Software Impacts 10 (2021) 100121

Contents lists available at ScienceDirect

Software Impacts

journal homepage: www.journals.elsevier.com/software-impacts

Original software publication

# Gradient descent training expert system

Jeremy Straub

*Department of Computer Science, North Dakota State University, 1320 Albrecht Blvd., Room 258; Fargo, ND 58108, United States of America*

ARTICLE INFO

*Keywords:*
Machine learning
Expert system
Training
Gradient descent
Rule-fact network

ABSTRACT

This software is used to build, train and present data for evaluation by a gradient descent training expert system (GDES). A GDES uses a machine learning training method, gradient descent, in a manner similar to a neural network; however, instead of a multi-layer network of densely connected nodes, it uses a known-meaning rule-fact network. Thus, the logical relationships (rules) between nodes (facts) are human (or, potentially in the future, autonomously) defined and can have an identified meaning; however, their weightings are optimized using machine learning techniques. This provides the explainability of an expert system with the optimization of a neural network.

## Code metadata

| | |
|---|---|
| Current Code version | *v1.0* |
| Permanent link to code/repository used of this code version | https://github.com/SoftwareImpacts/SIMPAC-2021-75 |
| Permanent link to Reproducible Capsule | |
| Legal Code License | *Apache License, 2.0* |
| Code Versioning system used | *none* |
| Software Code Language used | *C#* |
| Compilation requirements, Operating environments & dependencies | *Visual Studio 2019* |
| If available Link to developer documentation/manual | |
| Support email for questions | jeremy.straub@ndsu.edu |

## Software metadata

| | |
|---|---|
| Current software version | *v1.0* |
| Permanent link to executables of this version | https://github.com/jeremystraub/GDES/blob/main/Executable.zip |
| Permanent link to Reproducible Capsule | |
| Legal Software License | *Apache License, 2.0* |
| Computing platform/Operating System | *Microsoft Windows and other environments that support the .Net framework* |
| Installation requirements & dependencies | *Dot-Net Framework* |
| If available Link to user manual — if formally published include a reference to the publication in the reference list | |
| Support email for questions | jeremy.straub@ndsu.edu |

## 1. Introduction

Artificial intelligence and machine learning systems have found use in numerous areas of society. They are used for applicant screening [1], making medical recommendations [2] and even making sentencing recommendations for those convicted of crimes [3–5]. However, despite – or perhaps due to – the multitude of areas that these systems are used in, the public is concerned about them [6]. One area of pronounced concern is the systems' potential "dark side" [7] that may result in biased decisions which disenfranchise those from racial minorities, the poor and others. Concerns have been raised that training processes may cause systems to learn and then reinforce dominant power structures [8]. Due to these concerns, Noble has aptly termed such systems as "algorithms of oppression" [9]. Discriminatory decision making is only one example of the type of issues that may occur with machine learning systems. More generally, they may make spurious connections between non-causal data correlations that can lead to extreme failure with certain data, while the system seems (and can be shown via testing) to be working well overall. A sub-field of machine

*E-mail address:* jeremy.straub@ndsu.edu.

https://doi.org/10.1016/j.simpa.2021.100121
Received 7 July 2021; Received in revised form 18 August 2021; Accepted 24 August 2021

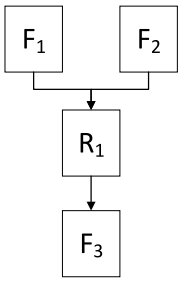


**Fig. 1.** Basic Rule-Fact Network Structure.

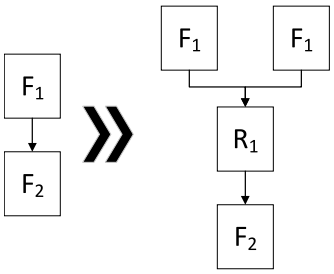


**Fig. 2.** Rule setting one fact to the value of another.

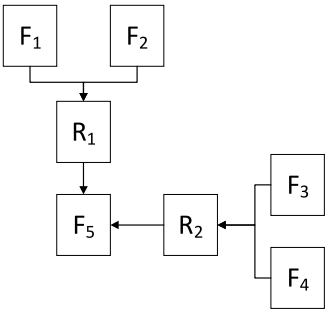


**Fig. 3.** Multiple rules setting a fact's value.

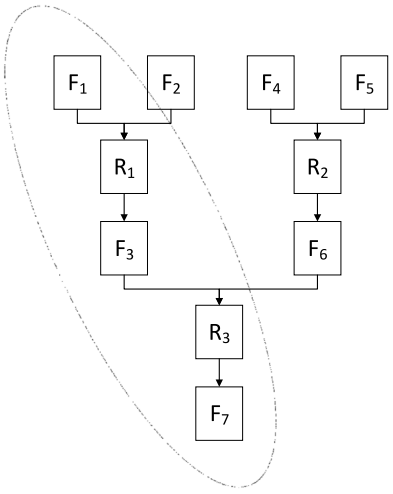


**Fig. 4.** Training network.

learning, eXplainable Artificial Intelligence (XAI), has developed due to the limitations of AI systems' to be able to "explain their autonomous decisions to human users" and lack of human understandability [10].

While a variety of XAI techniques have been proposed [11], the goal of explainability stops short of ideal. In [12], a "defensible" system was proposed which utilizes an expert system (see, e.g., [13]) rule-fact network as the basis for knowledge storage and backpropagation (see [14]) training to optimize the rule weighting values. Hence, it provides the benefit of known-meaning nodes (facts) and associations (rules) while also providing the optimization benefits of machine learning and neural network systems.

This software article discusses a version of this system which has been modified to be more user friendly and to process data for user applications. Specifically, it provides the requisite functionality to build the rule-fact network, load data into it, train the network (i.e., optimize rule weightings) and present data for evaluation. It, thus, facilitates the use of GDESes for relevant applications and additional analysis of the technique.

## 2. Software description

The software operates in three phases: first, a rule-fact network is defined; second, the system is trained; finally, data is loaded into facts to prepare for the presentation of data and data is presented for evaluation. The same approach can be used whether using the system to support research or operations in a particular application domain or if testing the system itself.

In the first phase, a rule-fact network is defined. The basic structure of this network is shown in Fig. 1. The system, currently, requires all rules to have exactly two input facts. Facts must have positive values between 0 and 1 (inclusive) and rules include a weighting for each fact between 0 and 1 with the two weightings for each rule summing to 1.

With this basic structure, a variety of more complex structures can be implemented. For example, Fig. 2 shows how a rule might be created to have one fact's value set another fact's value, if this was a requirement of the system. Similarly, multiple rules can be used to implement a concept that requires more than two input facts to model an association.

The system also supports network designs that have more than one pathway to set a fact's value. Fig. 3 shows how a fact could have its value set by two rules considering multiple different input facts. Whatever value of a fact is set last is its current value. Different pathways could, depending on application needs, produce conflicting values for a fact, if desired.

In the second phase, the system is trained using a supervised training process by presenting fact inputs and desired outputs. One or more facts values can be supplied as inputs (at least one must be supplied in all cases and is included in the command that triggers the training process) and a target fact is defined as the output. The desired value of this target fact is also provided to the training system. Multiple iterations of training are conducted with different inputs and outputs. During each iteration, the weighting values of the rules are updated (the rate at which they are updated is determined by a user-specified velocity value). These weightings are not reset between training iterations (or before the presentation of data for operations or testing). Notably, while the system's results are shaped by the training, the initial network design also has a significant impact on system results. More details about the training process can be found in [12] and in Section 3.

The system supports training across an entire pathway of a network, as it will be presented for operations or testing, as shown in Fig. 4. Alternately, different areas of the network can be individually trained, as shown in Fig. 5.

Finally, during the third phase, data is presented for processing by the system. One or more facts have their values set and then the system is launched to determine the value of a target output node. Notably, the resulting values of other facts can be accessed with the query fact command, if the result of operations on more than one fact needs to be known.

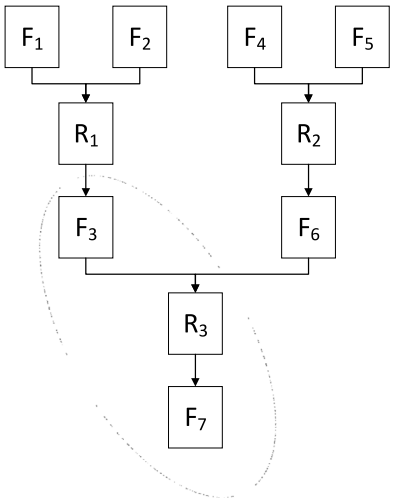


**Fig. 5.** Training subset of network.

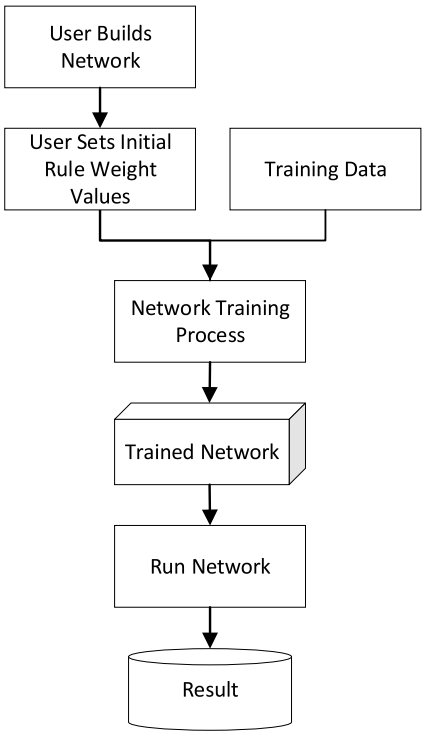


**Fig. 6.** Overview of system operations.

While these three phases will typically be run in the order described, this is not the only way the system can be used. For example, the network could be built and trained. Then data could be presented for evaluation. This could be followed by additional training and additional presentation of data for evaluation. Obviously, a network must be created, in all cases, before training or presentation for evaluation can occur. The network can be augmented, if desired, after training and data presentation have been conducted.

A list of commands for the system is presented, along with the command format (please see the referenced manual for full details on each command), in Table 1.

## 3. Algorithm

The software described herein implements the algorithms described in [12]. The system is comprised of an expert system engine, which processes rules in a forward fashion, and a training module, which is used to optimize the rule weightings. The process, shown in Fig. 6, starts with the user supplying their network design and initial rule weight values. This network (including the rule weightings) and the training data are used for training. The trained network and inputs for specific operational or test cases are then used to generate results.

The expert system that has been implemented supports partial membership and ambiguity. In this system, facts can have a probabilistic or partial membership value ranging between 0 and 1. Because of this, rules utilize weighting values which indicate the comparative impact of the input facts on the value of the output fact. Like fact values, rule weightings must be between 0 and 1 and must sum to 1.

The training process is depicted in Figs. 7 and 8. Fig. 7 depicts the training process overall while Fig. 8 depicts how the level of change applied to each rule weighting is determined. The training process is, at a high level, very similar to that of a neural network. The system under training is run with the inputs specified by a data record and the output result is compared to the result produced by the system. A portion of the difference, determined by the user-specified velocity value, is applied to the rule weightings. Multiple training epochs can be performed for each training data record, as specified by the user. This process, shown for a single training data record in Fig. 7, continues until all training data is exhausted.

A key way that the software described herein (and the system described in [12]) differs from a neural network is the process that is used to determine the amount of change that is applied to each rule during each training epoch. The algorithm for this is depicted in Fig. 8.

The process starts by identifying all of the nodes that directly impact the target fact (i.e., the fact that the training data supplies the target result for) and the system determines the level of their contribution. Once these initial nodes that directly impact the target fact have been identified, they are added to a contributions list. Then, all nodes that impact the target fact indirectly are iteratively identified, their indirect contributions are calculated and they are added to the contributions list. Note that contribution to all other applicable contributions list nodes is determined for each node that is added. The node adding process continues until an iteration runs without adding nodes.

Each rule's contribution is either direct or determined by multiplying rules that pass through other rules for impact by the cumulative impact of the intervening rules. The contribution of a rule, Ci, to the target fact can be determined using the equation [12]:

$$C_i = W_i \times \prod_{\{APT\}} W_{R(m,h)} \tag{1}$$

For this equation, $W_i$ is the relevant weighting for the given rule (i). $W_{R(m,h)}$ is the weighting of each rule (m indicates the rule and h indicates the relevant weight value) which the value passes through before the final fact. This rule set is denoted by {APT}. Notably, rules may have multiple contribution ($C_i$) values if they are part of multiple paths; however, only the highest contribution value is maintained and used.

A velocity setting value determined percentage of the error between the system output target fact value and training output value is applied to each contributing rule's weightings based on its contribution level and facts' values. Additional or reduced weight is given to the higher and lower values' input fact weightings, depending on the type of change that is needed.

The change that will be made to a particular rule is determined by summing all contributions to a target fact and dividing the contribution of a particular rule ($C_i$) by the contributions sum ($C_{Total}$) and applying the velocity (V) and $\Delta R$ value, which is the difference between the expected (training data value) and actual value for a training run. The difference value ($D_i$) is computed using the equation [12]:

$$D_i = \frac{C_i}{C_{Total}} \times V \times \Delta_R \tag{2}$$

**Table 1**
System commands and format.

| Command type | Command format |
|---|---|
| Create fact | F####:{FGUID}=000.000:Description (VAR) |
| Create rule | R####:{R1GUID}:{F1GUID}=0.000+{F2GUID}=0.000>>{F3GUID}:Description (VAR) |
| Train | TR:{FGUID}=000.000>####:0.00>{FGUID}=000.000 |
| Present for evaluation | PR:{FGUID}=000.000>>{FGUID} |
| Set fact | SF: {FGUID}=000.000 |
| Query fact | QF: {FGUID} |

Note that # and 0 indicate numeric values, {GUID} indicates a .Net format globally unique identifier (GUID) and "Description (VAR)" indicates a variable length description field. All other symbols must be used exactly as presented in the rule format.

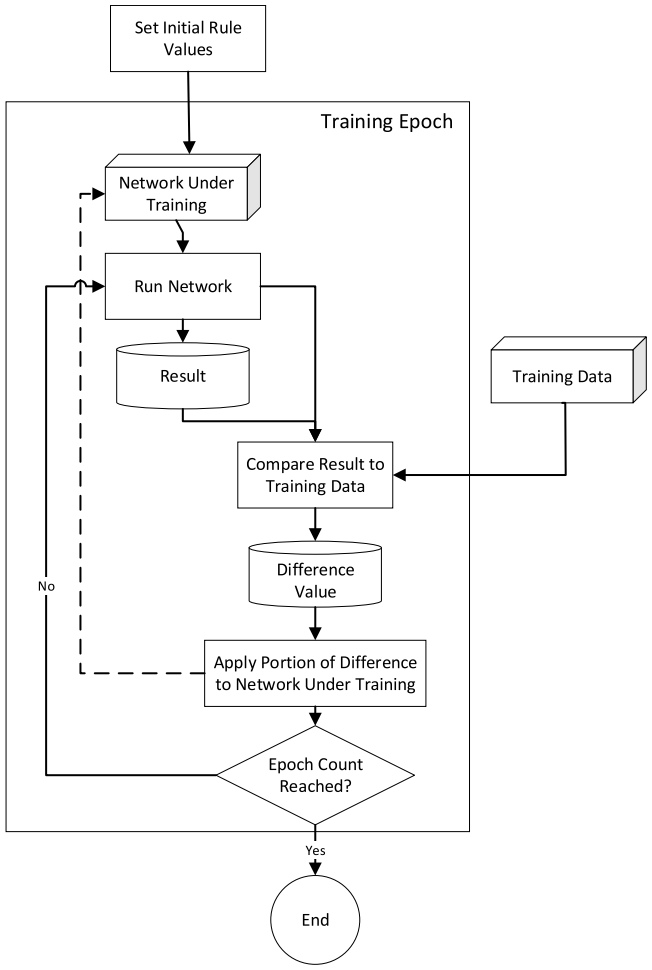


**Fig. 7.** Training Process. *Source:* Modified from [12].

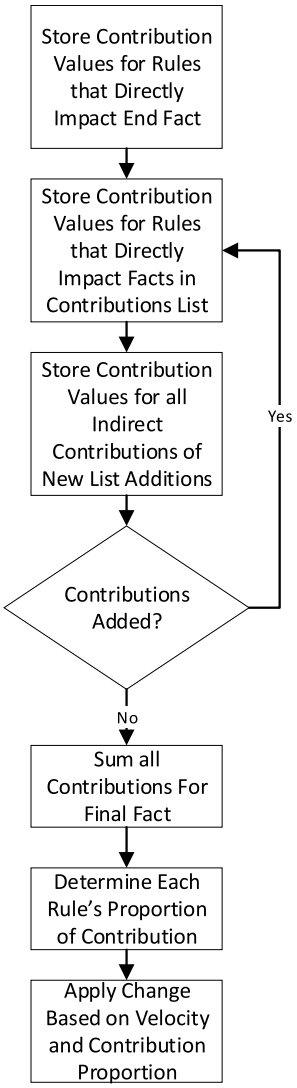


**Fig. 8.** Node Change Algorithm [12].

It uses the $\Delta$R value that is calculated using the equation [12]:

$$\Delta_R = \frac{|R_P - R_T|}{MAX(R_P, R_T)} \tag{3}$$

For this equation, $R_P$ is the target value and $R_T$ is the value produced by the network under training. The MAX function returns the largest value passed into it.

## 4. Advantages and limitations of the approach

The software presented herein provides key advantages as compared to both classical expert systems and neural networks. As compared to classical expert systems with fractional value support, it provides a mechanism to automate the optimization of rules. This allows the system to better reflect the real-world phenomena that it is modeling and to provide better recommendations, decisions or predictions.

As compared to neural networks, the system provides two key benefits. First, by utilizing a known-meaning network, network sizes will inherently be smaller than the densely connected networks used by neural network systems. This will reduce the number of nodes that must have their values computed and set during each training epoch and thus have training time benefits. Second, because all nodes (facts) and relationships (rules) have a known value, the decisions that the system makes are inherently human understandable and can be easily explained in terms of the decision-making factors that were considered and the relative importance given to each. This allows decisions to

be logically defended, as opposed to being opaque to users or simply explained in terms of system state and processes. This is directly responsive to the issues of public concern [6] and decision making bias [7–9] which are discussed in Section 1.

The principal limitation of the proposed approach is that a network must be created to reflect the phenomena being modeled. This, currently, is a potentially time-consuming manual process. Additionally, this manual creation process requires the phenomena to be well understood. Neural networks and some other machine learning technologies provide a key benefit, which the software described herein – in its current form – lacks: being able to model a phenomena that is not fully understood based solely on input data and results.

## 5. Use and impacts

This software is based on the software that was used for [12] with modifications to facilitate its more general use. Specifically, the network generation and characterization routines which were developed specifically for and enabled the experimentation in [12] are not included and have been replaced with a command format and processor for building rule-fact networks, training them and presenting facts for processing. The system is currently being used for two e-laws projects (focusing on U.S. federal sentencing guidelines and patentability assessment), an intentionally deceptive online content identification project and a phishing link identification project.

The federal sentencing guidelines project [15] is using the software to develop an application which will, based on key facts of a criminal offense (or set of related offenses), recommend a sentence for the offender. It is being developed based on federal sentencing guideline rules [16] and trained using data from the federal sentencing guidelines commission [17]. The proposed system will provide an entirely transparent and human-understandable sentence determination process, taking into account typical applications of judicial sentencing discretion within the sentencing guidelines.

The patentability assessment project [15] is using the software to develop an application which will be provided patent application details and make a recommendation regarding several key characteristics of patentability. It is being developed based on the U.S. Patent and Trademark Office's (USPTO) Patent Examiner's Handbook [18] and being trained using details from the USPTO's Patent Application Information Retrieval (PAIR) system and a USPTO curated dataset [19,20]. The proposed system aims to provide greater patentability decision consistency and increase patent assessment processing speed.

The intentionally deceptive online content identification and phishing link identification projects [21] are using existing data sets initially collected for prior projects using neural networks. Both systems aim to use a smaller network size (and, thus, require reduced training time) to produce results which are similar to the prior neural network implementations. In both cases, the opaque decision-making criteria of the neural network will be replaced by a completely transparent and human-understandable decision-making process.

## Declaration of competing interest

The author declares that he has no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgments

Thanks are given to researchers on the four projects that are currently using this software for feedback on their system needs and system functionality. This work has been supported, in part, by the U.S. National Science Foundation (NSF Award # 1757659).

## Appendix A. Supplementary material

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.simpa.2021.100121. Supplemental material includes a user maual and example scripts.

## References

[1] D.K. Malhotra, K. Malhotra, R. Malhotra, Evaluating Consumer Loans using Machine Learning Techniques, Emerald Publishing Limited, 2020, http://dx.doi.org/10.1108/s0276-897620200000020004.
[2] X. Zhou, Y. Li, W. Liang, Cnn-rnn based intelligent recommendation for online medical pre-diagnosis support, IEEE/ACM Trans. Comput. Biol. Bioinform. 1 (2020) http://dx.doi.org/10.1109/tcbb.2020.2994780.
[3] A. Deeks, The judicial demand for explainable artificial intelligence, Columbia Law Rev. 119 (2019) 1829–1850.
[4] N. Stobbs, D. Hunter, M. Bagaric, Can sentencing be enhanced by the use of artificial intelligence? Crim. Law J. 41 (2017) 261–277, https://eprints.qut.edu.au/115410/ (accessed February 24, 2021).
[5] V. Chiao, Predicting proportionality: The case for algorithmic sentencing, Crim. Justice Ethics 37 (2018) 238–261, http://dx.doi.org/10.1080/0731129X.2018.1552359.
[6] T. Araujo, N. Helberger, S. Kruikemeier, C.H. de Vreese, In AI we trust? Perceptions about automated decision-making by artificial intelligence, AI Soc. 35 (2020) 611–623, http://dx.doi.org/10.1007/s00146-019-00931-w.
[7] C. O'Neil, Weapons of Math Destruction, Broadway Books, New York, NY, USA, 2016.
[8] C. D'Ignazio, L.F. Klein, Data Feminism, MIT Press, Cambridge, MA, 2020.
[9] S.U. Noble, Algorithms of Oppression: How Search Engines Reinforce Racism Paperback, NYU Press, New York, NY, USA, 2018.
[10] D. Gunning, M. Stefik, J. Choi, T. Miller, S. Stumpf, G.Z. Yang, Xai-explainable artificial intelligence, Sci. Robot. 4 (2019) http://dx.doi.org/10.1126/scirobotics.aay7120.
[11] G. Vilone, L. Longo, Explainable artificial intelligence: A systematic review, 2020, ArXiv, http://arxiv.org/abs/2006.00093 (accessed April 27, 2021).
[12] J. Straub, Expert system gradient descent style training: Development of a defensible artificial intelligence technique, Knowl.-Based Syst. (2021) 107275, http://dx.doi.org/10.1016/j.knosys.2021.107275.
[13] D. Waterman, A Guide to Expert Systems, Addison-Wesley Pub. Co., Reading, MA, 1986.
[14] R. Rojas, The backpropagation algorithm, in: Neural Networks, Springer Berlin, Heidelberg, Berlin, 1996, pp. 149–182, http://dx.doi.org/10.1007/978-3-642-61068-4_7.
[15] L. Brown, R. Pezewski, J. Straub, Determining sentencing recommendations and patentability using a machine learning trained expert system, 2021, https://arxiv.org/abs/2108.04088v1 (accessed August 9, 2021).
[16] U. Sentencing Commission, United states sentencing commission variable codebook for individual offenders standardized research data documentation for, 1999.
[17] Commission Datafiles, US Sentencing Comm, 2021.
[18] Examiner handbook to the U.S. patent classification system, 2016 US Patent and Trademark Office. https://www.uspto.gov/patents/laws/examiner-handbook-us-patent-classification-system.
[19] A.V. Giczy, D. Scientist, A.A. Toole, C. Economist, N.A. Pairolero, Identifying artificial intelligence ( AI ) invention : A novel AI patent dataset, 2021.
[20] Artificial intelligence patent dataset, 2021 https://www.uspto.gov/ip-policy/economic-research/research-datasets/artificial-intelligence-patent-dataset.
[21] B. Fitzpatrick, X. Sherwin Liang, J. Straub, Fake news and phishing detection using a machine learning trained expert system, 2021, https://arxiv.org/abs/2108.08264v1 (accessed August 18, 2021).