Meta-Data (Required for the transfer of your article to MethodsX – will not be typeset)

***Title:** *Max. 20 words.* • *A good title should contain the fewest possible words that adequately describe the content of a paper.*	*Neural Networks As Representation of Collective Intelligence*
***Authors:**	*Krzysztof Wasniewski, PhD*
***Affiliations:**	*The Andrzej Frycz-Modrzewski Krakow University, Faculty of Management*
***Contact email:** *Include institutional email address of the corresponding author*	*kwasniewski@afm.edu.pl*
Co-authors**: *full names and e-mails.* ***[NOTE: it is the corresponding authors responsibility to inform all co-authors if submitting as a companion paper to a research article]	##
***Keywords:** *At least 3 keywords.* • *There is no limit on the no. of keywords you can list.* • *Please remember that effective keywords should not repeat words appearing in your title, and should be neither too general nor too narrow.*	*neural networks, collective intelligence,*
SECTION:** • *Agricultural and Biological Sciences* • *Biochemistry, Genetics and Molecular Biology* • *Chemical Engineering* • *Chemistry* • *Computer Science* • *Earth and Planetary Sciences* • ***Economics and Finance • ***Energy*** • *Engineering* • *Environmental Science* • *Immunology and Microbiology* • *Materials Science* • *Medicine and Dentistry* • *Neuroscience* • *Pharmacology, Toxicology and Pharmaceutical Science* • *Physics and Astronomy* • *Psychology* • *Veterinary Science and Veterinary Medicine*	*Economics and Finance, Energy*

Method Article

*** Title: Neural Networks as Representation of Collective Intelligence**
***Authors: Krzysztof Wasniewski, PhD**
***Affiliations: The Andrzej Frycz-Modrzewski Krakow University, Faculty of Management**
***Contact email: kwasniewski@afm.edu.pl**
*** Keywords: artificial intelligence, collective intelligence**

ABSTRACT
Quantitative variables regarding social phenomena can be considered as informative about the cumulative outcomes of past collective decisions in the given society, and it is possible to hypothesize collective intelligence in these decisions. Observing the way that a neural network processes empirical data can provide insights as for how the given society is collectively intelligent. Demonstrably changing a value of the aggregate fitness function yielded by the neural network is a formal proof of collective intelligence. Euclidean distance between the original empirical data, and that processed by the neural network is, in turn, informative about the outcomes the given collective intelligence actually optimizes. Practical formulation of a neural network into the structure of an Excel spreadsheet is discussed, and examples of the relevant Excel spreadsheets accompany this article.

- **this method allows using empirical, quantitative data in social sciences as base for emulating the collective intelligence of the societies studied**
- **differently from classical quantitative models, this method builds a series of scenarios, or formal hypotheses, as for the underlying values and goals of the societies studied**
- **formalisation of the neural network into an Excel spreadsheet, whilst slower than algorithms in Python or in E, allows hands-on observation of each single step in the treatment of data**

SPECIFICATIONS TABLE

Subject Area	*Select one of the following subject areas:* • *Agricultural and Biological Sciences* • *Biochemistry, Genetics and Molecular Biology* • *Chemical Engineering* • *Chemistry* • *Computer Science* • *Earth and Planetary Sciences* • ***Economics and Finance*** • *Energy* • *Engineering* • *Environmental Science* • *Immunology and Microbiology* • *Materials Science* • *Medicine and Dentistry* • *Neuroscience* • *Pharmacology, Toxicology and Pharmaceutical Science* • *Physics and Astronomy* • *Psychology* • *Veterinary Science and Veterinary Medicine*
More specific subject area:	*Manifestation of collective intelligence in social structures, and the method of studying it.*
Method name:	*Collective intelligence in social structures*
Name and reference of original method	*The so-called 'swarm intelligence' model, presented e.g. in:* *Stradner, J., Thenius, R., Zahadat, P., Hamann, H., Crailsheim, K., & Schmickl, T. (2013). Algorithmic requirements for swarm intelligence in differently coupled collective systems. Chaos, Solitons & Fractals, 50, 100-114.* *The general fitness function:* *De Vincenzo, I., Massari, G. F., Giannoccaro, I., Carbone, G., & Grigolini, P. (2018). Mimicking the collective intelligence of human*

	groups as an optimization tool for complex problems. Chaos, Solitons & Fractals, 110, 259-266 *Theory of social games with imperfect recall:* *Selten, R. (1990). Bounded rationality. Journal of Institutional and Theoretical Economics (JITE)/Zeitschrift für die gesamte Staatswissenschaft, 146(4), 649-658.*
Resource availability	*If applicable, include links to resources necessary to reproduce the method (e.g. data, software, hardware, reagent):* *In the section "**Supplementary material and/or Additional information"**, at the end of this article, the reader can find a list of links to Excel spreadsheets that correspond to those sets, under the general heading 'List of accessory Excel files'. These files are available in the repository of the author's scientific blog.*

The methodological problem to solve

In social sciences, quantitative variables frequently take the form of coefficients, e.g. density of population, energy efficiency, GDP per capita etc. Some of those coefficients bear the marks of local equilibriums. Density of population is probably the most obvious example: it is a local balance between the headcount of population, its spatial movement, and resources available in the given territory. Some other coefficients, such as energy efficiency measured as the value of real output per unit of energy consumed, are somehow puzzling: they can be interpreted both as efficiencies strictly spoken (i.e. as functional attributes of a process subject to optimization), and as local equilibriums. The same coefficients are socially important, and their optimization is an important challenge for whole societies. As we ask, whether at all and to what extent can we optimize, as a society, a given proportion, another question emerges: 'So far, as a society, have we been optimizing this proportion at all? What other types of social optimization is this one coherent with?'. In other words, when we ask if we can collectively learn something, discovering whether somebody somewhere had already learnt something similar is a useful insight.

Besides this fundamental problem, there is another one, more analytical. When we construe an econometric model, or, even more broadly, a sociometric one, distinction between the explained variable, on the one hand, and the explanatory variables, on the other hand, is largely arbitrary. The distinction can be sharpened by using, for example, time-lagged values in the variables deemed explanatory. Still, arbitrariness remains.

Recent advances in Artificial Intelligence result in more and more frequent application of neural networks in social research, as an alternative to classical, regression-based models. From the cognitive point of view, the essential difference between a neural network and stochastic regression is the utilisation of residual errors in estimation. In regression, the predictive function minimizes the average error, i.e. all the local errors are computed, and an error-minimizing function is derived. In neural networks, each consecutive local error is utilised as material for learning as regards the next empirical observation. Neural networks allow deepening the study of equilibrium states in social sciences, by extending the concept of equilibrium into a broader assumption of collective intelligence being at work. Moving equilibriums can be studied as moving homeostasis. The general assumption underlying this method is that empirical values observable in any quantitative variable used to describe a society, such as, for example, GDP per capita, can be considered as the cumulative outcome of past collective actions and decisions. Consequently, mathematically verifiable,

logical links between those variables are informative about correlations between the actions and decisions in question.

The so-called 'swarm theory' (Stradner et al. 2013) provides a possible theoretical basis for using neural networks as viable representations of collective intelligence in human societies. The swarm theory argues that collective intelligence manifests as changing cohesion between individual patterns of behaviour. Three typical levels of cohesion are distinguished: static coupling (the tightest one), dynamic correlated coupling, and dynamic random coupling (the loosest one). The relative strength of coupling between actions and decisions in a society is scalable and measurable as a fitness function, i.e. as the Euclidean distance between the corresponding numerical vectors (de Vincenzo et al. 2018). If a set of empirical data, pertinent to social phenomena, and treated by a neural network, displays visible changes in the value of its general fitness function, it can be assumed that the data in question represents collective intelligence.

Neural networks are made for optimizing. Input variables are being experimented with in order to find such their combination, which yields the greatest accuracy in driving the output variable towards desired values. This property of neural networks can be used in order to discover, which social phenomena, from among those measured with quantitative variables we study, are really the desired outcome of collective intelligence. In other words, a neural network can be used to discover teleological orientations in collective intelligence of human societies. For the same set of empirical data, informative about the state of society, and made of $\boldsymbol{n}$ variables, we can build many neural networks endowed with the same logical structure, and yet differing in the classification of variables into respectively, the output variable subject to optimization, and input variables, instrumental in that view. The values produced by each such network can be compared with the source empirical data, and, logically, the neural network that yields the greatest similarity to the latter can be treated as its best approximation. The output variable of that most similar neural network can be considered as the most likely to be the value optimized by the society under scrutiny. This methodological paper presents an analytical procedure to run such a test.

The logical structure of the neural network used

When using a neural network to simulate collective intelligence, advanced algorithms of deep learning are not necessarily an asset. We need to keep in mind that no external super-intelligence optimizes collective learning in human societies. Empirically observable manifestations of collective intelligence are far from being as lean and efficient, as e.g. man-made algorithms of facial recognition. In the here-presented method, a multi-layer perceptron (MLP) has been originally used, without deep learning as such. Still, possibilities are virtually endless. The logical structure of the MLP used in the here-presented method attempts to represent as accurately as possible the way a real collective intelligence works: theoretical representativeness is the chief concern, with speed and precision of learning being of lesser importance.

The concept of multi-layer perceptron is rooted in the fact that we can associate logically different mathematical functions with corresponding cognitive processes of an intelligent structure: mathematical function A corresponds to cognitive process X1, function B represents cognitive process X2 etc. Each function is a layer of neurons in an MLP. A layer can contain one or more neurons. The baseline solution is one neuron, still many neurons in the same layer can compete, i.e. their numerical outputs can be subject to meta-selection, e.g. the greatest wins, the smallest wins etc. In this paper, the neural network consists of four distinct types of mathematical functions, corresponding to four distinct cognitive processes: perception of external information, auto-perception, neural activation, and forward feeding of results for further learning.

The layer of perception standardizes the source data. X represents the original set of empirical data, consisting of n input variables x and one output variable y; $X = \{x_1, x_2, \ldots, x_n, y\}$, where each x_i represents the i-th input variable, and y is the output variable. Besides being structured into variables, X is structured into m phenomenal occurrences 'o', so as $X = \{o(1), o(2), \ldots, o(m)\}$, and each j-th phenomenal occurrence contains the local values f variables observed: $o(j) = \{x_1(j), x_2(j), \ldots, x_n(j), y(j)\}$. Z represents the standardized set of empirical data, transformed from X via a function of perception $fp(X)$, **where** $z_1 = fp(x_1)$, $z_2 = fp(x_2)$, …, $z_y = fp(y)$; Z is structured into m phenomenal occurrences just as X is. Thus, $z_1(1)$ is the standardized value of variable x_1 in the phenomenal occurrence 1 etc. In the method such as it is presented here, it is assumed that the order of phenomenal occurrences is fixed. It corresponds to the assumption that a society, as collective intelligence, deals with an already given order of events to learn from.

Usually, standardization is not considered something that a neural network does in the strict sense of the term: the data simply should be fed into the network in standardized form. Still, standardization in itself is part of intelligent perception. Typical standardization consists in denominating each variable over its maximum in the set. This is linearly ordered perception, where each stimulus is perceived on a linear scale of its relative strength. However, perception can be oriented probabilistically. If instead of standardizing the source data over the maxima of respective variables, we standardize under the curve of normal distribution marked by the mean value of source data, and its standard deviation, we reproduce the workings of an intelligence perceiving reality so as to assign importance to phenomena relatively to their likelihood of happening. We can make the perception more selective, i.e. make it distinguish more sharply between the important stimuli and the unimportant ones by inducing standardization under a Poisson curve. In mathematical terms, it means that many local observations in source data will suddenly acquire standardized values equal to zero, and others will be equal to 1, and that changes a lot in the final output of the perceptron. As we make a parallel to collective intelligence, standardizing the source data under various types of probability curves reflects the perception of a culture (collective intelligence) endowed with various a priori distinctions as for the relative importance of events. Here comes an important difference between the application of neural networks to simulate collective intelligence, on the one hand, and their application for optimizing the execution of specific tasks. In the latter case, the neural network should simply contain a sub-algorithm such as Gaussian Mixture, and experiment with indefinitely many Gaussian distributions for standardizing source data. Yet, in the here-presented method observing the process of learning is more important than optimizing the outcomes of learning.

Empirical data used for teaching the neural network is normally a table. The idea of teaching used in this method is that empirical data consists of a finite number of phenomenal occurrences, and each occurrence is an occasion for the network to learn something. Generally, empirical data in this method should be divided, along one of its dimensions, into successive, phenomenal occurrences, or simply lines in the database. The term 'successive' is theoretical: it can mean succession in time as such, yet it can also translate into any arbitrary order. This order depends essentially on the assumptions we make regarding the collective intelligence mimicked by the perceptron. The author used this method on macroeconomic data in pooled form, grouped into "country – year" observations. Each "country – year" observation is a phenomenal occurrence $o(j)$.

The perceptron starts learning with the first phenomenal occurrence $o(1)$ in the set Z. Two operations are being carried out: neural activation and observation of the fitness function. $V[z_i(j)]$ stands for local value of the fitness function V in variable x_i (or y), in the phenomenal occurrence j; $V[z_i(j)]$ is calculated as the mean local Euclidean distance of $z_i(j)$ from other variables in the given j-th phenomenal occurrence, as in equation (1) below.

$$V[z_i(j)] = \frac{\sum_{i=1}^{n}\sum_{k=1}^{n-1}\sqrt{[z_i(j)-z_k(j)]^2}}{n+1} \qquad (1)$$

V[Z(j)] is the general fitness function of the set ***Z*** in the ***j***-th phenomenal occurrence, calculated as the mean value of ***V[z_i(j)]*** , as in equation (2).

$$V[Z(j)] = \frac{\sum_{i=1}^{n} V[z_i(j)]}{n+1} \qquad (2)$$

Note that fitness functions can include various detailed assumptions. The here presented one is probably the simplest measure of coherence between variables. Additional assumptions can cover, for example, arbitrary or non-arbitrary weighing of individual Euclidean distances etc.

Neural activation occurs by feeding data ***Z(1***) into the neural activation function ***g(1)***, used in the perceptron for estimating the value of output variable ***y*** in the phenomenal occurrence ***1***. In the original application of this method, the author used the function of hyperbolic tangent, or $g(j) = \frac{e^{2h}-1}{e^{2h}+1}$. Note that other activation functions (i.e. other than hyperbolic tangent) can be used; the key is the meaningfulness of results yielded by the perceptron. The variable ***h*** in ***Z(1)*** is calculated as $h = \sum_{i=1}^{n} z_i * w$, where ***w*** is a random weight ***0 > w > 1***.

For each consecutive phenomenal occurrence used to teach the perceptron, local error of estimation is computed ***e_y(j)***, as in equation (3). The factor ***g'(j)*** is the first derivative of the activation function ***g(j)***. The idea behind adding it to the estimation of local error is that errors matter by their sheer magnitude as well as by the relative steepness of the neural activation function in the given phenomenal occurrence ***o(j)***.

$$e_y(j) = [g(j) - y] * g'(j) \quad (3)$$

The first round of learning with data, i.e. the perception, processing, estimation of coherence and that of accuracy yield two values: the vector of variable-specific local fitness functions ***V[Z(1)]***, and the local error ***e(1)***. The perceptron learns on its capacity to estimate the output variable 'y', and on the mutual coherence (Euclidean distance) of input variables. The underlying theoretical assumption is that collective intelligence attempts to achieve some outcomes and evaluates its own capacity to do it (that's why governments fall when they fail on key economic promises, for example), as well as its own coherence. The last assumption means that any culture learns and optimizes within a repertoire of moves coherent with the given set of social norms, e.g. changes in the healthcare system usually mean incremental change in public spending rather that brutal swing from 100% public funding to 100% private.

In the second round of learning, as well as in any consecutive one, thus when processing data ***Z(2)*** ≥ ***Z(j)*** ≥ ***Z(m)***, the logical structure changes slightly. The parameter h of the neural activation function g(j) incorporates both the error generated in the previous round of learning, and the values of local fitness functions in the same preceding round, as in equation (4). Besides incorporating the lesson from previous rounds, the perceptron keeps experimenting with random weights ***w(z_i)***, which, in turn, reflects the innovative component of collective intelligence.

$$h(2 \leq j \leq m) = \sum_{i=1}^{n}[z_i + e_y(j-1)] * w * V[z_i(j-1)] \qquad (4)$$

The procedure proposed in this method includes a formal check of intelligence, which, by the author's experience, is really a formality. The series of ***m*** general fitness functions ***V[Z(j)]***, as well as the series of ***m*** errors ***e(j)*** must both be non-monotonous. In other words, there must be demonstrable adjustment.

Assessment of results generated by the neural network

The set ***X*** of empirical observations, structured into ***m*** phenomenal occurrences and ***n*** variables, treated with the neural network, generates a transformed set ***S*** of values. Technically, occurrence 1 in the set ***S*** is identical to that in the set ***X***, and the remaining '***m – 1'*** occurrences are different. With the same logical structure of the perceptron, the set ***X*** can generate as many different sets ***S***, as there are variables, thus '***n***'. In each mutation of the set ***X***, another variable from among ***n*** is taken as desired output of the neural network. The set ***X*** can therefore generate ***n*** alternative sets ***S***. Note that the exact empirical look of each set S_i is probabilistic: each time we run the perceptron over ***m*** phenomenal occurrences, it produces a slightly different set S_i of values. Each set S_i can be compared as for its similarity to the original set ***X***. Many possible procedures of comparison are possible, and once more, this method goes for simplicity, and consistently with the logical structure of the perceptron, Euclidean distance has been chosen to gauge similarity.

In the set ***X***, each variable x_i yields a mean value $avg(X;x_i)$ over ***m*** phenomenal occurrences. In the same manner, each variable x_i in each set S_i is endowed with a mean value $avg(S_i;x_i)$. Both the mean values value $avg(X;x_i)$ and the value $avg(S_i;x_i)$ are vectors, characteristic for their respective sets. Each set S_i can be compared to the set ***X*** as for the Euclidean distance $E(X;S_i)$ between these vectors, as in equation (5). Among the ***n*** sets S_i generated from the set ***X***, the set S_i endowed with the smaller Euclidean distance $min[E(X; S_i)]$ is the most similar to ***X***. Consequently, we can assume that the output variable of this specific set S_i is the most likely value optimized by the society represented in the original empirical set ***X***. Other sets Si, endowed with higher $E(X;S_i)$ are, respectively, less and less likely representations of values pursued by the society studied.

$$E(X;S_i) = \sqrt{\sum_{i=1}^{n}\left[\frac{avg(S_i;x_i)}{avg(X;x_i)}\right]^2} \qquad (5)$$

The neural network in the form of an Excel spreadsheet: case study with data regarding energy efficiency

The present method includes structuring the source data in set X into an Excel spreadsheet, and using the spreadsheet to emulate a neural network. The interest of using Excel is strictly scientific: whilst being a lot slower than algorithm written in languages like Python, especially with large datasets, an Excel spreadsheet allows observing hands-on, one round after another, the learning process of the neural network. Besides, the spreadsheet, such as discussed here and downloadable from the links provided in the "Resource availability" section, allows graphical insight into the learning process, due to the flat, two-dimensional structure of the tables, as well as via self-updating graphs.

The Excel spreadsheet adjacent to this methodological paper provide empirical data used in the article "*Energy efficiency as manifestation of collective intelligence in human societies*" submitted with the journal "*Energy*". The general structure of the spreadsheet covers 4 tables and 2 graphs. The tables are: **Source Data, Standardized, Perceptron, Notes**. Graphs show the distribution of local error ***e(j)*** over the given series of phenomenal occurrences, and the distribution of general fitness function ***V[Z(j)]***. The '**Source Data**' table contains the set ***X*** of source empirical data. In this empirical case, the set X contains 12 variables: : i) GDP per kg of oil equivalent in energy consumed (the outcome variable), ii) fixed assets per one patent application, iii) aggregate depreciation of fixed assets as % of the GDP, iv) number of resident

patent applications per 1 million inhabitants, v) supply of broad money as % of the GDP, vi) energy use per capita, vii) depth of food deficit per capita, viii) % of renewables in the total consumption of energy, ix) % of urban population in total population, x) GDP and xi) population as scale factors, and xii) GDP per capita. Conformingly with the methodology presented in previous sections, this structure implies 12 versions of the neural network, and the creation of 12 sets $S_i(i: 1, 2,\ldots, 12)$. Each version of the neural network assumes that one among the 12 variables is the output one, or '***y***', and the remaining ones are input information $x_i(i:1, 2, \ldots, 11)$.

As the digital vehicle is Excel, empty cells in the '**Source Data**' table are to avoid. Thus, the set ***X*** presented in the Excel file adjacent to this paper is filtered for records containing empty observations. In the very bottom of the 'Source Data' table the reader can find a few lines, marked: ***Maximum, Average, and Quartiles.*** These are useful, in the first place, for the standardization of source data, and for further assessment of Euclidean distance between the source set ***X***, and the derived sets S_i.

The table labelled '***Standardized***' contains the set ***Z*** of data standardized for being processed with the neural network. The method of standardization used in the Excel file linked to in the 'Resource availability' section standardizes the source data in the simplest possible way, i.e. linearly over local maxima. The cell C2 in the 'Standardized' table contains the formula: **='Source data'!C2/'Source data'!C$1231**, where the address '**Source data'!C2** corresponds to the source value $x_i(1)$ – variable 'GDP per unit of energy use (constant 2011 PPP $ per kg of oil equivalent)', phenomenal occurrence 1, which is India in 1990, in the 'Source data' table, and **'Source data'!C$1231** is the local maximum of the same variable. All the other cells in the 'Standardized' table reproduce the same logic.

The '**Perceptron**' table contains the logical structure of the neural network strictly spoken. Note that lines in this table are moved one step down, as compared to lines in tables 'Source Data', and 'Standardized'. Line 4 in 'Perceptron' is functionally connected to line 3 in 'Source Data' and 'Standardized' and so forth. **The first two columns, A and B**, reproduce the labelling of phenomenal occurrences $o(j)$ from the 'Source Data' table. In this specific Excel file, they name the 'country – year' observations for research on energy efficiency. This labelling of lines can be important as far as the user is interested in the exact order of phenomenal occurrences, as well as in observing the behaviour of the neural network in a precise point $x_i[o(j)]$.

The logical structure of the neural network is reflected in the functional assignment of columns. In other words, the table 'Perceptron' is divided into sections of columns, each bearing a different label, and each corresponding to a different neuron (or layer of neurons) in the perceptron. **Columns C ÷ N of the 'Perceptron'** table are labelled '**Initial values**'. Column **C** contains the output variable, whilst **D ÷ N** cover the input variables. Consequently, The cell C3 contains the formula: =Standardized!C2, cell C4 refers "=Standardized!C3" etc. The first line in column D spells "=Standardized!D3", but in the next lines, this precise input variable – "Fixed assets per 1 resident patent application" – is being fed with error $e(j-1)$, from the preceding round of learning, i.e. the preceding phenomenal occurrence $o(j-1)$. Error is fed from **column AI**, whose exact logical connection is described further in this section. Cell D4 refers "=Standardized!D3+$AI3", cell D5 is "=Standardized!D4+$AI4" etc. The remaining columns of the D ÷ N section follow the same logic: in lines 4 ÷ 1230 they sum standardized values z_i from the 'Standardized' table with the error $e(j-1)$ from the column AI.

The first line of these values (line #3 in this precise Excel file) contains the standardized values in the first phenomenal occurrence $o(1)$ in the set ***Z***. Cells D3 ÷ N3 (column-wise), reproduce the same connection to

the ‘Standardized’ table. Further lines in columns C ÷ N are split in two categories: standardized output variable in column C, and standardized values of input variables, fed with error ***e(j)***, in columns ***D*** ÷ ***N***.

The next group of columns in the ‘**Perceptron**’ table, which the reader should get interested in, are **columns AJ ÷ AU**. Whilst being pushed to the very right end of the table, they become important quite early, as they contain local fitness functions ***V[zᵢ(j)]*****,** conformingly to equation (1). Below, the logical structure of these columns is exemplified for the output variable, and for one input variable. As the reader will study them, please bear in mind that the final denominator, ‘/12’ in this case, depends on the number of variables in the perceptron.

Formulation of the local fitness function *V[zᵢ(j)]* for the output variable: Energy Efficiency: GDP per unit of energy use (constant 2011 PPP $ per kg of oil equivalent)

- **cell AJ3**: =(C3+((C3-D3)^2)^0,5+((C3-E3)^2)^0,5+((C3-F3)^2)^0,5+((C3-G3)^2)^0,5+((C3-H3)^2)^0,5+((C3-I3)^2)^0,5+((C3-J3)^2)^0,5+((C3-K3)^2)^0,5+((C3-L3)^2)^0,5+((C3-M3)^2)^0,5+((C3-N3)^2)^0,5)/12

- **cell AJ4**: =(C4+((C4-D4)^2)^0,5+((C4-E4)^2)^0,5+((C4-F4)^2)^0,5+((C4-G4)^2)^0,5+((C4-H4)^2)^0,5+((C4-I4)^2)^0,5+((C4-J4)^2)^0,5+((C4-K4)^2)^0,5+((C4-L4)^2)^0,5+((C4-M4)^2)^0,5+((C4-N4)^2)^0,5)/12

Formulation of the local fitness function *V[zᵢ(j)]* for an input variable: e.g. Fixed assets per 1 resident patent application

- **cell AK3:** =(D3+((D3-C3)^2)^0,5+((D3-E3)^2)^0,5+((D3-F3)^2)^0,5+((D3-G3)^2)^0,5+((D3-H3)^2)^0,5+((D3-I3)^2)^0,5+((D3-J3)^2)^0,5+((D3-K3)^2)^0,5+((D3-L3)^2)^0,5+((D3-M3)^2)^0,5+((D3-N3)^2)^0,5)/12

- **cell AK4:** =(D4+((D4-C4)^2)^0,5+((D4-E4)^2)^0,5+((D4-F4)^2)^0,5+((D4-G4)^2)^0,5+((D4-H4)^2)^0,5+((D4-I4)^2)^0,5+((D4-J4)^2)^0,5+((D4-K4)^2)^0,5+((D4-L4)^2)^0,5+((D4-M4)^2)^0,5+((D4-N4)^2)^0,5)/12

Column AV produces the general fitness function ***V[Z(j)]*** as in equation (2), simply by drawing an average from columns AJ ÷ AU, e.g. **cell AV3**: = AVERAGE(AJ3:AU3), cell AV4 = AVERAGE(AJ4 :AU4) etc.

Now, as the reader has got at least somehow familiar with the way of handling the fitness function, or equations (1) and (2), in the ‘**Perceptron**’ table of the spreadsheet, it is time to move back to the left side, and to see the way of generating the neural activation function. The first step is to generate the vector ‘***h***’, the exponent of activation function, which, as the reader probably remembers, is generated differently in the first phenomenal occurrence on the one hand, from those generated in further ones. Vector ***h*** is computed in **columns O ÷ AA** of the table. The general philosophy of structuring this table is to make the process rather piecemeal, in order to be able to follow small local changes. Hence, **columns O ÷ Z**, grouped in the section labelled ‘*Values weighed with random and V’*, contain local components of the vector h, corresponding to individual variables. Column AA of the table sums up all the local components of the vector h, and here comes a little trick used by the author. The ‘Perceptron’ table should be mutable into as many versions as there are variables in the set ***X***, in order to simulate the way this intelligent structure optimizes each of those variables, alternatively, as the output variable. Computation of the vector ***h*** is part of such mutation. Thus,

cell AA3 in column AA is formulated as: ***=SUM(O3:Z3)-O3***. In this specific version of the spreadsheet, the perceptron optimizes energy efficiency as output variable. The component, weighed version of this variable is to find in column O, and, consequently, value in this column is to eliminate from the computation of the vector h: this is the '-O3' component in the formula to find in cell AA3.

In **the first row of the perceptron**, each of the component values for computing the vector ***h*** is formulated as: **=RAND()*{address of the cell with standardized values in the "Initial values" section in columns C ÷ N}**. The 'RAND()' factor is, of course, the generator of pseudo-random values, which corresponds to the random weight '***w***' in the theoretical structure of the neural network.

In each further row of the perceptron, thus in phenomenal occurrences $2 \leq j \leq 1228$ in this precise case, cells in columns C ÷ N are fed with local fitness functions of the corresponding variables, observed the previous phenomenal occurrence. Thus, **cell O4** spells: =RAND()*C4*AJ3, **cell P4** is: =RAND()*D4*AK3 etc. References 'AJ3', and 'AK3' connect to cells with local fitness functions from the preceding round.

Columns AB – AG in the 'Perceptron' table generate the **functions of neural activation**. The plural 'functions' refers to the fact that in this precise structure of the spreadsheet, two classical activation functions have been dropped into the toolbox: the sigmoid in **columns AB ÷ AD**, and the hyperbolic tangent in **columns AE ÷ AG**. The theoretical structure of the neural network, as described in the preceding section of this paper, uses the hyperbolic tangent, still the sigmoid can be useful in some simulations. In the author's experience, the sigmoid function, expressed as $g(j) = \frac{1}{1+e^{-h}}$, where '***h***' is the same vector ***h*** as discussed previously, has the capacity to level out sudden surges in the input data. Speaking in metaphorically psychological terms, the sigmoid calms down the reaction to surprising states of nature, whilst the hyperbolic tangent corresponds to a much less filtered neural reaction. That 'calming' property of the sigmoid makes it learn faster, as it drives the residual error asymptotically down to zero at a noticeably faster pace. Still, in neural networks it is just as in real life: faster learning does not necessarily mean more accurate a learning. The sigmoid is prone to the so-called 'overfitting', or too hasty a conclusion from the data at hand. As the overall logical structure is applied to social systems, the hyperbolic tangent corresponds to weak (or even non-existent) assumptions as for counter-cyclical economic policies. On the other hand, the sigmoid can be useful in simulating social systems with some economic filters on, such as significant automatic stabilizers in fiscal policy etc.

Each of the two neural functions in the 'Perceptron' table is spread into three distinct columns: the first column computes the value of the activation function ***g(j)*** as such, the second one computes its first derivative ***g'(j)***, and the third one computes the local error ***e(j)*** as in equation (3). The 'Perceptron' table contains additional features that allow steering the process of feeding the error e(j) forward into consecutive phenomenal occurrences. These features are to find in **columns AH and AI**. As it was stated previously, error is fed from cells in **column AI**, and yet this column can be used as an autonomous layer in the neural network. In the basic version of the perceptron, such as outlined in the theoretical section of this paper, the error to feed forward is that generated by the hyperbolic tangent, thus by cells in columns **AE ÷ AG**. Thus, in this baseline case, **cell AI3** spells simply '=AG3' etc. Still, mutations are possible: errors from the two neural functions – the sigmoid and the hyperbolic tangent – can be put in competition or in congruence, by averaging them, weighted-averaging etc. In such cases, cells in column AI can take on formulas like '=AG#*0,5+AD#*0,5'.

Column AH contains something that the author designates as '**coefficient of memory**'. This is an instrumental extension out of the theoretical model presented in the previous section. It is simply the ordinal

number of the given phenomenal occurrence, and it refers theoretically to the theory of social games with imperfect recall such as discussed by Reinhard Selten (Selten 1990). The baseline assumption of the here-presented method is that each consecutive phenomenal occurrence in the neural network incorporates all the previous learning, on the preceding errors and observed fitness functions. Once again, theoretical branching is possible. Societies can be forgetful, which means that new learning is more important than old learning. Analytically, it means that error ***e(j)*** to propagate from cells in column AI can be divided by '*1/{coefficient of memory in column AH}*. Conversely, societies can be conservative, which implies lesser importance attached to new learning, as opposed to old learning. In such case, error e(j) gets multiplied by '*1/{coefficient of memory in column AH}*.

After the table 'Perceptron' in the spreadsheet, the reader will find **the table** '**Notes**', which is quite simple. In its baseline look, **columns B ÷ M** contain the average values from **columns C ÷ N in the table 'Perceptron'**. In the case of the specific dataset that serves to exemplify the logical structure at hand, **cell B7** in the table 'Notes' refers: =AVERAGE(Perceptron!C$3:C$1230), **cell C7** spells: =AVERAGE(Perceptron!D$3:D$1230) etc. The values yielded by these formulas are technically standardized, i.e. they are more akin to the standardized variables z_i from the set $\boldsymbol{Z}$, than to variables x_i from the original set $\boldsymbol{X}$. Thus, we need to reverse-de-standardize them, and this operation is carried in the line 10 of the '**Notes**' table. **Cell B10**, for example, refers: ***=B7*'Source data'!C1231***, thus it takes the mean value calculated in cell B7, and multiplies it by the corresponding maximum in the '**Source data**' table, thus in the original set $\boldsymbol{X}$. Note that the de-standardized mean for the output variable will be identical to the original mean in the X set, as the perceptron is supposed to be pegged on keeping that value as the target. De-standardized means from line 10 can be directly fed into equation (5), in order to compute the corresponding Euclidean similarity with the original set $\boldsymbol{X}$.

The last issue to discuss as regards the Excel spreadsheet used to emulate a neural network is the way of building its many mutations, or multiple sets $\boldsymbol{S_i}$, defined by the exact variable from the set X pegged as the output one. The output variable has three main characteristics in the '**Perceptron**' table. First of all, in the section 'Initial values', i.e. in columns C ÷ N of 'Perceptron', the output variable is not fed with error ***e(j-1)***. Thus, each line in the column corresponding to the output variable simply refers to its sibling in the 'Standardized' table, whilst input variables in the remaining columns in the section C ÷ N are fed with the local error. Second of all, the output variable is subtracted from the sum total of weighted values, calculated individually in columns O ÷ Z, and then summed in column AA. Thus, the output variable is the '-{cell reference}' component in the formula to be found in the cells of column AA. In the third place, the output variable serves to calculate the local error e(j) in columns AD and AG of the 'Perceptron' table, respectively for the sigmoid function, and the hyperbolic tangent. Modification of these three features in the 'Perceptron' table reorients the neural network on optimizing another variable as the output one, and thus creates an alternative set $\boldsymbol{S_i}$.

Acknowledgements:
This research has been conducted with funding provided by the Ministry of Science and Higher Education of the Republic of Poland, under the research grant no. WZiKS/DS/7/2018-KON

Supplementary material *and/or* Additional information:
List of links to accessory Excel files with mutations of the neural network, based on linear standardization over local maxima:

1) Output variable: GDP per kg of oil equivalent in energy consumed (the outcome variable) >> https://discoversocialsciences.com/wp-content/uploads/2019/09/Perceptron-with-big-database_pegged-energy-efficiency_educational.xlsx

2) Output variable: fixed assets per one patent application >> https://discoversocialsciences.com/wp-content/uploads/2019/09/Perceptron-with-big-database_pegged-fixed-assets-per-patapp.xlsx

3) Output variable: aggregate depreciation of fixed assets as % of the GDP >> https://discoversocialsciences.com/wp-content/uploads/2019/09/Perceptron-with-big-database_pegged-depreciation-in-GDP.xlsx

4) Output variable: number of resident patent applications per 1 million inhabitants >> https://discoversocialsciences.com/wp-content/uploads/2019/09/Perceptron-with-big-database_pegged-patent-applications.xlsx

5) Output variable: supply of broad money as % of the GDP >> https://discoversocialsciences.com/wp-content/uploads/2019/09/Perceptron-with-big-database_pegged-supply-of-money.xlsx

6) Output variable: energy use per capita >> https://discoversocialsciences.com/wp-content/uploads/2019/09/Perceptron-with-big-database_pegged-energy-per-capita.xlsx

7) Output variable: depth of food deficit per capita >> https://discoversocialsciences.com/wp-content/uploads/2019/09/Perceptron-with-big-database_pegged-food-deficit.xlsx

8) Output variable: % of renewables in the total consumption of energy >> https://discoversocialsciences.com/wp-content/uploads/2019/09/Perceptron-with-big-database_pegged-percentage-renewables.xlsx

9) Output variable: % of urban population in total population >> https://discoversocialsciences.com/wp-content/uploads/2019/09/Perceptron-with-big-database_pegged-percentage-urban-population.xlsx

10) Output variable: GDP >> https://discoversocialsciences.com/wp-content/uploads/2019/09/Perceptron-with-big-database_pegged-GDP.xlsx

11) Output variable: population >> https://discoversocialsciences.com/wp-content/uploads/2019/09/Perceptron-with-big-database_pegged-population.xlsx

12) Output variable: GDP per capita >> https://discoversocialsciences.com/wp-content/uploads/2019/09/Perceptron-with-big-database_pegged-GDP-per-capita.xlsx

[OPTIONAL. We also give you the option to submit both supplementary material and additional information. Supplementary material relates directly to the work that you have submitted and can include extensive excel tables, raw data etc. We would also encourage you to include failed methods or describe adjustments to your methods that did not work. Additional information can include anything else that is not directly related to your method, e.g. more general background information, useful links etc. Introduction is not a section included in the MethodsX format. This information could be moved to the end under Additional Information.

***References:** *[Include at least one reference, to the original publication of the method you customized.]*