, and Applications, SIAM, 2007

Chapter 4

NP-complete Problem Solving Needs Exponential Amount of Neurons

4.1 Introduction

In this article we make an assumption that P≠NP. Then NP-complete problems are exponential time. In neural networks, neurons are arranged into layers. In fully connected networks all neurons in a layer affect all neurons in the next layer.

4.2 Calculation

For two consecutive layers consisting of a and c neurons there are $a \cdot c$ operations consisting of multiplications with weights w_{ij}, $(a-1) \cdot c$ additions, s bias b_j operations and in the whole network the time taken by activation function calculations is us where u is constant, $u > 0$ and s is the number of neurons. The total number of operations in a network is in maximum, when the layers are fully connected and the intermediate layers are two equal-sized layers. If the number of neurons is such that the intermediate layers cannot be equal-sized, the bigger layer (size is one bigger) is put next to the input layer.
In Figure 4.1 we show a neural network with 7 neurons and 2 intermediate layers. In Figure 4.2 we show the intermediate layers in a case where they can be equal-sized. The number of operations in the whole network is then

$$op = N \cdot \frac{s-1}{2} + (N-1) \cdot \frac{s-1}{2} + \frac{s-1}{2}$$

$$+ \frac{s-1}{2} \cdot \frac{s-1}{2} + (\frac{s-1}{2} - 1) \cdot \frac{s-1}{2} + \frac{s-1}{2}$$

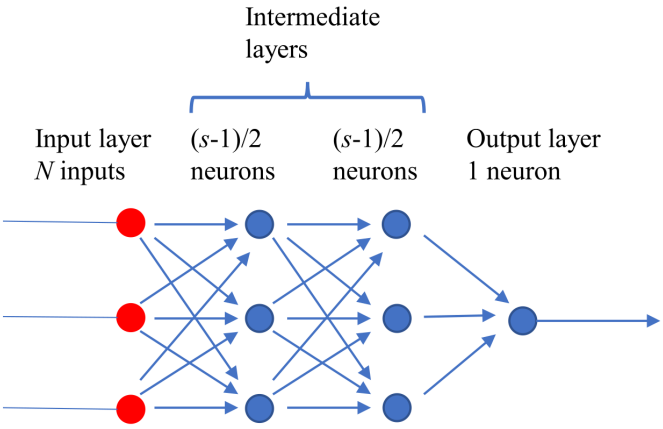

Figure 4.1: A Neural Network

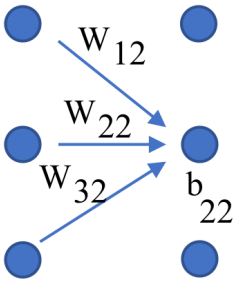

Figure 4.2: Calculation operations between two layers.

$$+\frac{s-1}{2}\cdot 1+(\frac{s-1}{2}-1)\cdot 1+1+us$$
$$=(N+1)\cdot(s-1)+\frac{s^2+1}{2}+(u-1)s$$

This is polynomial, but exponential, if

$$s=\alpha^{\beta N/2}=\alpha^{\gamma N},$$

where α, β and γ are constants and $\alpha,\beta,\gamma>0$.
When the number of neurons is such that the intermediate layers cannot be equal-sized, the number of neurons in the first intermediate layer is $(s-1)/2+1/2$ and the number of neurons in the second intermediate layer is $(s-1)/2-1/2$, and the total number of operations in the network is

$$op=N\cdot\frac{s}{2}+(N-1)\cdot\frac{s}{2}+\frac{s}{2}$$
$$+\frac{s}{2}\cdot(\frac{s}{2}-1)+(\frac{s}{2}-1)(\frac{s}{2}-1)+\frac{s}{2}-1$$
$$+(\frac{s}{2}-1)\cdot 1+(\frac{s}{2}-2)\cdot 1+1+us$$
$$=N\cdot s+\frac{s^2}{2}+us-2.$$

This is polynomial, but exponential, if

$$s=\alpha^{\beta N/2}=\alpha^{\gamma N},$$

where α, β and γ are constants and $\alpha,\beta,\gamma>0$.

4.3 Conclusions

We have shown that solving NP-complete problems by neural networks needs at least exponential amount of neurons.

Chapter 5

A Bad Start Sometimes Leads to a Good End: That Is How It Is In Random Swap

5.1 Introduction

Random Swap [1] is a variant of *k-Means* clustering algorithm. It finds a global or a good quality local minimum of *MSE* if executed long enough. In this article we show by experiments, that for the Bridge dataset, a bad first second of run the end result will be better than the end result from a good start.

5.2 Experimental Results

The first experiment is like this: Run three (or some other amount) Random Swaps for one second. In an experiment, the order of 3 curves in terms of MSE stays the same until 100 seconds and the order is reversed after 10000 seconds. The runs were made up to 18000 seconds, see Figure 5.1.

In the next experiment we make 100 runs (curves) coloring the bad start curves blue and good start curves red. After 10000 seconds (10^4 seconds) the order of the colours has been mostly changed to reversed, see Figure 5.2. This set of parallel runs took time 125h on a PC with 4 processor cores.

In Figure 5.3 the tails of curves from Figure 5.2 have been magnified. The order can be seen more clearly.

To visualize even better the ranks of curves in the start and in the end, we show a scatter diagram, see Figure 5.4. The linear regression line has slope -0,24 (negative slope) and it is statistically significant ($\alpha = 0.05$) if the linear

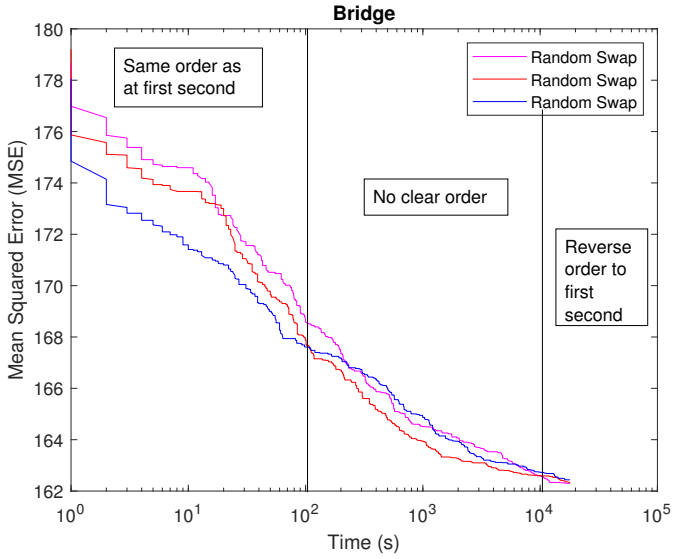

Figure 5.1: Runs for Bridge Dataset.

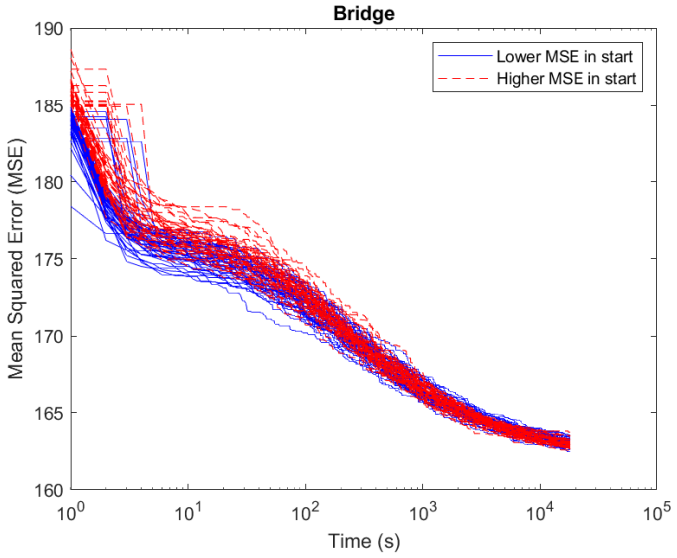

Figure 5.2: 100 Runs for Bridge Dataset.

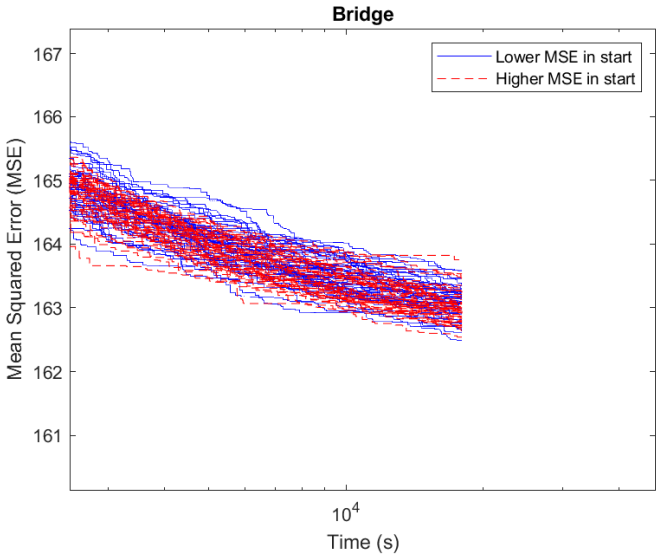

Figure 5.3: 100 Ends of Runs for Bridge Dataset.

regression assumptions hold.

5.3 Summary and Conclusions

We run Random Swap several times. If the start was bad, the more pbobably the end result was better than with those runs with good start. This phenomenon works with dataset bridge. We tried also with dataset house, for it this phenomenon was not present.

References

[1] P. Fränti, "Efficiency of random swap clustering", Journal of Big Data, 5:13, 1-29, 2018.

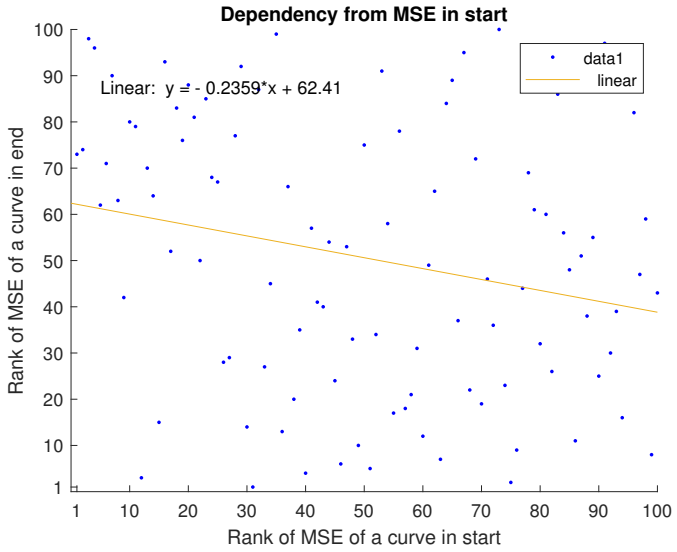

Figure 5.4: First MSE and End MSE Scatter With Regression.