Feature extraction in evolved/optimized spiking neural networks

Na Young Ahn, Daniel Boschmann, Shachar Dror, Jonas Reiher, Melina Reiter, Giuliana Tamassia

&

Cristian Jimenez-Romero

Short Description
In this report we cover the analysis work done with the results of a genetic evolution algorithm. The algorithm is used to optimize the architecture of a spiking neural network (SNN) tasked with a navigation problem. Our analysis contains several methods of data explorations: Topological Network Analysis, Regressions, and Principal Component Analysis (PCA). Our works point out some key feature to be implemented in order to achieve better preforming SNN.

1. Introduction

1.1. Motivation

While working with modern algorithms like a genetic algorithms or a SNN we have no insight in what they are doing. If the genetic algorithm converges to a local or even global optimum, we would not understand why the specific solution defines an optimum.
If we understand the SNN and what makes a good individual better than the others, we might be able to reuse parts of the SNN in bigger problems to start with better examples for further genetic optimizations. This shortens the optimization process and leads quicker to good individuals.

1.2. The basic challenge

Given a comparably small SNN with only three layers and a total of 14 nodes, we want to understand how the connections influence the performance of every individual of the population in the genetic algorithm. Because the relation between the connections and the performance might take any form from linear over, polynomial of high order, exponential or entirely different we have to first reduce the dimensions of the problem to make it more approachable for other analysis methods while losing as little information as possible or find different ways to augment the data.
Therefore we not only have to analyse the structure of the network, but also detect similarities between good performing individuals and differences between good and bad performing individuals and build an algorithm that gives a prediction for the performance.
Last we want to share our suggestions on how to adjust the existing procedure to further enhance the prediction.

2. General setting

In the regarded problem setting, the food collection behavior of a virtual ant, the agent, in a maze is simulated. This maze, depicted in Fig. 1, is pixel-based and features three possible block states. White pixels indicate walls and red pixels are other non-passable obstacles. The agent (yellow ant) scavenges for food, indicated by green pixels, while trying to avoid collisions.

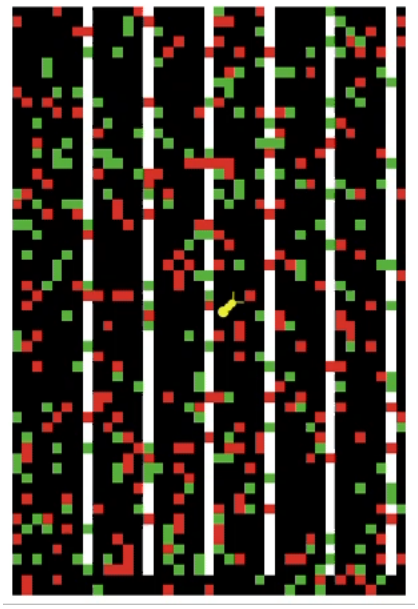


Figure 1: The simulation environment depicting a maze the ant needs to navigate. White pixels are walls, red pixels are other obstacles and green pixels are food.

2.1. Agent behavior

In order to fulfill this task and navigate the maze, the agent's decision process is modelled by a spiking neural network (SNN), as depicted in Fig. 2. This network consists of three layers with a total of 14 neurons or nodes. The connections between each two neurons are weighted and incorporate a transmission delay. For each time step, the neurons are aggregating the impulses incoming to them and then produce an outgoing signal according to a nonlinear and time-dependent activation.
The network's first layer consists of six neurons for perception, numbered from 11 to 16, that provide stimulation to the second layer based on the ant's sensory input. Neurons 11, 12 and 13 are indicating the colors white, red and green in the ant's line of sight. 14 and 15 get activated if an obstacle is hit or when food is collected and neuron 16 acts as a heartbeat, giving off a pulse every 5 time steps. This layer is fully connected to all neurons of the second layer. This second layer acts as the agent's computational unit or brain. All neurons here, numbered from 20 to 25, are recurrently connected in an all-to-all fashion with the exception of self-loops. Additionally, all of them are then connected to layer three. The neurons 30 and 31 the third layer serve as the activation units for the agent's actions. Outgoing signals of neuron 30 lead to a clockwise rotation of the ant while the activation of neuron 31 results in a forward movement. Over all layers the network then consists of 78 connections, each with a weight and a delay, resulting in 156 parameters in total.
The ant's performance in navigating the maze is quantitatively measured by a fitness function. Here, the fitness F for a single simulation run depends on the number of food units collected f and the number of collisions c:

$$F = f - b \cdot c, \tag{1}$$

where b is a bias parameter, weighting collisions against food collection with a factor of $b = 1.5$ in the default setting.
While the distribution of obstacles and food in the maze is initially randomly generated, it is fixed for every observed simulation run, as is the ant's starting position. It's navigation behavior is then deterministic for fixed weights and delays. The weights and delays resulting from the evolution are limited to certain ranges in the simulation. Weights are clipped to the continuous range $[-20.0, 20.0]$ while delays are whole numbers in $[1, 10]$.

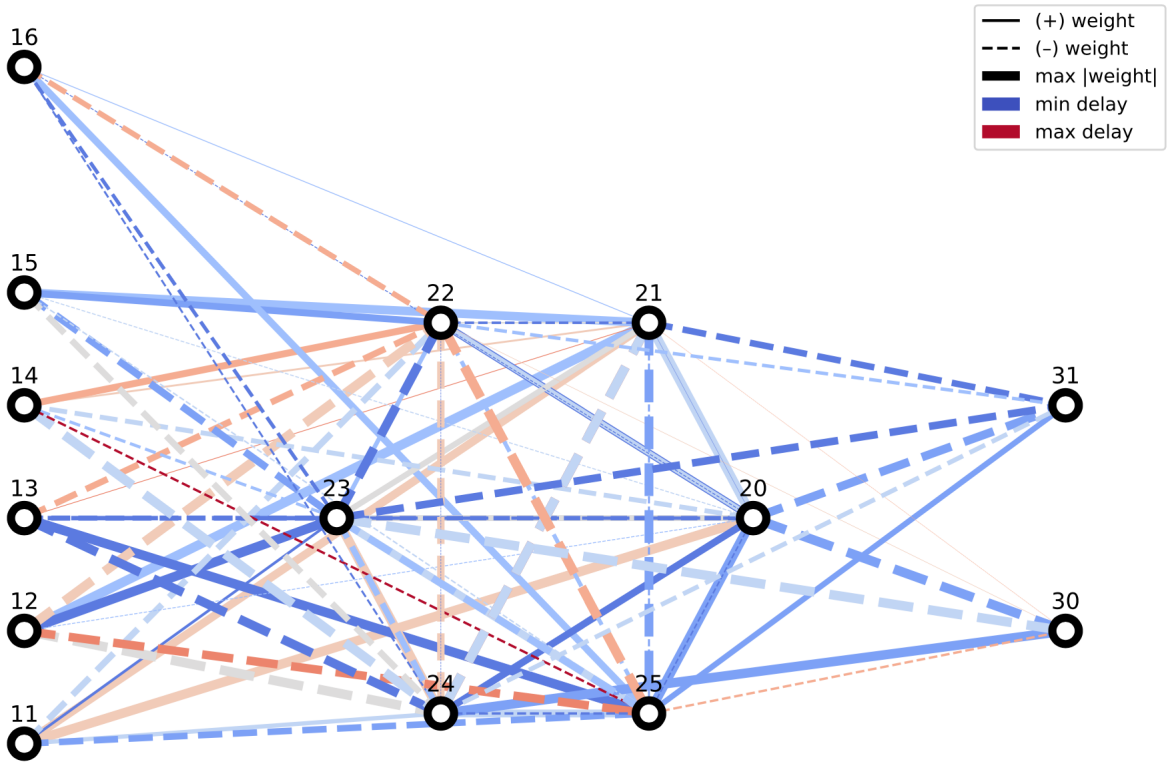


Figure 2: A spiking neural network modelling the ant's behavior from sensory inputs to actions. The line width indicates the connection weight with negative values depicted as dashed lines. The color coding from blue to red indicates growing delays.

2.2. Genetic Algorithm

To find an optimal parameter set for the Spiking Neural Network a so called Genetic Algorithm (GA) is used. It is based on Charles Darwin's theory of natural selection from biology. In general, a GA requires its solution to have a genetic representation. In the discussed example of an ant in a maze, one ant with its behavior and abilities represents one solution of the GA, as the ants properties do directly relate to the number of food it is able to collect and the number of times it collides with obstacles in the maze. This ant is called an individual. The data (weights and delays) of this ant are its chromosome. A second requirement to conduct a GA for optimization is a fitness function to evaluate the solution. Here, it is defined by Eq. (1). This fitness function thus measures the quality of a solution.

The GA usually consists of four steps, which are illustrated in Fig. 3. First, as an *initialization*, a random population of individuals is chosen. In a second *selection* step the fitness functions for all these individuals are evaluated, for example by running a simulation. The best performing individuals are kept and all others are discarded. In the third and fourth step a new generation is derived from these best performing individuals. The third step describes a crossover procedure by statistically recombining the existing individuals. So here, the chromosomes of two parent-ants, respectively the weight-delay packages of two SNNs, are combined to generate a child-ant with similar attributes. For each child, a new pair of parents is selected. In the fourth step, the new generation of child-ants is undergoing some random mutations. Thus some genes (some weights or delays) of a child's chromosome (weight-delay package) are altered. This generation of child-ants do now serve as the next parent generation, as the algorithm loops between the steps two, three and four.

Genetic Algorithms can deal with complex problems and are comparably fast, as the search space can be explored from different starting populations or different breeds in a parallel manner. On the other hand, a detection of multiple local maxima for the fitness values is likely, as the GA might be sensitive to the chosen initial conditions. Different starting populations may therefore converge to a

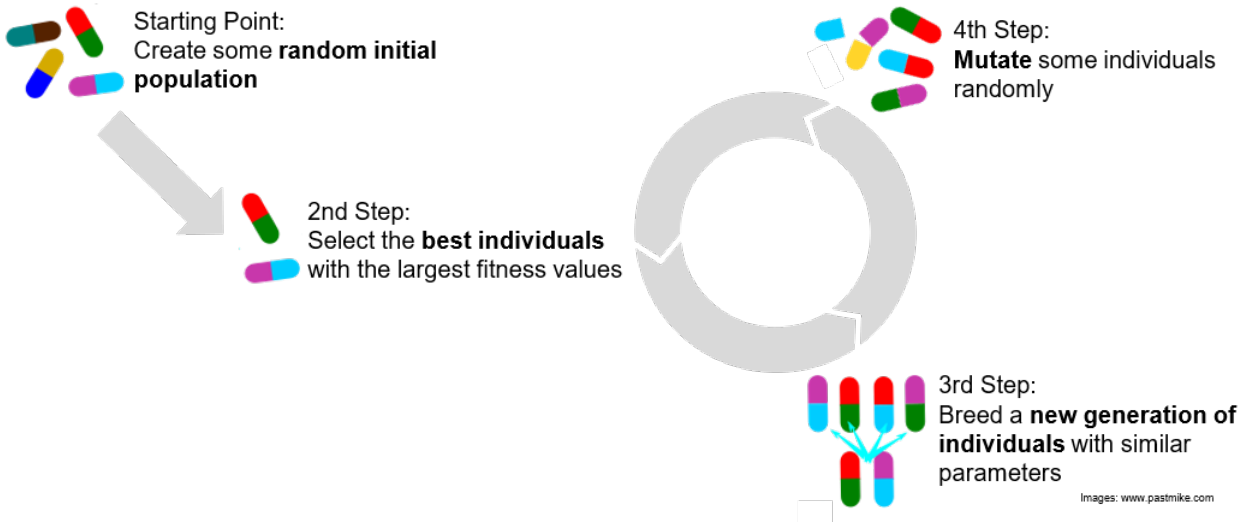


Figure 3: Four steps of a Genetic Algorithm.

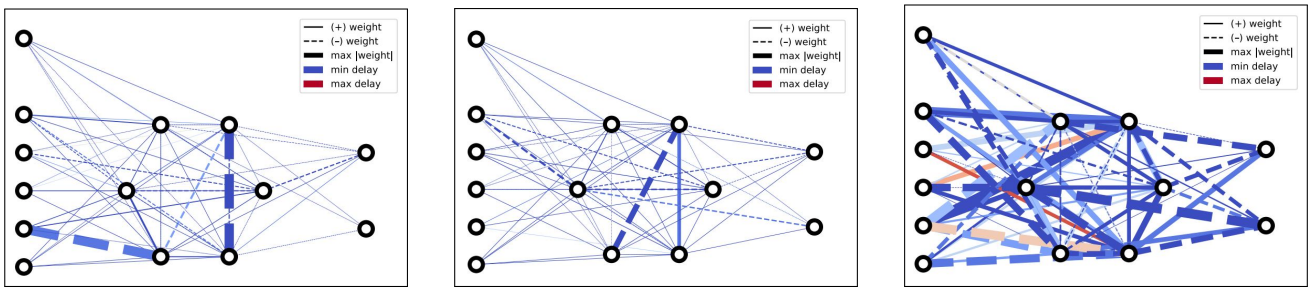
different solution. [1]

2.3. Data

Multiple datasets were provided on which to perform the further analysis and model building with the goal of predicting fitness values from the parameter values alone. Six of these datasets consists of a population of 96 individuals each, taken from the 99th generation of one evolutionary trajectory. A seventh and eighth dataset both contain 96 individuals out of every generation from 0 to 99 for one trajectory. In total, this adds up to 19776 individuals. For each individual in these populations the values of every connection's weight and delay are specified, as well as the fitness values achieved in simulation run.
These data points are provided in the form of `.csv`-files with 157 comma-separated values in three rows. The first row holds 78 weight values for every connection, the second row lists the corresponding delays and the third row contains the individual agent's fitness value.

2.4. Randomness in data

After analysing the data of the first few datasets we noticed the set itself do not differentiate much, while performing drastically different. Comparing different datasets with each other the distribution of the weights and delays differs strongly, while the individuals perform almost the same in average .



(a) best in dataset one vs 2nd best in dataset one

(b) best in dataset one vs worst in dataset one

(c) best in dataset one vs best in dataset two

Figure 4: Differences between individuals

In Figure 4 we can see the difference in connection values, while wide or dark (dark blue or dark red) connections imply a strong difference. To analyse the difference of two datasets we calculate the mean of all weights and delays of each dataset. Those means define a center for each set that lets us calculate the distance between two datasets. To confirm the distances of the centers shows us two clusters, we also calculate the size of the standard deviation in the same metric.
With that we recognise that while two individual networks might differ in the way the weights and delays are distributed, the resulting algorithm might work the same.

3. Pre-processing

3.1. Permutation of second layer

The nodes in Layer two of our network are supposed to work the same, therefore there would be equivalence classes of networks that only differ by permutation of the 2nd layer of the network. With the configurations of our neural network, the networks in Fig 5 would be working the same, while the connections would show no similarities. This is because both networks are in the same equivalence class.

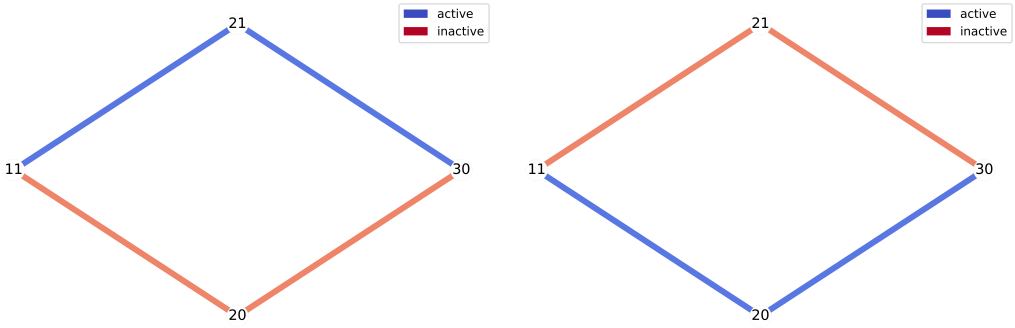


Figure 5: Same networks

One way to standardize the network and make actual similarities visible, is to permute the nodes in Layer two. because we need a way to sort all individual networks we permute all individuals such that the first six connections of the corresponding network would be sorted in increasing order. All other connections would be permuted accordingly.
To permute the data we build adjacency matrices for weights A^W and delays A^D separately. The permutation matrix P was build such that the weights $A^W_{1,7-12}$ would be sorted in increasing order. The permutation is executed by

$$P^T A^W P = A^W_{perm} \text{ and } P^T A^D P = A^D_{perm}.$$

After permutation of our data, data files that describe the same network have also the same layout of variables.
In Figure 6 the permutation process is illustrated, the heatmap represents the adjecency matrix before (left) and after (right) the permutation.

3.2. Problem between assumption and simulation

While the assumption was that the nodes in layer two are identical, the simulation shows that even individuals in the same equivalence class perform quite different when just tested on one maze with one random seed (a random but fixed order the nodes are processed in). This means that an individual and the permuted counterpart define the same network, but small priority changes of the nodes lets them behave different in the simulation.
After discovering this flaw in the simulation we tested other assumptions of the network. While the weights are supposed to be in range of $[-20, 20]$ and the delays in range of $[1, 10]$ the mutation of

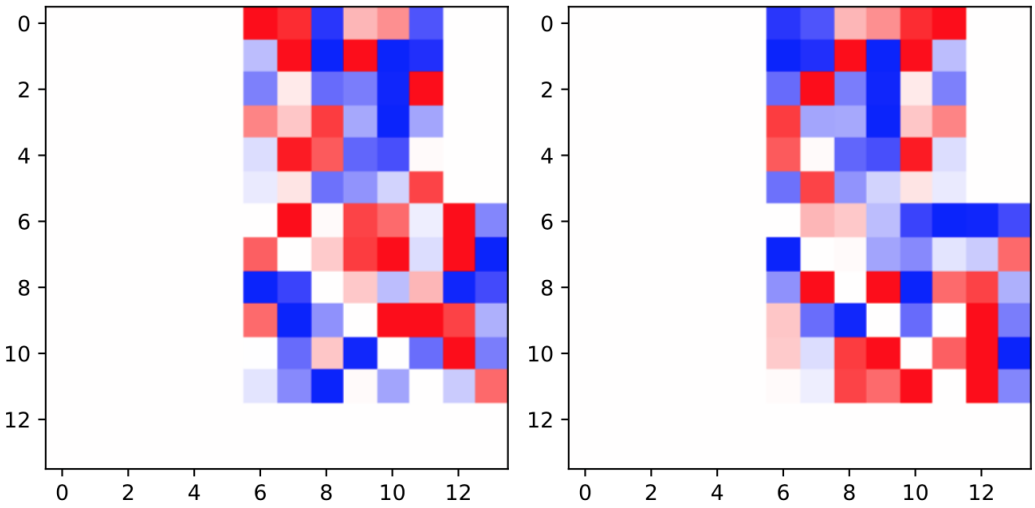


Figure 6: Heatmap of the permutation process

the genetic algorithm gives values of -656 to 1394 for weights and -500 to 1951 for delays. The first assumption was that the algorithm would clip the values to -20 or 20 and 1 or 10 for weights and delays respectively, but further simulations showed again a difference between the individuals with and without clipped values.

4. Network Analysis

As discussed in Sec. 3.1, all the neurons in layer two can in theory be permuted without changing an individual ant's behavior, as they do not have a fixed task. However, this does not apply to layer one and layer three. The input layer is directly related to the corresponding sensors feeding signals into it. For layer three, both nodes represent a fixed action the ant is about to perform, as described in Sec. 2.1.
What can not be permuted, however, are the effective connections each input from layer has to each output neuron in layer three. This can be looked at to further the understanding of the underlying black-box SNN regardless of the exact connections in layer two. For this, all paths from one node in layer one to one node in layer three should be taken into account. An evaluation of which input has which influence on the output can then be conducted.

4.1. Evaluating connection strengths

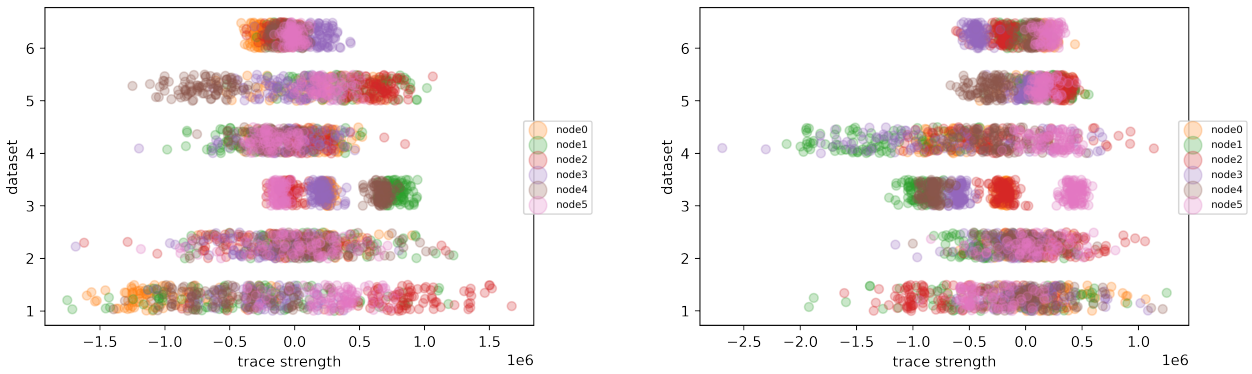
For the network topology regarded in this project, there are infinitely many paths between a given input and output node. Since in layer two every node is connected to every other node, an input signal can make an arbitrary number of loops before leading to an output signal. To nevertheless estimate the combined strength of all paths, a maximum number of transmissions in layer two needs to be set. Here, all paths with up to three bounces between nodes in layer two were considered. With increasing delays and temporally decreasing activation levels in individual neurons, this is found to depict the network transmission dynamics reasonably well.
The activation of individual neurons follows a more complex nonlinear dynamic, but a combination of weights and delays along a path can still be used to estimate it's importance to the transmission from start to end. Connections with large weights and shorter delays should be more relevant. For a single path it is therefore proposed to look at the multiplication of all it's weights over the sum of delays. Signals over multiple paths to the same output neuron can approximately be summarized.
For approximately computing the transmission strength from a given start node in layer one to a given end node in layer three, the following procedure is proposed to be performed for every individual:

- Find all paths i from start node to end node with at most three inner connections in layer two.

- For each path i, compute it's approximative strenght s_i from weights $w_{i,j}$ and delays $d_{i,j}$ along the path as $s_i = \prod_j w_{i,j} / \sum_j d_{i,j}$.
- Compute the total connection strength S over all regarded paths as $S = \sum_i s_i$.

4.2. Connection strength results

In the following Fig. 7 the distribution of every input node's connection strengths to an output layer is depicted for individuals of every dataset. Fig. 7a shows these values for connections to the rotation inducing node and Fig. 7b for the node responsible for forward movement.



(a) Output node 30

(b) Output node 31

Figure 7: Connection strengths from all input nodes to both output nodes for samples from different datasets.

It becomes apparent that for a single dataset the connections from the same starting node usually form clusters, meaning that the same input nodes are responsible for outgoing actions across individuals. However, this holds only for the single evolutionary trajectories considered within each dataset. Across datasets different input nodes show the largest connection strengths to each output.
This observation suggests that the ant can achieve behavior suitable for the maze navigation regardless of which input sensor it takes into account primarily. For example, moving forward towards green patches and turning otherwise is mostly equivalent behavior to turning away from red and white patches and moving forward otherwise. Each dataset, and therefore each evolutionary trajectory, finds a different way to locally optimal behavior, though.
One property that can be observed across datasets, however, is that nodes with positive trace strengths towards output node 30 tend to show negative trace strengths towards node 31 and the other way around. This basically shows that both actions are chosen as opposites to one another and a sensory input speaking for one of the actions speaks against the other one.

5. Regression

To begin with, it was unclear to us what the nature of our problem is: linear or non-linear and do we have outliers/noise. Starting with the first dataset we preformed both linear and polynomial regression, that dataset had 192 samples, both with and without cross-validation. Same analysis was done for the joined first and second data. For the full complete dataset of all 3 combined we have conducted also RANSAC regression to gain information about the nature of any outliars. To complete our analysis we have conducted statistical significant analysis for the best performing method - linear regression, and have reduced our variable set from 156 to 54 based on the P-values scores that were lower then 0.1 for each variable. We then created reduced linear regression with those variables and received similar performance without cross validation and improved performance for the cross-validated model.

6. Principal Component Analysis (PCA)

Using the reduced dataset, retrieved from the regression step, that only keeps the weights and delays of high p-values, we further reduce the dimensionality while preserving the information of the data in terms of variance by applying Principal Component Analysis (PCA).

6.1. Mathematical description

Principal Component Analysis (PCA) is a structure detection method of multivariate statistics. This means that the analysis tries to extract the structure from the data. Thus the method serves to simplify and structure datasets. This reduces to the variables that capture the bulk of the scattering. We are looking for as few linear combinations as possible of the observed variables to approximate the data. These are referred to as the principal components.

Let a bivariate variable X consisting of two feature vectors (X_1, X_2) tested on n samples $\begin{pmatrix} x_{11} & x_{12} \\ \vdots & \vdots \\ x_{n1} & x_{n2} \end{pmatrix}$.

Thus the individual samples are given by $x_i = (x_{i1}, x_{i2})$ with $i = 1, ..., n$.
A projection of a point x on a vector $e^T = (e_1, e_2)$ of length one is given by:

$$(xe)e^T = ((x_1e_1 + x_2e_2)e_1, (x_1e_1 + x_2e_2)e_2).$$

In order to approximate the data as closely as possible, a projection direction is sought so that the distance between the projection line and the observations as a whole is as small as possible. We thus get the following problem:

$$\sum_{i=1}^{n}(x_i - (x_ie)e^T)(x_i - (x_ie)e^T)^T \stackrel{!}{=} \min_{e_1,e_2,e^Te=1}.$$

Shaping now the minimization problem with the help of the properties $e^Te = 1$ and $xe = e^Tx^T$ and the definition of the covariance matrix S_{XX}, we get:

$$\begin{aligned}
\sum_{i=1}^{n}(x_i - (x_ie)e^T)(x_i - (x_ie)e^T)^T &= \sum_{i=1}^{n}(x_i - (x_ie)e^T)(x_i^T - e(x_ie)^T) \\
&= \sum_{i=1}^{n}(x_ix_i^T - (x_ie)e^Tx_i^T - x_ie(x_ie)^T + (x_ie)e^Te(x_ie)^T) \\
&= \sum_{i=1}^{n}(x_ix_i^T - x_iee^Tx_i^T) \\
&= \sum_{i=1}^{n}x_ix_i^T - \sum_{i=1}^{n}e^T(x_i^Tx_i)e \\
&= (n-1)(tr(S_{XX}) - e^TS_{XX}e).
\end{aligned}$$

To minimize this expression, the variance of the transformed dataset $e^TS_{XX}e$ must be maximized. This is because the variances of the original dataset are fixed and therefore $tr(S_{XX})$ is also fixed. So it results in the following task for the principal component analysis:

A projection direction is sought, so that the values obtained by the projection x_ie have as much variance as possible:

$$s_Y^2 = e^TS_{XX}e \stackrel{!}{=} \max_{e,e^Te=1}.$$

To solve this problem, we use the Lagrange multiplier theorem.
With the empirical variances and covariances $S_{X_1}^2, S_{X_2}^2, S_{X_1X_2}$ of S_{XX} is obtained:

$$S_y^2 = (e_1, e_2) \begin{pmatrix} S_{X_1}^2 & S_{X_1X_2} \\ S_{X_2X_1} & S_{X_2}^2 \end{pmatrix} \begin{pmatrix} e_1 \\ e_2 \end{pmatrix} = e_1^2 S_{X_1}^2 + 2e_1e_2S_{X_1X_2} + e_2^2 S_{X_2}^2.$$

Then the Lagrange function is defined by:

$$f(e_1, e_2, \lambda) = e_1^2 S_{X_1}^2 + 2e_1e_2S_{X_1X_2} + e_2^2 S_{X_2}^2 + \lambda(1 - e_1^2 - e_2^2)$$

f is partially differentiable:

$$\frac{\partial f}{\partial e_1}(e_1, e_2, \lambda) = 2e_1 S_{X_1}^2 + 2e_2 S_{X_1X_2} - 2\lambda e_1, \tag{2}$$

$$\frac{\partial f}{\partial e_2}(e_1, e_2, \lambda) = 2e_1 S_{X_1X_2} + 2e_2 S_{X_2}^2 - 2\lambda e_2, \tag{3}$$

$$\frac{\partial f}{\partial \lambda}(e_1, e_2, \lambda) = (1 - e_1^2 - e_2^2) \tag{4}$$

Nulling of the derivations (2), (3) returns:

$$\begin{pmatrix} S_{X_1}^2 - \lambda & S_{X_1X_2} \\ S_{X_2X_1} & S_{X_2}^2 - \lambda \end{pmatrix} \begin{pmatrix} e_1 \\ e_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$\Leftrightarrow (S_{XX} - \lambda I)e = 0.$$

Since (4), $e_1^2 + e_2^2 = 1$ must apply, we can rule out the trivial solution $\begin{pmatrix} e_1 \\ e_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$. So the equation has a solution exactly when λ is selected so that the matrix $S_{XX} - \lambda I$ does not have full rank. Thus, λ just corresponds to the solutions of the characteristic equation $det(S_{XX} - \lambda I) = 0$, i. e. the eigenvalues of S_{XX}. So the solution of the problem is given by the eigenvectors $e_1 = (e_{11}, e_{12})^T$, $e_2 = (e_{21}, e_{22})^T$ of S_{XX}.

Let $Y = (y_1, y_2) = \begin{pmatrix} y_{11} & y_{12} \\ \vdots & \vdots \\ y_{n1} & y_{n2} \end{pmatrix}$ be the transformed dataset, where y_1 and y_2 are comparable to the variable Y_1 and Y_2. Y_1 and Y_2 is designated as the principal component. With $E = (e_1, e_2) = \begin{pmatrix} e_{11} & e_{12} \\ e_{21} & e_{22} \end{pmatrix}$ the equation applies:

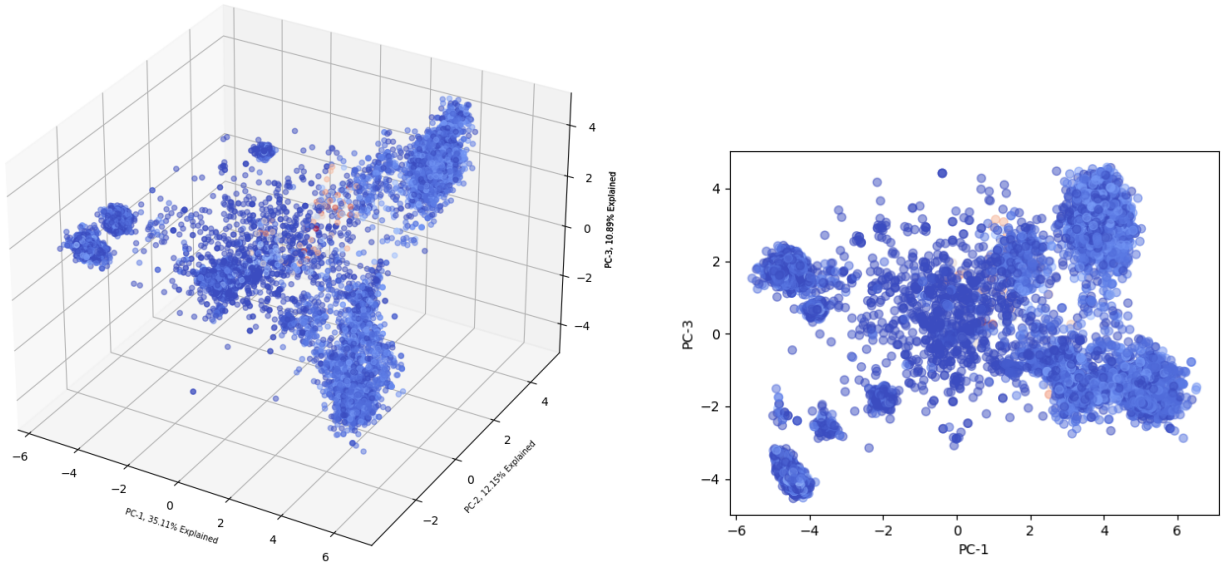
$$Y = XE.$$

We can use the PCA not only for two-dimensional datasets, but also for a p-dimensional dataset. If we work now with a p-dimensional dataset, a data matrix X is given. This consists of p feature vectors $(X_1, \ldots, X_p)$, which in turn contain the n tested samples $X = \begin{pmatrix} x_{1i} \\ \vdots \\ x_{ni} \end{pmatrix}$ with $i = 1, \ldots, p$. We are looking for vectors $e_j = \begin{pmatrix} e_{1i} \\ \vdots \\ e_{pj} \end{pmatrix}$ with $j = 1, \ldots, p$, so that the data can be approximated as well as possible by hyperplane $y_j = e_{1j}x_1 + \cdots + e_{pj}x_p$. In the same way as in the two-dimensional case, we get the equation $Y = XE$ again, with principal components $Y_1, \ldots, Y_p$ and eigenvector matrix $E = (e_1, \ldots, e_p) = \begin{pmatrix} e_{11} & \ldots & e_{1p} \\ \vdots & & \vdots \\ e_{p1} & \ldots & e_{pp} \end{pmatrix}$.

6.2. PCA implementation

We implemented the PCA on the reduced p-dimensional dataset, where $p = 54$. These are our reduced variables from the regression. Since PCA is largely influenced by scales, and our features

(weight, delay) have different scales, we first standardized the data to a scale of zero mean and unit variance.
Then we project the data into a three dimensional sub-space preserving essential parts of the data in terms of its variance. We plotted the sub-space that is defined by the first three principal components of the data combined with a color coded fitness, see Fig. 8.
For our dataset, we observed that the first principal component describes 35.14% , the second principal component 12.15% and the third principal component 10.89% of the total variation. Altough PCA is not preserving metrical properties we gain the possiabily to visualize our data and better explore it. Also with overall 60%,of varation described by first 3 PC we get a good estimation to our data behaviour.
To our understanding applying the PCA after reducing dimensions had positive, valuable effect on our analysis- with just PCA on the full data set we encountered results of all principal components describes around 10% of the data.



(a) Projection onto 3 main principal components (b) Projection onto 2 main principal components

Figure 8: All training data points projected to main principal component space

In Figure 8 we see all training data reduced into the three dimensional space determined by top 3 principal components. Every point is a training sample. The spectrum of blue to red color corresponds to low to high fitness value. A cluster of high fitness values can be observed in the center of the plot. We can also detect some other clusters of blue data points, with low fitness (near zero). To identify these clusters more precisely, we have applied k-means to the data in the next step.

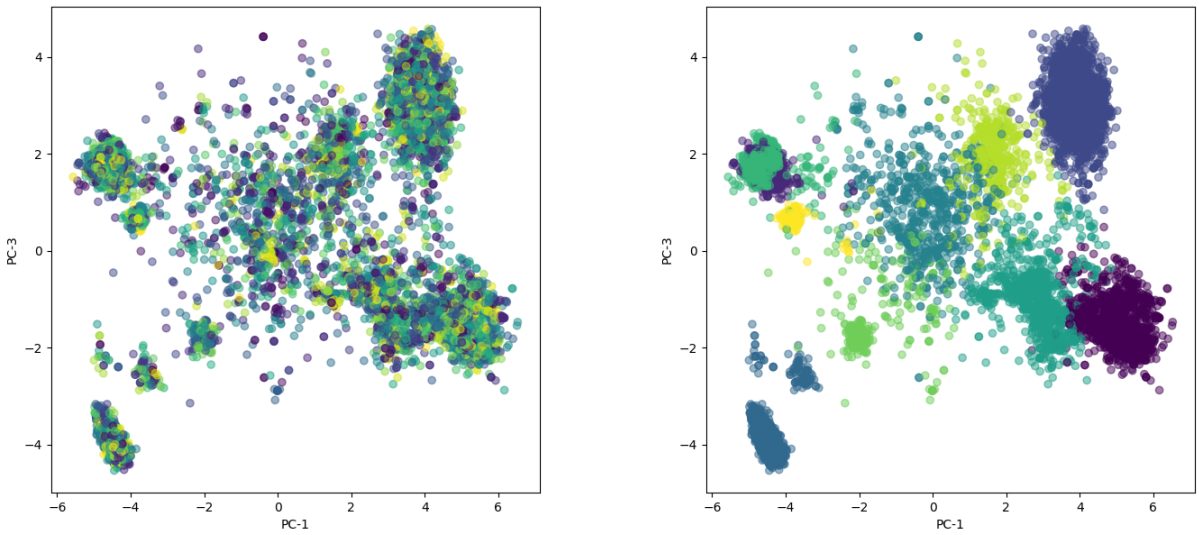
7. K-Means Clustering

K-Means clustering is a method by which a certain number of objects can be divided into homogeneous groups. The goal is to have different objects within a cluster as similar as possible after the classification. K-Means algorithm can be applied to multi-dimensional objects and approaches the respective cluster centers through repeated recalculations until no significant change occurs. The algorithm is guaranteed to converge after finite number of iterations. It has a local optimum and the final result depends on initialization.

7.1. K-Means implementation

To measure the fitness for a new individual, we first cluster the data with the k-means algorithm. Therefore we initially generated 2500 clusters (Fig. 9a). Each cluster contains a rather small amount

of individuals to be able to differentiate between good and bad samples. Hence we have also tried it with only 10 clusters (Fig 9*b*). In the Figure 9 each clusters were assigned the same color.



(a) 2500 clusters

(b) 10 clusters

Figure 9: Results from the k-mean algorithm with different numbers of clusters

Then each test data points are assigned to the cluster of the minimal euclidean distance. We predict the fitness of the test data points by giving the median or mean fitness value of the cluster it belongs. To account for the case when the test data point was nonetheless assigned a cluster and its median fitness value, despite relatively far minimal distance, we assign the categorical variable "invalid" for such case where the distance between assigned cluster centroid and the test data point was greater than the distance between assigned cluster centroid and the furthest data point within that cluster. For better measuring this validity threshold we our predictions against the samples in the validation set - 10% of the data which were not used to fit the PCA but were projected to it and measured.

8. Results

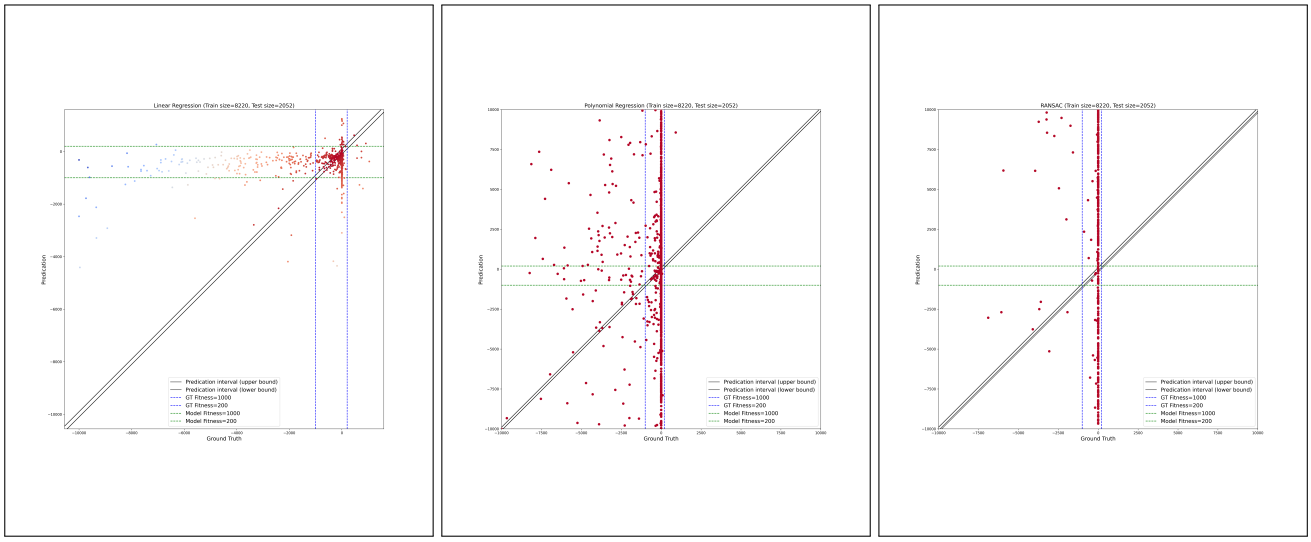
8.1. Augmentation of the dataset

While the augmentation of the dataset in form of clipping at the appropriate values and permute the order of nodes in layer two should not result in changes to the behaviour of an individual, the simulation shows quite different results. This leads to the conclusion that the performance value should not rely on the simulation for just one maze and random seed. Our suggestion would be to get the average of as much as possible (depends on calculation time) fitness values from individuals of the same equivalence class for different mazes and random seed.
To take just one fitness value leads to too much randomness in evaluating whether or not a individual would perform overall good or bad.

8.2. Regression

Our analysis of the different regressions is shown in Figure 10. Best results our achieved using linear regression- with R2 score of 0.072. Others have reached high negative scores (in thousands) and are seem to have collapsed to an almost constant prediction of fitness 0. Although scoring is low we do recognize high accuracy for high fitness prediction, probably due to the built in bias in

the original data (more samples of well fitted individuals, only few evolutionary trajectories). We also deduce from this behaviour that our problem is probably linear and have no outliers.



(a) Linear regression (b) Polynomial regression (c) RANSAC regression

Figure 10: Regression Analysis

Our statistical significant analysis over the linear regression had led to 54 statistically significant variable, out of which 41 are weight and 13 are delays with only two weight-delay coupling, test results are shown in table 1.

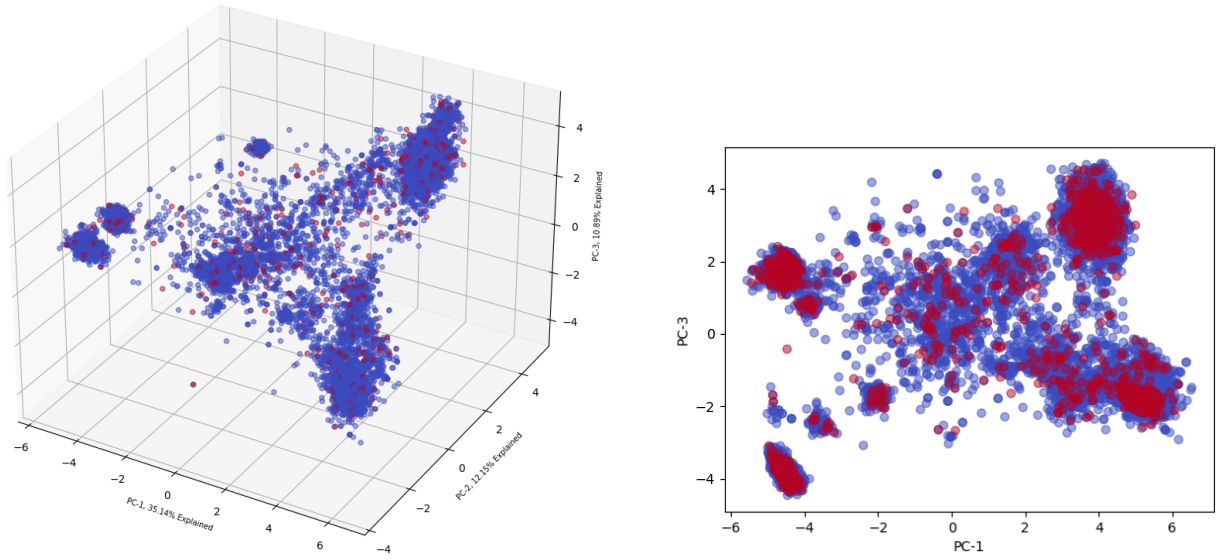
Figure 11: Regression Analysis

	coef	std err	t	P Value
const	-3.770.563	12.864	-29.312	0.000
w6	-939.606	24.979	-3.762	0.000
w8	1.080.863	22.687	4.764	0.000
w9	-1.334.574	23.659	-5.641	0.000
w12	483.064	17.544	2.753	0.006
w13	-593.894	16.195	-3.667	0.000
w16	425.261	21.374	1.990	0.047
w23	-1.538.064	36.181	-4.251	0.000
w27	-934.566	21.965	-4.255	0.000
w29	-569.744	18.236	-3.124	0.002
w30	552.090	21.927	2.518	0.012
w34	-797.247	19.470	-4.095	0.000
w43	-908.044	23.988	-3.785	0.000
w54	1.249.656	30.979	4.034	0.000
w58	-728.775	19.255	-3.785	0.000
w63	788.864	22.281	3.540	0.000
w65	-632.860	21.811	-2.902	0.004
w67	-1.115.980	26.036	-4.286	0.000
w68	-1.165.841	21.105	-5.524	0.000
w71	-917.661	34.055	-2.695	0.007
w72	-887.359	15.903	-5.580	0.000
w74	-887.556	24.965	-3.555	0.000
w75	1.068.530	27.005	3.957	0.000
d1	2.405.851	61.958	3.883	0.000
d8	-7.503.954	121.030	-6.200	0.000
d9	12.005.058	253.116	4.743	0.000
d14	-4.703.215	121.741	-3.863	0.000
d31	5.964.866	85.258	6.996	0.000
d57	8.846.862	89.259	9.911	0.000
d67	-8.666.020	165.700	-5.230	0.000
d68	-2.870.689	51.230	-5.604	0.000
d69	-7.047.909	161.576	-4.362	0.000

Figure 12: Linear regression

8.3. Prediction of fitness with PCA and K-Means clustering model

Relying on the improved explained variances of the first three principal components obtained by the significant feature extracted total dataset, we have created a three dimensional principal component space with datasets used to create the model scattered in blue. Then we tested for three different test datasets, which was projected onto the created principal component space in red.



(a) Projection onto 3 main principal components (b) Projection onto 2 main principal components

Figure 13: Test data points i.e. 10% of total data projected to main principal component space

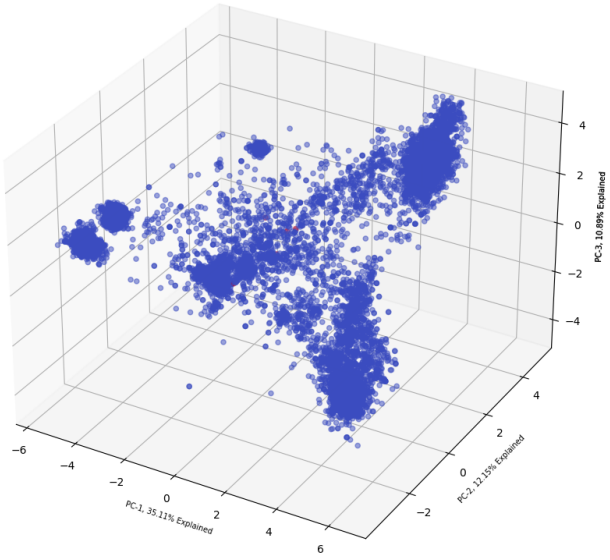
The first test dataset and setting was obtained by splitting all provided data into 90% train set and 10% test set. After creating the principal component space with the training set, we scale and transform our test set into the principal component space. In the 3d space as in Figure 13, we observe a sparsely distributed test data points i.e. red dots amongst several distinct clusters of the training data i.e. blue dots. However by looking at the 2d space comprised of principal components 1 and 3, we discover that the new data is not only clustered in the center where train data with high fitness value was congregated, but also other clusters where the train data did not display high fitness but was clustered together. With this observation in mind, we proceeded with the prediction.

cluster fitness value assignment	median		mean	
cluster size (k)	10	2500	10	2500
accuracy	0.629	0.563	0.002	0.054
total fitness data	1978	1978	1978	1978
classified "invalid"	1	498	1	468
classified "valid" and "good"	1245	833	4	82

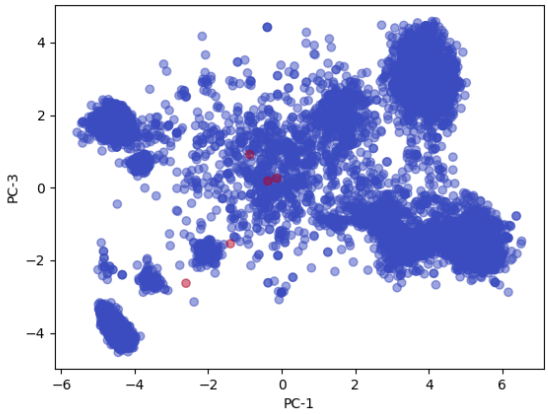
Table 1: Comparison of prediction by cluster fitness value assignment method on train_test_split dataset

Table 1 summarizes our evaluation structure. After assigning a predicted fitness value according to the closest cluster and identifying validity of that fitness value assignment by checking the distance threshold per test data point, we compared the predicted fitness value to the actual fitness value that was originally provided and calculated the accuracy of our prediction. To give depth to the simple accuracy measure, we also counted the number of "invalid" cases, and gave a tolerance of ± 10 to the fitness value to determine a valid and good classification.
To get a better understanding we tested for various parameters by changing the number of clusters and changing the method of cluster fitness value assignment. While small improvement in accuracy was observed with a lower cluster size, we could see a significant impact on the accuracy according

to how the fitness of each cluster was assigned: using mean or median. This gives us some idea into how fitness values are distributed and how dominant the outliers are even within clusters.

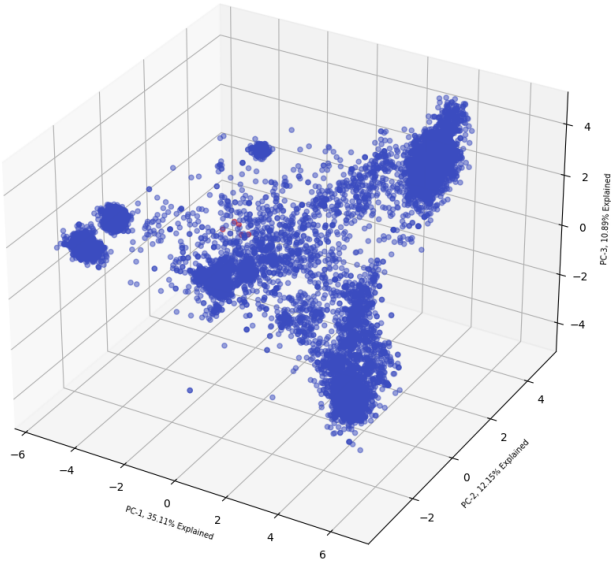


(a) Projection onto 3 main principal components

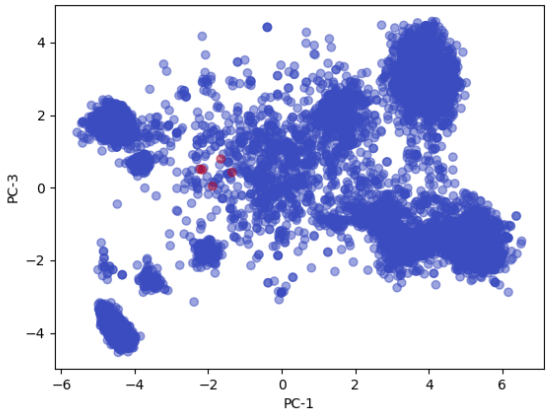


(b) Projection onto 2 main principal components

Figure 14: Test_group1 data points projected to main principal component space



(a) Projection onto 3 main principal components



(b) Projection onto 2 main principal components

Figure 15: Test_group2 data points projected to main principal component space

The second and third test datasets were generated in a completely different random setting. Each comprises of five datafiles, hence we observe less amount of red dots on Figure 14 and Figure 15. Interestingly for both cases we also observe the test data points projected near the center of the 3d, 2d plots. One difference would be that the test data points for Test_group2 is more congregated with each other than that of Test_group1.

	test_group1	test_group2
actual average fitness	[-10000.0, -27.9, -22.0, -106.3, -9396.6]	[-51.2, -51.0, -44.0, -23.0, -755.0]
predicted fitness with k = 2500, median	[-4995.5, -3831.0, -711.0, 0.0, 0.0] all invalid	[-9657.5, 0.0, -6261.0, 0.0, 1.0] all invalid
predicted fitness with k = 10, median	[0.0, -99.75, -99.75, 0.0, -99.75] all valid	[0.0, -99.75, -99.75, 0.0, -99.75] all valid
predicted fitness with k = 2500, mean	[-3767.25, -3831.0, -711.0, 0.0, 0.0] all invalid	[-9657.5, 0.0, -7507.333, 0.0, 1.0] all invalid
predicted fitness with k = 10, mean	[-2126.524, -2172.564, -2172.564, -2126.524, -2172.564] all valid	[-2126.524, -2172.564, -2126.524, -2172.564, -2172.564] all valid

Table 2: Comparison of prediction by number of clusters and cluster fitness value assignment method on both test_group{1,2} datasets

Table 2 summarizes the prediction results for two different cluster amount values and two cluster fitness value assignment methods, namely mean and median. Accuracy of course was very low converging to 0.0. However, looking at the validity categorizations, we can confirm some assumptions we had on the effect of number of clusters to the validity that lower the amount of clustering, we see an underfitting thus, any data point in the principal component space is bound to have distance less than the maximum distance from the centroid to the data point in the cluster.

9. Outlook

The main issue with this result is that the test sets contained only samples outside of our pre-declared prediction range. The test sets are also in fact too small (e.g. ten samples from two different populations) to make decisions. Of course, the model can be improved by optimizing the parameters such as different number of clusters, method of fitness value assignment to cluster, checking with better scoring methods, etc. But above all, we suggest running as many simulations with different permutations in different mazes to average out the uncertainties and get a mean fitness value for each individual ant, respectively each network. Then repeat the proposed training process to produce better performing inference model.

A. Appendix

References

[1] X. Yang. *Nature-Inspired Optimization Algorithms*. Elsevier Inc., 2 edition, 2021.