Estimating similarity of neural network dynamics

Maciej Komosinski Krzysztof Rosiński

Technical Report RA–10/10

Institute of Computing Science
Poznan University of Technology
December 2010

maciej.komosinski@cs.put.poznan.pl

Abstract

This report concerns estimation of the similarity between neural networks of any topology. Motivations and benefits of having an automated and quantitative network comparison mechanism are presented. The concept of neural network dynamics (neuron output signal) is considered. A measure is proposed for estimating similarity of active (i.e., working) neural networks. Properties of the measure are analyzed theoretically and verified empirically. The experiments have been performed on a set of evolved networks responsible for controlling 3D structures (agents, robots). These experiments demonstrate the capabilities and the limitations of the proposed measure as a mechanism to support humans in analyzing large sets of neural networks.

Contents

1 Introduction

Artificial Life is a fast-developing branch of science strongly related to many other disciplines. There are many simulation environments that enable execution of sophisticated experiments [11], and thus the need of obtaining expensive equipment (in order to carry them out physically) may be postponed until positive results are observed. However, computer simulations produce enormous amounts of data. It is not feasible for a human to analyze each evolved creature/agent and compare it to the others. There is a need for tools that could preprocess the data and arrange it in groups to make it easier to interpret. An intuitive clustering criterion is creature dissimilarity (or similarity) [12], yet the problem is in measuring high-level behaviors with available low-level data. One can compare creatures by a combination of their physical features and effector behavior. In general, the physical structure is a 3-dimensional graph that consists of sticks (edges) and tendons/muscles (vertices). For the purpose of comparing such structures, protein structure alignment algorithms may be used (tertiary protein structure may have similar representation [8]). Such metrics are good for comparing static structures, but they provide no knowledge about creature behaviors. Therefore, there is a need to compare neural networks.

Technically, a neural network is a set of connected neurons, each usually containing weighed inputs, an output, and an activation function. In the considered case, some inputs may be connected to receptors (creature senses/sensors) and some outputs may be connected to effectors (mainly muscles). In the higher level of abstraction a neural network represents the creature brain and nervous system, able to receive information from the environment and affecting it via effectors. The hidden and input layers of a neural network usually contain information about its reflexive behavior, however they may even encode cognitive functions and natural intelligence [6]. In order to compare these networks, some information about their behavior must be extracted. According to the methods presented in [1], there are two main approaches of analyzing neural networks: the "black-box" approach, which analyzes the network by sampling input signals and evaluating the output, and the pedagogical approach that tries to build a logical, or mathematical, relation between the input and the output layers by checking neural connections and weights.

If a measure that compares dynamics of any neural network [21] with any other would exist, and if it would return a sensible estimate of similarity, then new possibilities to process data would appear. Additional experiments can be performed on the analyzed networks in order to verify the impact of each networks characteristics on their similarity (i.e., global convexity analysis, or parameter-similarity correlation). The measure may be used to build a dissimilarity matrix (data preprocessing), such data may be used to cluster a large amount of data into few general profiles (centroids or medoids) for the purpose of performing more sophisticated and time-consuming operations or assigning objects to classes. Finally, a dissimilarity matrix enables representing the analyzed objects in a two- or three-dimensional space (using multi-dimensional scaling, with possible information loss), enhancing human interpretation capabilities [13].

2 Possible approaches to comparing neural networks and their dynamics

In order to compare neural network dynamics, one has to analyze the appropriate output series and extract some information that describes them. Time series analysis has many applications besides the presented problem: economy [7], signal processing [20], control of continuous processes [2], meteorology [22], neurology [23], and many more. Time series and their corresponding models have been analyzed theoretically and practically throughout the past decades [4, 3]. In the literature concerning this topic [5, 19], four most commonly used fami-

lies of methods have been distinguished: polynomial methods, least-square methods, Fourier methods and space-state model methods. The polynomial methods rely on polynomial algebra, difference and differential equations or space-state equations. Due to their complexity and characteristics, the polynomial methods are not fit for solving the presented problem: the polynomials that could be used would have large degrees and often the number of terms would differ for different sample count.

The least-square methods are a set of statistical operations that may be used on series of discrete data. Such include classical regression analysis, recursive least square estimation, polynomial trend estimation, and time series smoothing. These methods are commonly used to analyze time series due to their simplicity, low computational complexity, and mostly informative results (i.e., stock and shares price analysis). The mean square error method is a frequent tool for calculating data series analysis and it is an important function of the proposed algorithm.

The Fourier methods (discrete Fourier transform, Fourier integrals) transform the series from the time domain to the frequency domain, which is useful when the series are periodic (i.e., in signal processing). These methods are essential to perform a cepstral analysis [9] of a sample of sound, which is currently a key method of pitch recognition. The Fourier transformation has been used in the proposed algorithm because usually neuron responses tend to be periodic.

The space-state methods are frequently used in control engineering and operational research. The space-state models rely on mathematical analysis and applied mathematics: the model itself is represented by a set of differential equations that explain the relation between inputs and outputs. This is by far the most sophisticated model and applying it to the algorithm would require the use of numerical methods, which give accurate results only at a cost of high computational complexity. Furthermore, information about creature receptors would be required to create such models, yet it is unavailable – the only input data are the time series. Summarising, the space-state methods are a not suitable for the considered purpose.

Based on the above analysis for evaluating similarity of neural network dynamics the mean-square error (MSE) function has been chosen. It operates on series of neural output data, or on the discrete Fourier transform of such series. The algorithm and other technical issues are presented in the following sections.

3 Numerical measure of similarity of neural network dynamics

3.1 The algorithm

Given a set of neural networks, one expects to get a dissimilarity matrix, $DissimilarityMatrix$. Therefore, for each pair of networks in the set, the algorithm tries to map the neurons from the first one to the neurons in the second one and estimate how fit the match is.

The following naming convention has been used to describe the algorithm:

- Scalars – italic font, starting with a lower-case letter (e.g. $dissimilarity$).
- Vectors/matrices/sets – italic font, starts with a capital letter (e.g. $Networks$).
- Functions – plain font, all letters lower-case letters (e.g. neurosim).

Below the most important variables and functions are described:

- $Networks$ – an array containing sets of neural network time series (1 set per creature).
- $DissimilarityMatrix$ – a square matrix containing network dissimilarity values.
- $NeuronMatrix$ – a temporary matrix helpful in finding a good neuron match.

- $Effectors$ – a set of neuron types considered as effectors.
- neurosim – a similarity evaluation function (described after the algorithm).
- neurons($Network, Types$) – extracts neurons of specified types from a neural network.
- matrix($sizeX, sizeY$) – creates an empty matrix of the requested size.
- rmv($Matrix, col(value)$) – removes a column that contains a specified value from the matrix. A column is removed even if many columns contain the same value, in such case the column with the lowest index is removed. In order to simplify the algorithm such features have not been presented in the code listing.
- rmv($Matrix, row(value)$) – like rmv($Matrix, col(value)$), except that it removes a row instead of a column.

```
1:  Networks ← loadData()
2:  DissimilarityMatrix ← matrix(|Networks|, |Networks|)
3:  for i = 1..|Networks| − 1 do
4:    for j = i + 1..|Networks| do
5:      CommonNeuronTypes ← (types(Networks[i]) ∪ types(Networks[j])) ∩ Effectors
6:      dissimilarity ← 0
7:      maxNeuro ← max(|neurons(Networks[i], Effectors)|, |neurons(Networks[j], Effectors)|)

8:      for k = 1..|CommonNeuronTypes| do
9:        Neurons_i ← neurons(Networks[i], CommonNeuronTypes[k])
10:       Neurons_j ← neurons(Networks[j], CommonNeuronTypes[k])
11:       NeuronMatrix ← matrix(|Neurons_i|, |Neurons_j|)
12:       curDif ← abs(|Neurons_i| − |Neurons_j|) · penaltyValue
13:       for l = 1..|Neurons_i| do
14:         for m = 1..|Neurons_j| do
15:           NeuronMatrix[l][m] ← neurosim(Neurons_i[l], Neurons_j[m])
16:         end for
17:       end for
18:       while rows(NeuronMatrix) > 0 ∧ cols(NeuronMatrix) > 0 do
19:         bestMatch ← min(NeuronMatrix)
20:         curDif ← curDif + bestMatch
21:         rmv(NeuronMatrix, row(bestMatch))
22:         rmv(NeuronMatrix, col(bestMatch))
23:       end while
24:       dissimilarity ← dissimilarity + curDif
25:     end for
26:     NotMatched ← neurons(Networks[i]∪Networks[j], Effectors\CommonNeuronTypes)

27:     penalty ← |NotMatched| · penaltyValue
28:     DissimilarityMatrix[i][j] ← (dissimilarity+penalty) / maxNeuro
29:   end for
30: end for
```

The neurons are grouped according to their type (lines 9-10), thus a neuron may be compared only to a neuron of the same group (i.e., "bend muscles" are not compared to "rotation muscles" because their role is usually different). When the same type of groups is extracted from both networks, a penalty value proportional to group size difference is added to the overall network pair dissimilarity (lines 26-27). After that, a temporary neuron dissimilarity

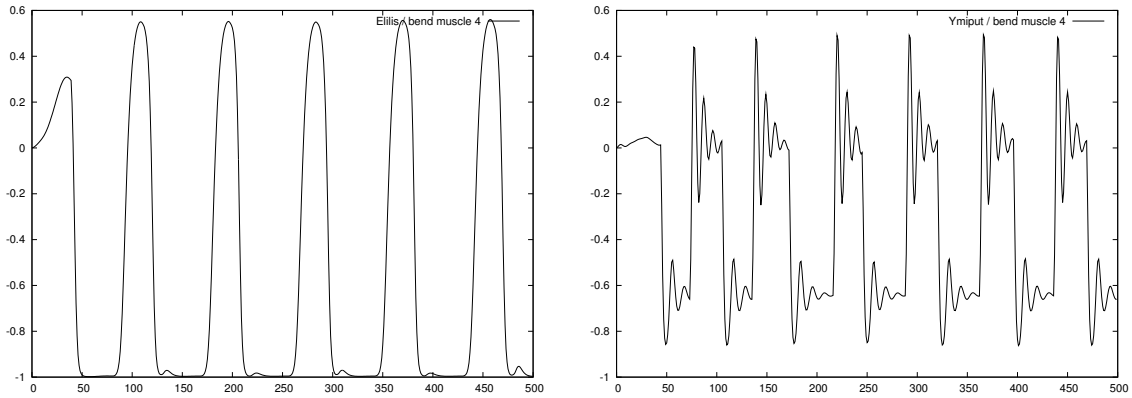

Figure 1: Corresponding neuron outputs from two related creatures.

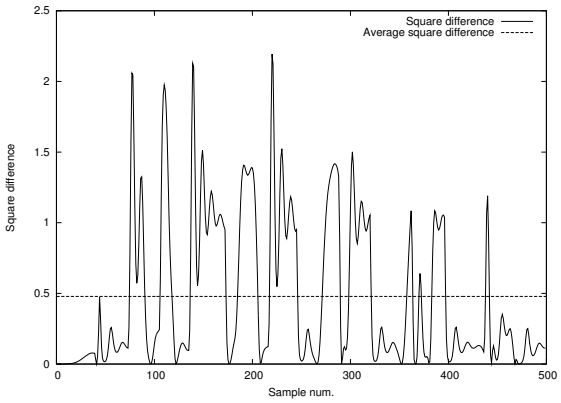

Figure 2: Square error function of the corresponding neuron outputs.

matrix is evaluated by the neurosim, which is the mean-square error function that operates on raw neuron outputs, or on Fourier transform of neural outputs. In the next phase, in order to ensure that the comparison is symmetric, each iteration selects the best match (line 19) and removes the appropriate rows and columns from the neuron dissimilarity matrix (lines 21-22). When this process is completed, the dissimilarity value is added to the global dissimilarity (line 24). The total normalized network dissimilarity is then assigned to the appropriate cell in the $DissimilarityMatrix$ (line 28).

3.2 Illustrative examples

The following examples use creatures from the Framsticks simulator [18, 17]. The simulator models three-dimensional structures made from "parts" and "joints" (sticks). Open-source SDK is provided [16] along with a number of tools to facilitate the manipulation of genotypes and the transformation of such 3D models.

The first example presents the results of comparing networks with a fixed topology. Only neuron input weights may differ. Such situation usually occurs in a short-term evolution, thus there is a need to determine if original and evolved creatures are alike.

Fig. 1 presents the output time series of two corresponding neurons (the probability that the algorithm matches them together is significant). Although they are not the same, the trend is similar. Fig. 2 contains the square error function of these time series. The mean square error of this pair equals 0.47. The normalized MSE is an element of the global dissimilarity value. Assuming that other neurons produce a similar MSE values, the dissimilarity of the analyzed pair would be ≈ 0.12, which is a fair result.

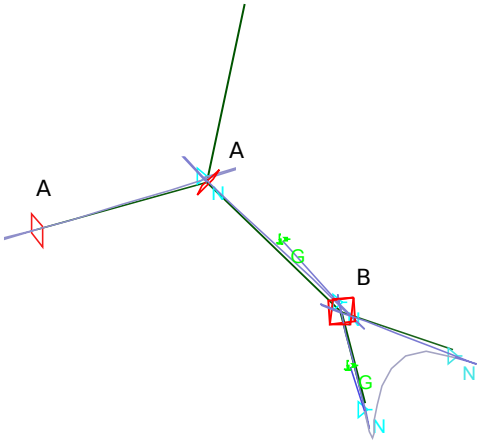

Figure 3: The model of the "Rolling blender" creature [18, 17] with body sticks depicted as lines. Effectors (muscles) are depicted as red squares, and sensors ("G" for gyroscope) are green. Standard neurons ("N") are blue.

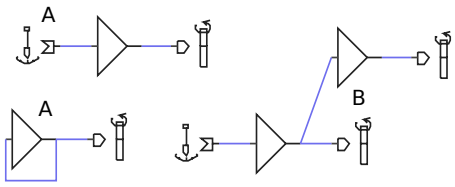

Figure 4: Rolling blender's neural network (effectors A on the left, B on the right).

The second example shows the effects of comparing neural networks which significantly differ in output neuron count (the second network has twice as many effectors as the first one). In such cases, the metric imposes a penalty on the similarity value, thus the dissimilarity increases. Assuming without the loss of generality that there is exactly one type of output neurons, the minimal dissimilarity value for networks A and B equals $\frac{\mathrm{ABS}(|A|-|B|)}{\mathrm{MAX}(|A|,|B|)}$, where $|A|$ and $|B|$ indicate the power of the output neuron set of networks A and B respectively. Figure 3 presents an example model of two creatures. The first creature contains only effectors marked as A, while the second one contains effectors A and B (networks associated with these effectors are presented on Fig. 4). These two creatures are expected to have a low dissimilarity value: their physical models are exactly the same, they only differ by 5 neurons (2 effectors), yet their activity remains similar (they keep rolling in the same manner). Even if the additional neurons have no affect on the agents performance, the dissimilarity value exceeds 0.5. In this example two additional effectors mean that the half of the total effectors cannot be matched. The penalty in such a situation is high and thus the dissimilarity value will be inadequate compared to the subjective dissimilarity produced by a human expert.

3.3 Fourier window size and its impact on the results

In general, the series may not be periodic, thus the frequency domain representation may contain noise and it may be hard to determine what window size provides sufficient information about the analyzed series. The experiment has been performed to measure the information gain from increasing windows size. A random set of 10 neural networks has been selected, upon which the algorithm was executed with a different Fourier window size: 4, 16, 64, 256, 1024.

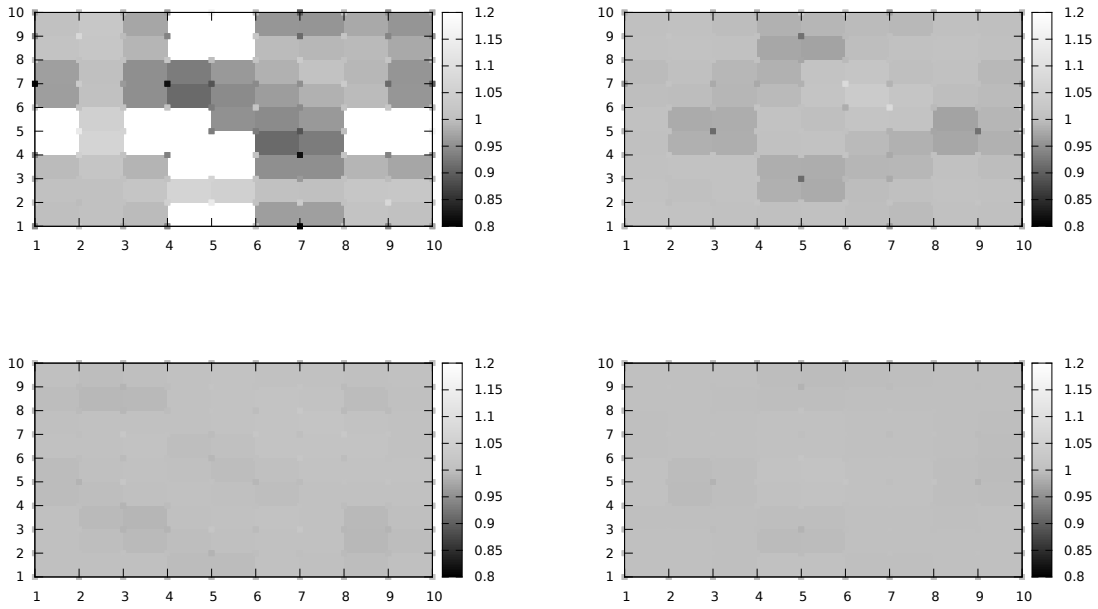

Figure 5: Relative information precision change, Fourier window increased: 16/4, 64/16, 256/64, 1024/256, respectively.

Each matrix (except the first one) has been divided by the matrix from the predeceasing test. The matrices have a similar value range, therefore dividing their elements by elements from another matrix gives the relative dissimilarity change. If increasing the window would not increase information precision, the relative dissimilarity change should be 1. The information change matrices are presented on Fig. 5. It may be observed that the information precision is lowest in the top-left matrix. Increasing window size beyond 64 seems not to improve the result quality in a dramatic manner, thus the window size of 100 has been selected for experiments. This should ensure good quality of results: the set of test creatures was derived from the experiment population, and each individual had a number of effectors that affected the evaluation, thus the choice of the window size is statistically justified.

3.4 Properties

A measure $d(i, j)$, where i and j are neuron output time series, may be considered a metric if the following conditions are satisfied:

(a) $d(i, j) \geqslant 0$

(b) $d(i, j) = d(j, i)$

(c) $d(i, j) = 0 \Leftrightarrow i = j$

(d) $d(i, k) \leqslant d(i, j) + d(j, k)$

The mean square error function prevents the result from being a negative value, which satisfies condition (a). Condition (b) requires that each pair of elements has exactly one dissimilarity value, regardless of the element order. This condition is also satisfied, a greedy algorithm ensures that every time a pair of networks is compared, the same neuron match is generated, thus w dissimilarity value remains unchanged. Condition (c) states that a dissimilarity value is 0 if and only if a creature is compared to itself (or any other creature that is by any means

equal). The algorithm returns a 0 dissimilarity value when each pair of neuron may be perfectly matched, which indeed indicates equality on this level of abstraction. The final condition (d) requires the triangle inequality to be fulfilled. Unfortunately this cannot be accomplished due to the characteristics of the used methods and the neuron outputs being real numbers. A simple hypothetical counter example for this condition can easily be generated. Let there be three creatures i, j and k, each having only one effector. For simplicity let us consider that the simulation lasted one time unit and generated a scalar for each neuron. The output values are 1,0,−1 for i,j,k respectively, thus $d(i,j)$ and $d(j,k)$ equals 1. $d(i,k)$ on the other hand equals 4 (2 squared), which results in $d(i,k) \nleq d(i,j) + d(j,k)$. This means that the proposed measure cannot be considered strictly a metric. An experiment has been made to evaluate how many comparison pairs satisfy this condition in practice. A dissimilarity matrix has been calculated for all evolved creatures (247 unique creatures). From this set a random ordered triple (a,b,c) has been selected 10.000 times. A test has been considered positive when $d(a,c) \leqslant d(a,b)+d(b,c)$. According to this experiment condition (d) is fulfilled in 99.2% of cases.

The proposed measure puts most impact on the size of the neural network, and therefore creatures with significantly different brain size tend to have a high dissimilarity value. This is justified by the fact that complexity of a neural network is related to its size.

The method is vulnerable to signal transformations. Comparing similar periodic impulses with different frequency and amplitude (for example, bigger creatures have slower but stronger bending muscles) may give a high mean square error and thus produce a high dissimilarity value. The Discrete Fourier Transformation (DFT) method is more suitable for handling some affinity transformations (like, for example, signal offset). On the other hand, the frequency domain approach is vulnerable to irregular signal peaks that produce distortion.

3.5 Time complexity

The time complexity of the algorithm is dependent on the characteristics of compared neural networks. Comparing c creatures in order to create a dissimilarity matrix requires exactly $\frac{c(c-1)}{2}$ comparisons. Let us define g – the number of neuron types in all creatures, n – the average number of neurons per neuron type, and t – simulation time (the number of samples in time series). Each comparison consists of two phases: calculating the penalty for comparing networks of different size, and evaluating a (suboptimal) mean square error value for the appropriate neuron sets. Because penalty calculation performs only simple arithmetic operations on neuron type cardinality, its complexity is $O(g)$. Evaluating the minimal mean square error of 2 neurons requires t operations; a greedy algorithm performs this procedure at most n^2 times for each neuron group. To sum up, each comparison between two neural networks requires $g(1+n^2t)$ operations, thus the computational complexity of running the algorithm is $O(c^2gn^2t)$.

3.6 Penalty function for networks with different number of effectors

As it was mentioned in the previous sections, each comparison requires calculating a penalty function for comparing networks of different size, and a dissimilarity function for neuron output signals. For the purpose of presented experiments, the penalty function imposes a maximal possible dissimilarity value for each neuron that exceeds the minimal cardinality of the respective neuron groups for compared creatures. This approach makes neural networks with a significantly different number of elements unlike: a creature that has two legs (2 sets of muscles) has a greater chance to be similar to another two-legged creature than to a six-legged creature.

Another approach assumes that the dissimilarity value for each pair of neurons has been already calculated, and this would allow the penalty function to provide a value equal to the average (or median) of the calculated set. This approach is quite intuitive, as we expect a

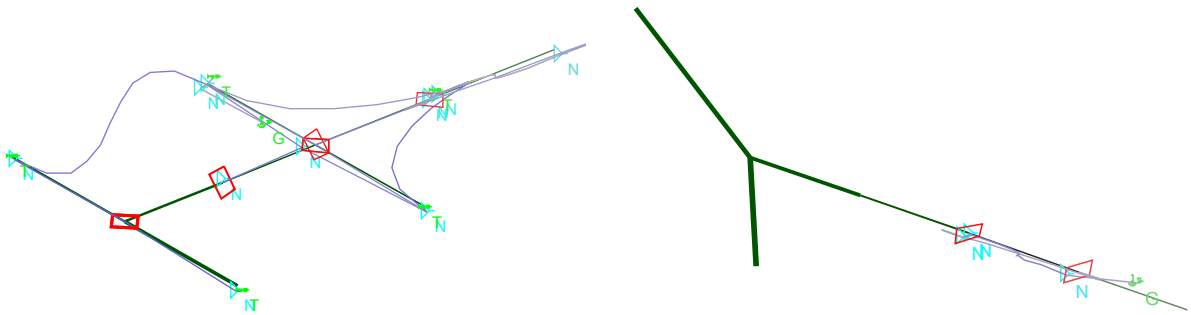

Figure 6: “Fast lizard” and “Scorpion” morphologies.

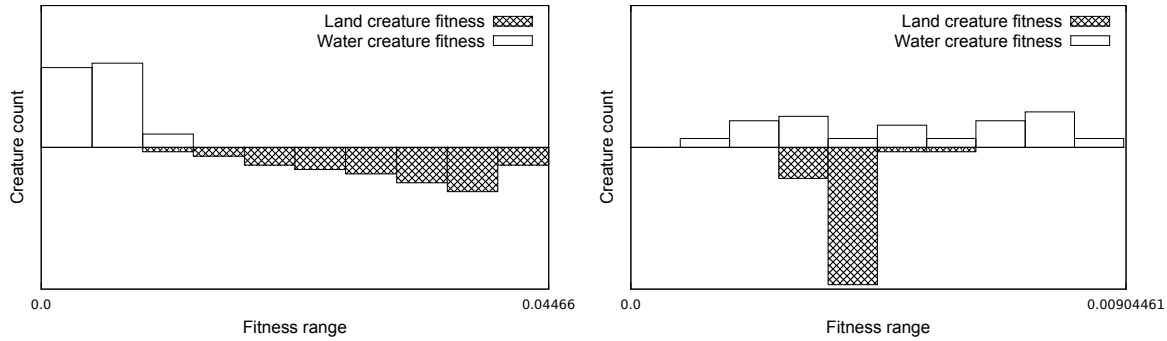

Figure 7: Fast lizard and Scorpion fitness distribution.

long snake to be similar to a short snake despite the fact that they differ in size, because the excessive effectors have a common function with primary ones.

It was also considered not to use a penalty function at all. Although there are examples that justify this variant, in general this approach is weak. The lack of a penalty function would surely increase the similarity of creatures with large differences in the number of neurons, as a small set of neurons has a greater chance to find a good match in a much larger set.

Each penalty function variant has its positive and negative aspects. The proposed algorithm uses the first type – imposing a maximal possible penalty. The motivation for this choice is that it is more acceptable not to discover similarity between similar creatures, than to discover it between differing ones.

4 Application in automated analysis

Just as in the previous sections, here simulated creatures come from the Framsticks system [18, 17]. In all experiments, the morphology of the body was fixed (not modified during optimization), and only specific aspects of the neural network were subject to change. In terms of performance/fitness, we consider average velocity of creatures.

4.1 Fixed NN topology

The first set of creatures had a fixed neural network topology, and the body structure was fixed for each set of creatures: scorpions, and fast lizards (Fig. 6). Performance of each group was evaluated in two environments: with no water and with a high water level. The second environment has a greatly decreased gravity compared to the first one. The fitness distribution of the analyzed groups is presented in Fig. 7.

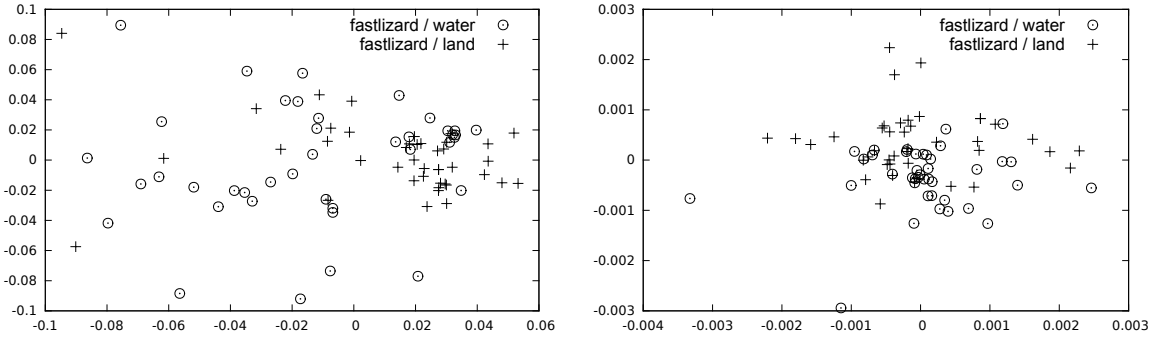

Figure 8: Fast lizard dissimilarity matrix mapped on a 2D surface – methods: raw MSE (24.6% of the total variance preserved) and DFT (25.5% of the total variance preserved).

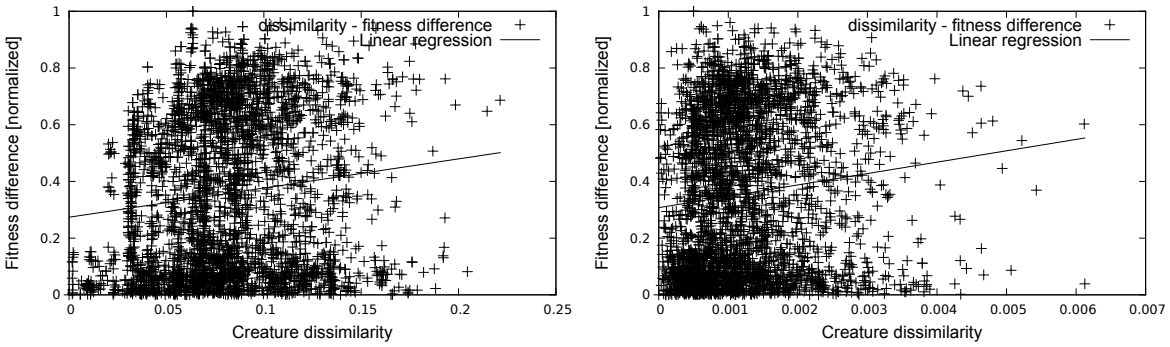

Figure 9: Fast lizard correlation analysis – methods: raw MSE ($r = 0.12$) and DFT ($r = 0.11$).

The dissimilarity matrix of neural signals of the fast lizards was scaled into 2 dimensions and presented on Fig. 8. It may be observed that creatures from different environments may be easily distinguished. After executing the classical multidimensional scaling procedure, creatures from each environment have a significantly different medoid position, yet there are many outliers that make linear separation impossible. The "scorpion" dissimilarity is presented on Fig. 11. In this experiment the two environments have been separated in a better manner. Furthermore it can be observed (especially from the DFT methods output) that some scorpions from the water environment are more similar to scorpions from the land environment, than to other water scorpions. These usually have significantly lower fitness value, thus this effect is justified.

Global convexity analysis (Fig. 10) shows whether creatures fitness correlates with their similarity. Since the histograms of these groups are known (Fig. 7), these groups can be easily distinguished. Analysing the RAW method graph for fast lizard population (Fig. 10), one can see that the regression slope is approximately zero. Furthermore, fast lizard similarity-fitness correlation analysis (Fig. 9) gives the same conclusion. The DFT methods indicates otherwise: the regression slope is negative with reference to the dissimilarity value, thus it has a positive correlation with the similarity measure. The scorpion (Fig. 13) analysis also indicates correlation. The Pearson product-moment correlation coefficient (absolute value) exceeds 0.5 which confirms the observation. This means that creatures from the currently analyzed groups tend to be affinity to creatures with a similar velocity value.

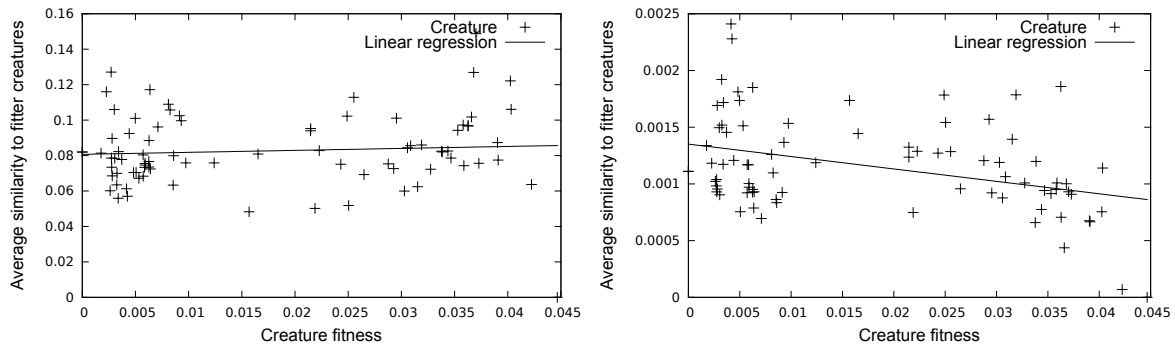

Figure 10: Fast lizard dissimilarity global convexity measurement – methods: raw MSE and DFT.

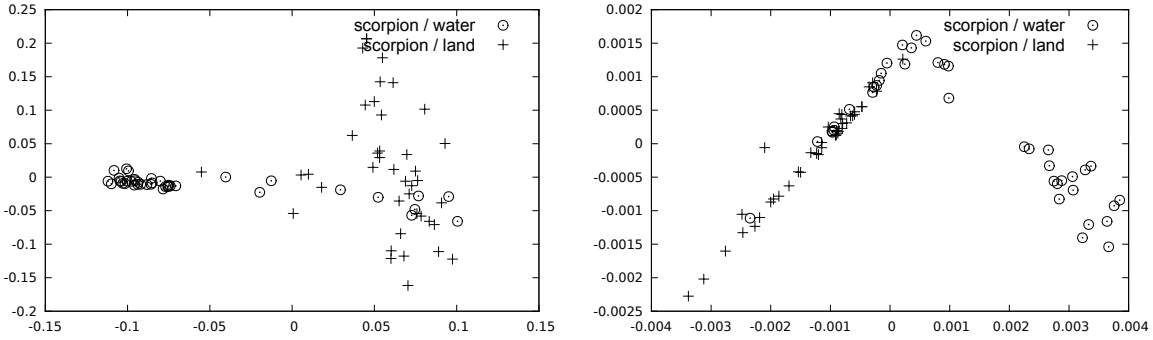

Figure 11: Scorpion dissimilarity matrix mapped on a 2D surface – methods: raw MSE (14.7% of the total variance preserved) and DFT (59.0% of the total variance preserved).

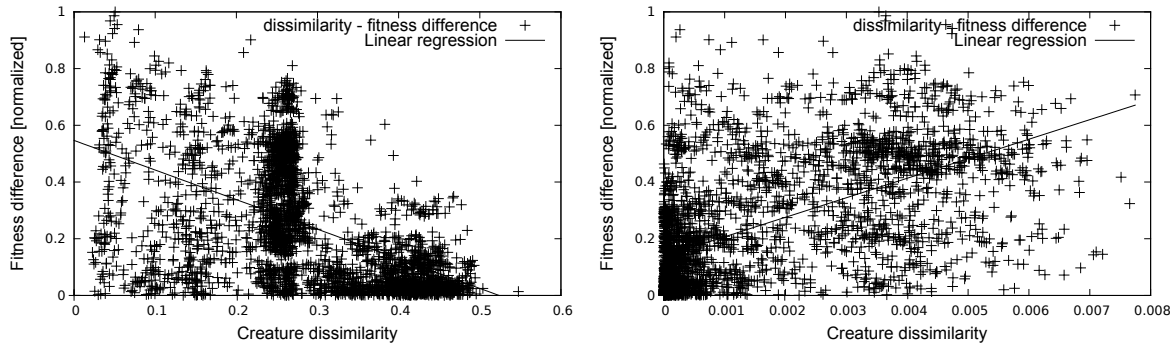

Figure 12: Scorpion correlation analysis – methods: raw MSE ($r = -0.53$) and DFT ($r = 0.57$).

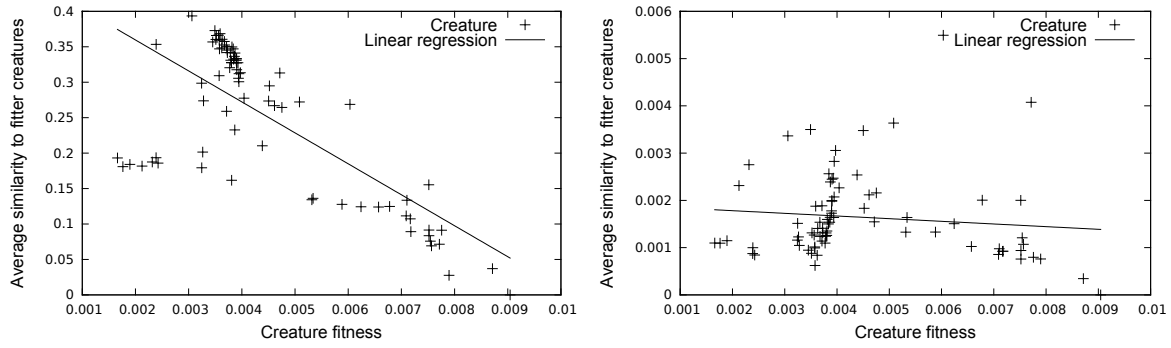

Figure 13: Scorpion dissimilarity global convexity measurement – methods: raw MSE and DFT.

Figure 14: "Minisnake" morphology: initially, muscles work in a different plane in each group of creatures.

4.2 Variable NN topology, two variants of body

The second set of creatures contains organisms that do not differ in body morphology, yet their neural networks structures have been evolved and contain different numbers of neurons (Fig. 14). The set consists of two groups of creatures that differ in initial spatial orientation. Therefore, one group has been named 'vertical', and the second one 'horizontal'. Since the body capabilities are similar for both groups, the fitness distribution is also alike (Fig. 15).

The dissimilarity matrix mapped on a 2D surface is presented on Fig. 16. Unlike in the previous experiment, it is difficult to find any patterns in the creature distribution, even though mapping of the DFT results preserved over 70% of the total variation. The indiscernibility of creatures in this case may be a result of their morphological similarity. It may also mean that both groups are very much alike, yet executing the multidimensional scaling procedure

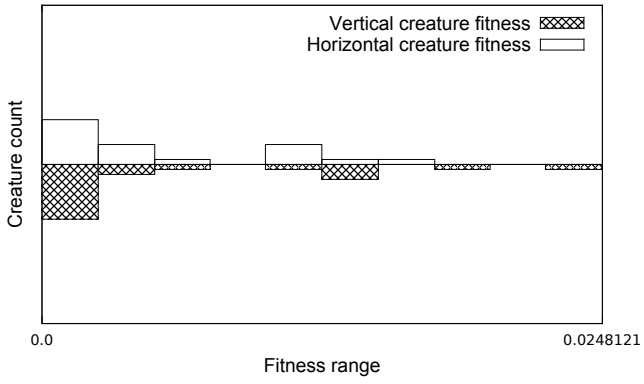

Figure 15: Minisnake fitness distribution.

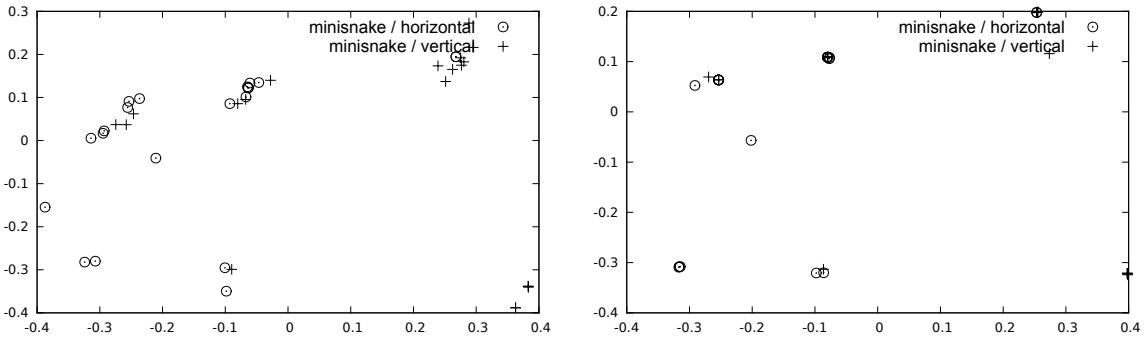

Figure 16: Snake dissimilarity matrix mapped on a 2D surface – methods: raw MSE (52.8% of the total variance preserved) and DFT (70.4% of the total variance preserved).

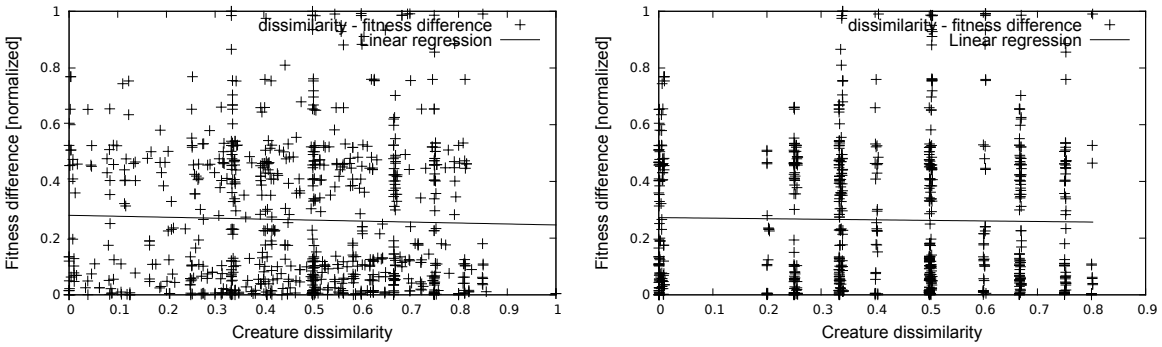

Figure 17: Snake correlation analysis – methods: raw MSE ($r = -0.03$) and DFT ($r = -0.02$).

scales minor varieties into major differences. However when these groups are compared to more diversified ones a greater similarity may be observed (Sect. 4.4). The fitness-similarity correlation graph is presented in Fig. 17. Most of the compared pairs are arranged vertically along a few dissimilarity value. This is a result of comparing very primitive organisms with simple networks, which contain few effectors: effector type and differences in number have the highest impact on the dissimilarity value. Since there are far more compared pairs than structurally different pairs of networks, differences have practically a discrete domain. The points are arranged into vertical stripes, and these groups lack correlation between similarity and performance (Pearson correlation ≈ 0).

4.3 Variable NN topology

The 'star' set of creatures has been evolved with no restrictions regarding neural networks. Unlike the previous experiment, only one variant of body exists (Fig. 18), but the experiments have been performed in different environments (similarly to the Fast lizard and Scorpion groups). There are small disproportions between land and water fitness distribution (Fig. 19); most organisms have a low performance value.

Analysing Fig. 20 it may be observed that the land and water creatures are mixed and no clusters may be isolated. This case is similar to the previous experiment, only the circumstances (not restricted evolution) greatly increased the differences between individuals. There is also a positive correlation between dissimilarity and fitness (Fig. 21), which indicates that creatures are more frequently similar to creatures with a different performance value. This may be a result of the evolutionary methods used to generate the creatures. When some creatures were derived from the same parent, they may be considered similar by the algorithm,

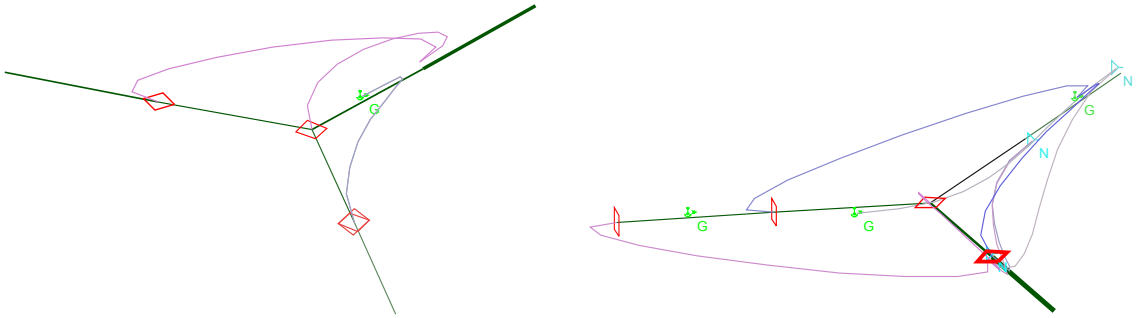

Figure 18: The “Star” morphology.

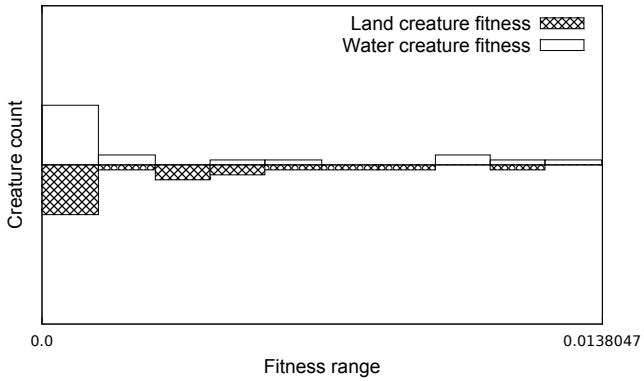

Figure 19: “Star” fitness distribution.

however chances that all of them will achieve the same performance value are low. Usually most of the created organisms are going to be evolutionary ‘dead-points’ (poor local optima), but a few may be considered fit – which justifies the results. A correlation value of ≈ 0.15 does not indicate a linear relation between the parameters.

4.4 Mixed populations

Each of the groups considered in earlier experiments contained similar creatures, because in each experiment they were evolved from a common ancestor. These groups were now mixed

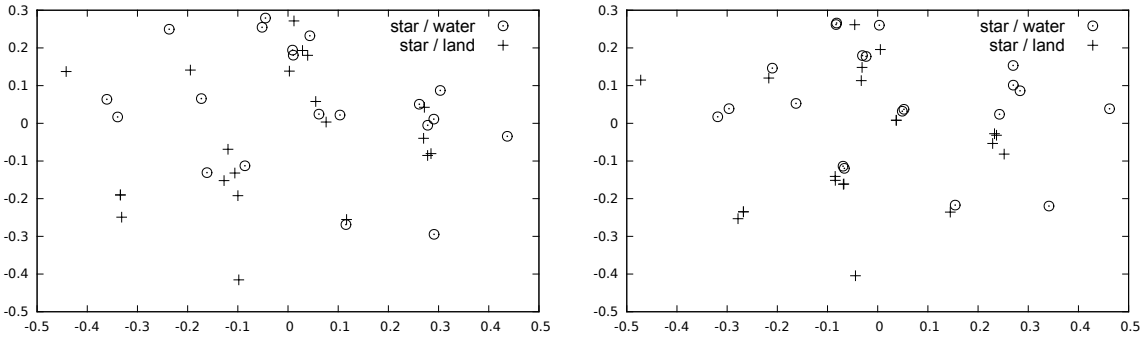

Figure 20: “Star” dissimilarity matrix mapped on a 2D surface – methods: raw MSE (42.7% of the total variance preserved) and DFT (48.9% of the total variance preserved).

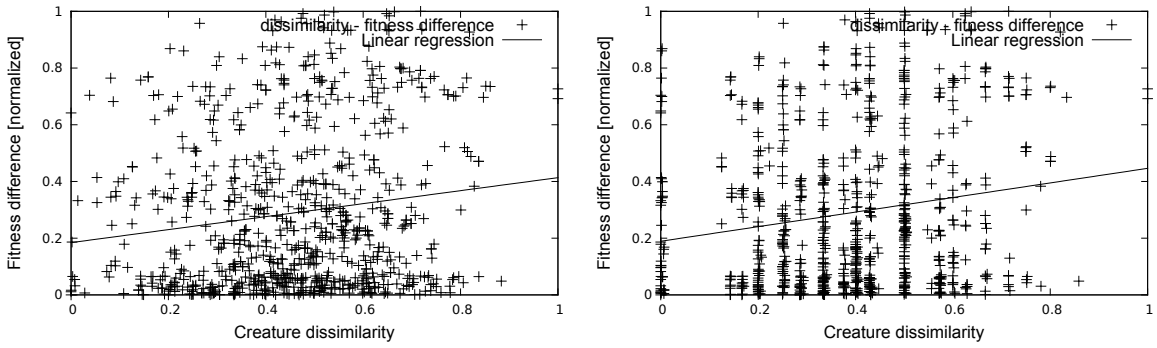

Figure 21: "Star" correlation analysis – methods: raw MSE ($r = 0.15$) and DFT ($r = 0.16$).

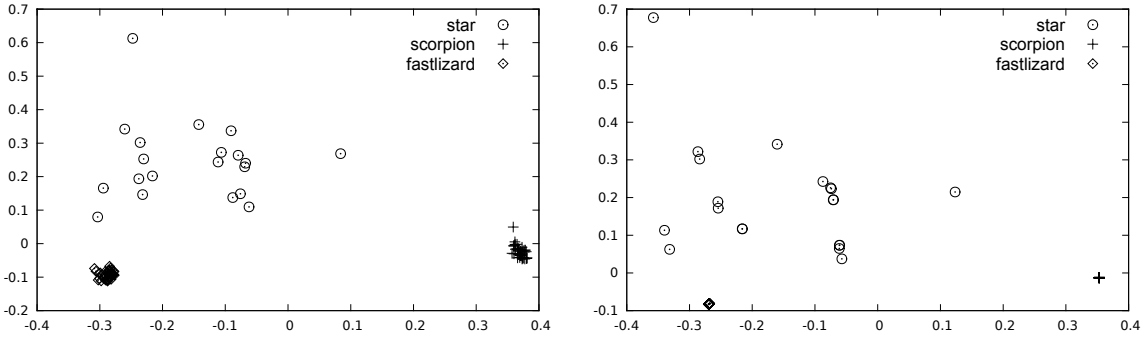

Figure 22: Mixed land populations mapped on a 2D surface – methods: raw MSE (59.3% of the total variance preserved) and DFT (84.4% of the total variance preserved).

in order to obtain results from a heterogeneous set. Fig. 22 and 23 present results for land and water environments respectively. It may be observed that clusters are formed in the comparison process. Yet there are exceptions to this: some creatures, especially from the Star group, are scattered in a relatively large radius and away from other creatures. This may be caused by a non-restricted evolution model applied to this creature group.

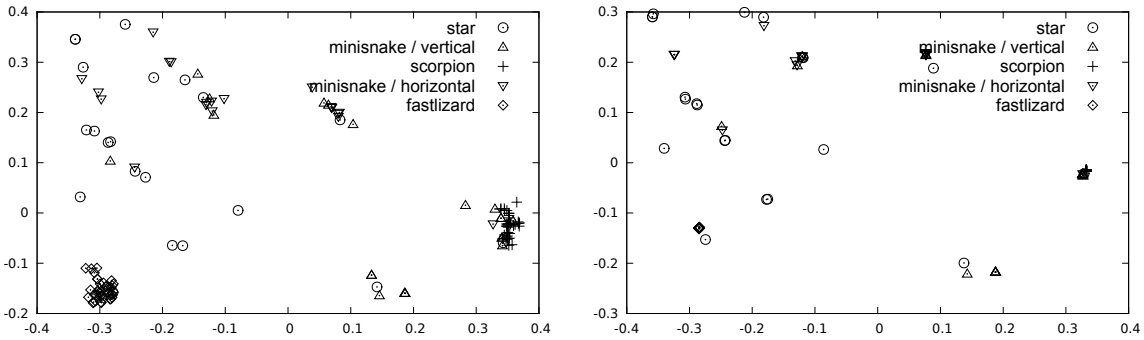

Figure 23: Mixed water populations mapped on a 2D surface – methods: raw MSE (53.3% of the total variance preserved) and DFT (68.0% of the total variance preserved).

5 Conclusions and further work

Based on these preliminary results it may be said that the proposed method of comparing neural network dynamics gives relatively objective information about their similarity. It works exceptionally well when potentially similar organisms have a common neural network structure, which greatly enhances neuron matching accuracy. On the other hand, creatures left for unrestricted evolution tend to have a more unpredictable similarity value of their neural network activity. All in all, due to a relatively low computational complexity, the proposed tool is already helpful – the generated dissimilarity matrices may not be perfect, but they still provide a general overview of the analyzed populations.

The next step in developing a successful similarity measure that estimates neural network dynamics is to individually study evolved creatures and their neural signals, and point out the desired properties of the ideal measure. The goal is to try to capture and then mimic the behavior of a human expert that would estimate similarity of a series of output values in simple networks. Constructing specific boundary cases may be helpful in this approach as well.

The properties of the measure could be improved by increasing the depth of its analysis. One example of this would be to analyze and try to match subnetworks. Another example would be to assign neurons to groups not only based on their type, but also on their function – i.e., if a neural network module that controls a leg consists of a bending and a rotation muscles, this relation should be observed. Such an ability would decrease the penalty when comparing a four-legged creature and a six-legged creature that employ similar control modules. Another way to improve the algorithm is to use additional information – e.g., to consider body structure [13, 10, 14] to better match sensors and effectors between two networks, or to consider behavioral traits [15]. It is however important not to increase the complexity of the matching procedure too much, as the similarity measure needs to be fast if it is to be applied to analyze large sets of neural networks.

References

[1] Robert Andrews, Joachim Diederich, and Alan B. Tickle. Survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge-Based Systems*, 8:373–389, 1995.

[2] George Box and Gwilym Jenkins. *Time Series Analysis: Forecasting & Control.* Holden-Day, 1990.

[3] David R. Brillinger. *Time Series: Data Analysis and Theory.* SIAM, 2001.

[4] Peter J. Brockwell and Richard A. Davis. *Time Series: Theory and Methods.* Springer, 1991.

[5] James Durbin and Siem Jan Koopman. *Time Series Analysis by State Space Methods.* Oxford University Press, 2001.

[6] Stephen Grossberg. *Neural Networks and Natural Intelligence.* MIT Press, 1988.

[7] G. Gudmundsson. Time-series analysis of imports, exports and other economic variables. *Journal of the Royal Statistical Society*, 134:383–412, 1971.

[8] L. Holm and C. Sander. Protein structure comparison by alignment of distance matrices. *Molecular Biology*, 223:123–138, 1993.

[9] S. Imai. Cepstral analysis synthesis on the mel frequency scale. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '83.*, 1983.

[10] Maciej Komosinski. Applications of a similarity measure in the analysis of populations of 3D agents. *Journal of Computational Science*, 21:407–418, 2017. URL: http://dx.doi.org/10.1016/j.jocs.2016.10.004, doi:10.1016/j.jocs.2016.10.004.

[11] Maciej Komosinski and Andrew Adamatzky, editors. *Artificial Life Models in Software*. Springer, London, 2nd edition, 2009. URL: http://www.springer.com/978-1-84882-284-9, doi:10.1007/978-1-84882-285-6.

[12] Maciej Komosinski, Grzegorz Koczyk, and Marek Kubiak. On estimating similarity of artificial and real organisms. *Theory in Biosciences*, 120(3-4):271–286, December 2001. URL: http://dx.doi.org/10.1007/s12064-001-0023-y, doi:10.1007/s12064-001-0023-y.

[13] Maciej Komosinski and Marek Kubiak. Quantitative measure of structural and geometric similarity of 3D morphologies. *Complexity*, 16(6):40–52, 2011. URL: http://dx.doi.org/10.1002/cplx.20367, doi:10.1002/cplx.20367.

[14] Maciej Komosinski and Agnieszka Mensfelt. A flexible dissimilarity measure for active and passive 3D structures and its application in the fitness–distance analysis. In Paul Kaufmann and Pedro A. Castillo, editors, *Applications of Evolutionary Computation*, pages 106–121, Cham, 2019. Springer. URL: http://www.framsticks.com/files/common/DissimilarityMeasure3DStructuresFitnessDistance.pdf, doi:10.1007/978-3-030-16692-2_8.

[15] Maciej Komosinski and Konrad Miazga. Measuring properties of movement in populations of evolved 3D agents. In Harold Fellermann, Jaume Bacardit, Ángel Goñi-Moreno, and Rudolf M. Füchslin, editors, *Artificial Life Conference Proceedings*, pages 485–492. MIT Press, 2019. doi:10.1162/isal_a_00208.

[16] Maciej Komosinski and Szymon Ulatowski. Framsticks SDK (Software Development Kit). URL: http://www.framsticks.com/sdk.

[17] Maciej Komosinski and Szymon Ulatowski. Framsticks: Creating and understanding complexity of life. In Maciej Komosinski and Andrew Adamatzky, editors, *Artificial Life Models in Software*, chapter 5, pages 107–148. Springer, London, 2nd edition, 2009. URL: http://www.springer.com/978-1-84882-284-9.

[18] Maciej Komosinski and Szymon Ulatowski. Framsticks web site, 2019. URL: http://www.framsticks.com.

[19] David Stephen Pollock. *A Handbook of Time-Series Analysis, Signal Processing and Dynamics*. Academic Press, 1999.

[20] S. D. Stearns. *Digital Signal Analysis*. Prentice-Hall, 1990.

[21] Tim P. Vogels, Kanaka Rajan, and L. F. Abbott. Neural network dynamics. *Neuroscience*, 28(1):357–376, 2005. URL: http://www.annualreviews.org/doi/abs/10.1146/annurev.neuro.28.061604.135637?journalCode=neuro.

[22] T. Wigley, K.R. Briffa, and P.D. Jones. Average value of correlated time series, with applications in dendroclimatology and hydrometeorology. *Journal of applied meteorology and climatology*, 23:201–234, 1984.

[23] T. Yuzuriha. The autocorrelation curves of schizophrenic brain waves and the power spectrum. *Psychiatria et Neurologia Japonica*, 26:911–924, 1960.