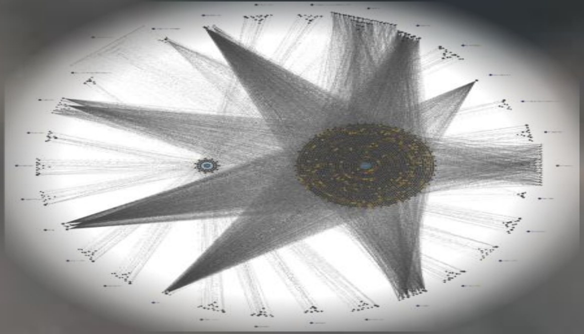
ICONIP 2023

November 20-23, 2023, Changsha, China

PAPER NO. 1220



Explainable Sparse Associative Self-Optimizing Neural Networks for Classification

Adrian Horzyk

AGH University of Krakow Poland

Jakub Kosno

Independent Researcher Poland

Daniel Bulanda

AGH University of Krakow Poland

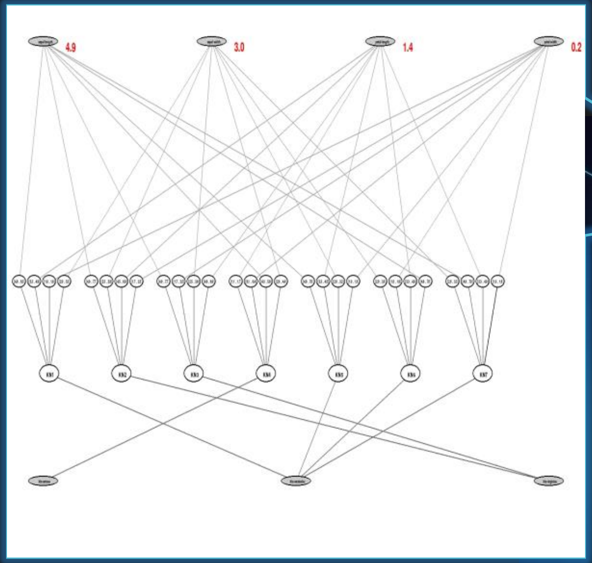
Janusz A. Starzyk

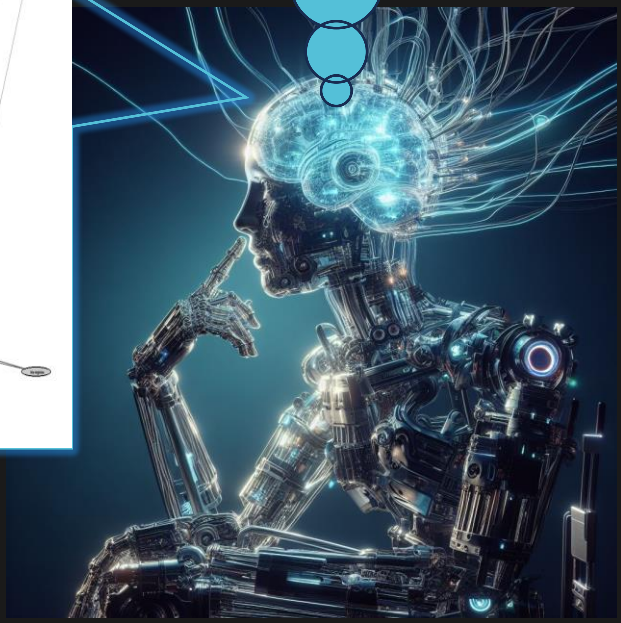
University of Information Technology and Management in Rzeszow, Poland and Ohio University, Athens, USA

Introduction and Contribution

- ➢ Artificial neural networks usually only adjust weights (and other parameters), but their structure of connections is rigid, regular, and not data-driven.
- ➢ Sparse, attention-like connections are more effective than rigid ones.
- ➢ The process of developing a neural network structure should be done concurrently with the process of adapting its weights to process data better.
- ✓ Associative Self-Optimizing Neural Networks (ASONN) are a neural network type that quickly develops sparse, well-generalizing and cost-effective structures that reproduces essential data relationships.
- ✓ ASONNs are fully explainable (models & results).
- ✓ ASONN's development process is based on association and aggregation of similar features and objects yielding 100% class discrimination for training data.
- ✓ ASONNs use only the most discriminative and cheapest input features to discriminate classes using different sub-hyperspaces called hypercuboids.

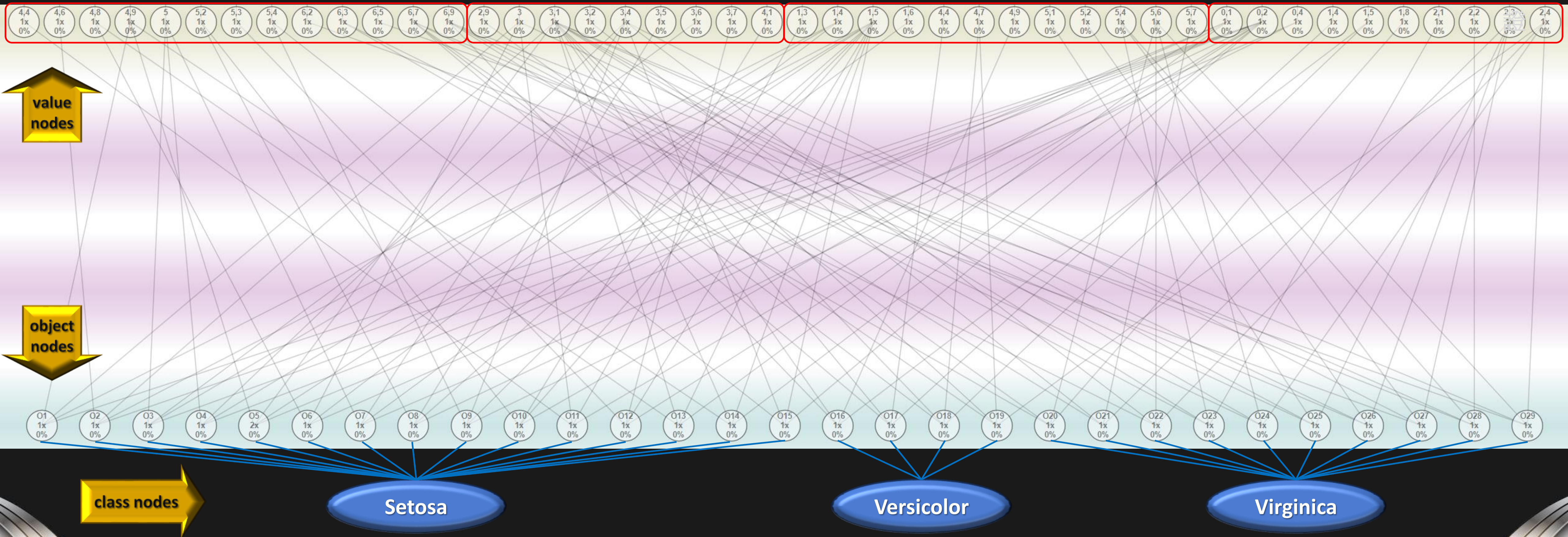
Brains have sparse irregular and dynamically developed neural network connections, while artificial networks are rigid and non-adaptable?





Backbone Associative Structure

ASONN classifiers are built on a sparse Associative Graph Data Structure (AGDS) that counts and aggregates all duplicate data points and orders all feature nodes for each data attribute separately.
Next, AGDS contextually connects the feature nodes (values) to the object nodes (training examples).

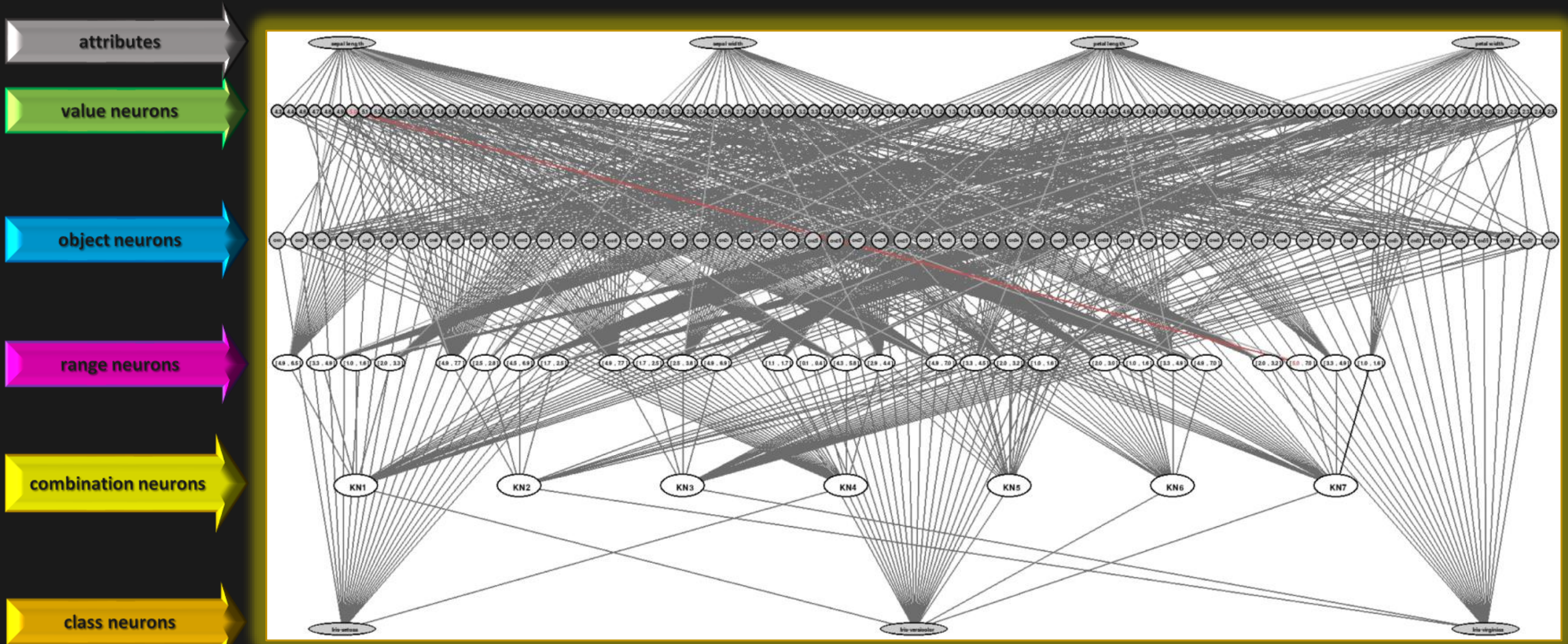


Class labels can be associated with object nodes (representing training examples) in the same way as the other features, so AGDS does not require to predefine a class label attribute, which can be chosen later.

ASONN Structure

ASONN structure consists of sparsely connected layers during construction:

1. Layer of value neurons (representing features)
2. Layer of object neurons (representing training examples)
3. Layer of range neurons (representing ranges of feature values)
4. Layer of combination neurons (representing combinations of ranges)
5. Layer of class neurons (representing class labels)



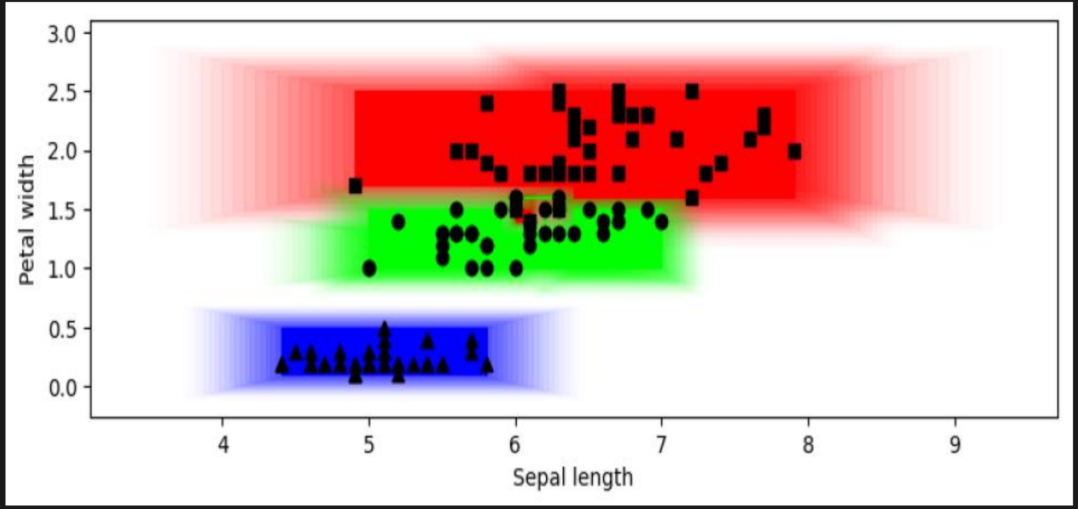
ASONN connections reflect the defined relationships between feature values, training examples, and classes in the training dataset and the definition of ranges and combinations determined by the construction algorithm for each discriminated class.

Sparse ASONN connections make natural attention to the most essential data-defining neurons!

Theoretical Background of ASONNs

Main Ideas and Objectives of ASONN Structures and Networks are:

- ASONN structures use aggregated representation of the same features to associate objects with similar characteristics (i.e., sharing similar features).
- ASONNs create sparse connections to the most relevant features of objects (strong attention), avoiding misleading connections that can disrupt calculations in subsequent layers, unlike dense and deep networks.
- Weights reproduce natural similarities, frequencies, and rareness of object features allowing for prioritizing informative features (soft attention) and eliminating the need for extensive training.
- The ASONN construction algorithm identifies fuzzy sub-hyperspace cuboids that effectively define and differentiate training examples of different classes.
- ASONNs achieve 100% training accuracy for unambiguous training data and demonstrate their generalization capabilities for test data.



Construction of fuzzy sub-hyperspace cuboids

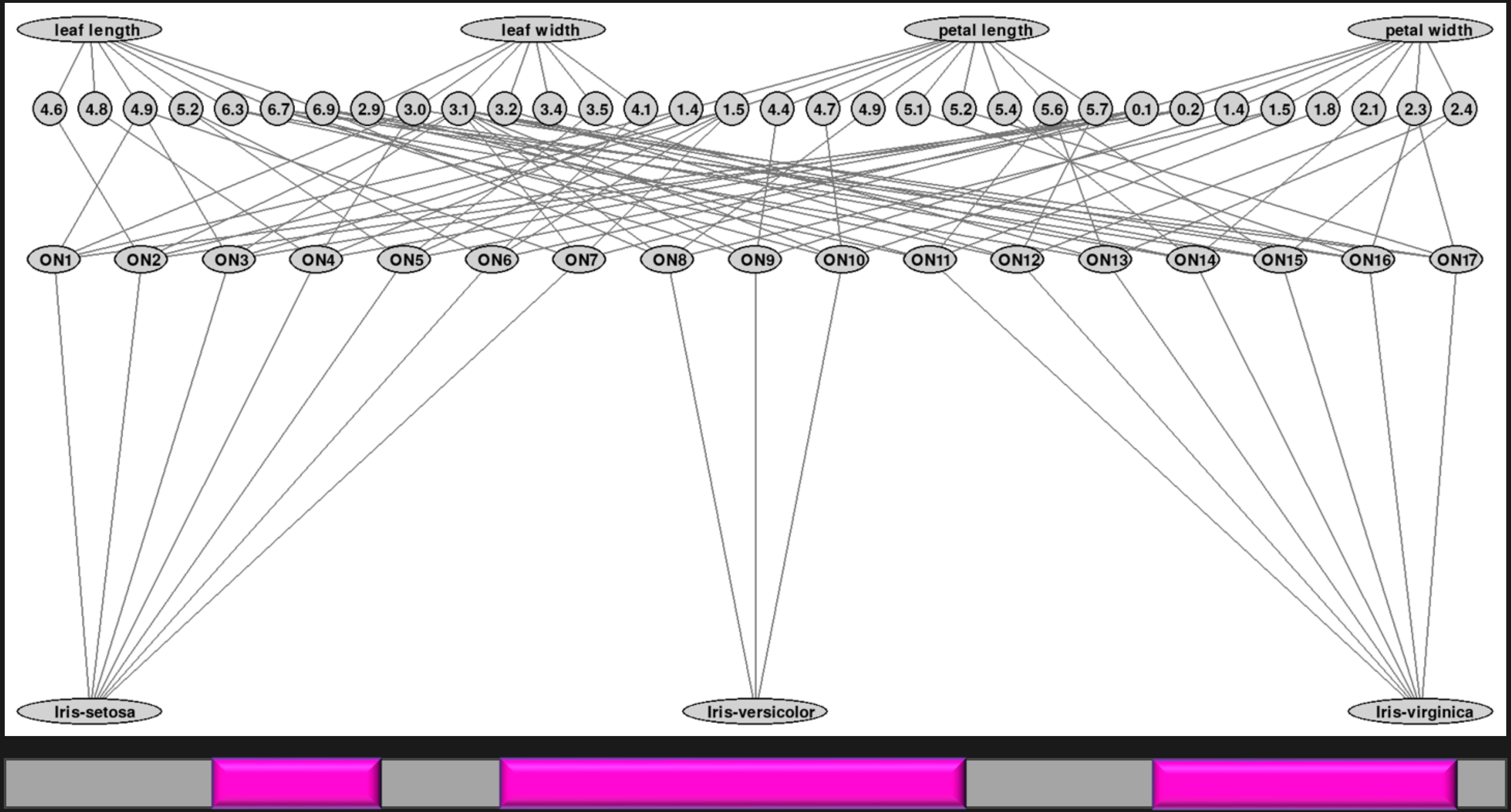
The ASONN construction identifies fuzzy sub-hyperspace cuboids that represent groups of similar training examples from the same classes.

Definition 1. Hypercuboid is an area of the input data subspace defined by the combination of the value subranges of all or selected attributes defining training examples.

Each cuboid grows and encompasses as many features of training examples of one class (seeds) as possible, while minimizing the inclusion of features also defining training examples of other classes (weeds).

ASONNs ensure that every training example belong to at least one hypercuboid!

The search process always starts from the most correlated training examples because they are the most difficult to discriminate and encompass by hypercuboids.



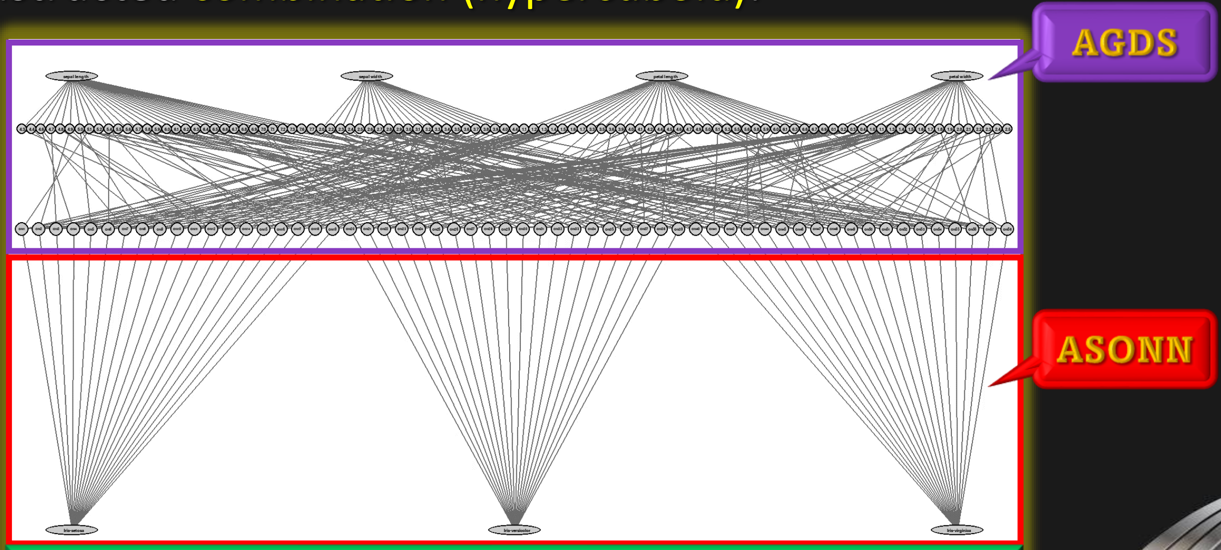
Definition 2. Seeds are all features of training examples from the same class as the class associated with the constructed hypercuboid, which are included in the expanded ranges defining this hypercuboid.

Definition 3. Weeds are all features of training examples from different classesthan the class associated with the constructed hypercuboid, which are included in the expanded ranges defining this hypercuboid.

ASONN Construction Algorithm

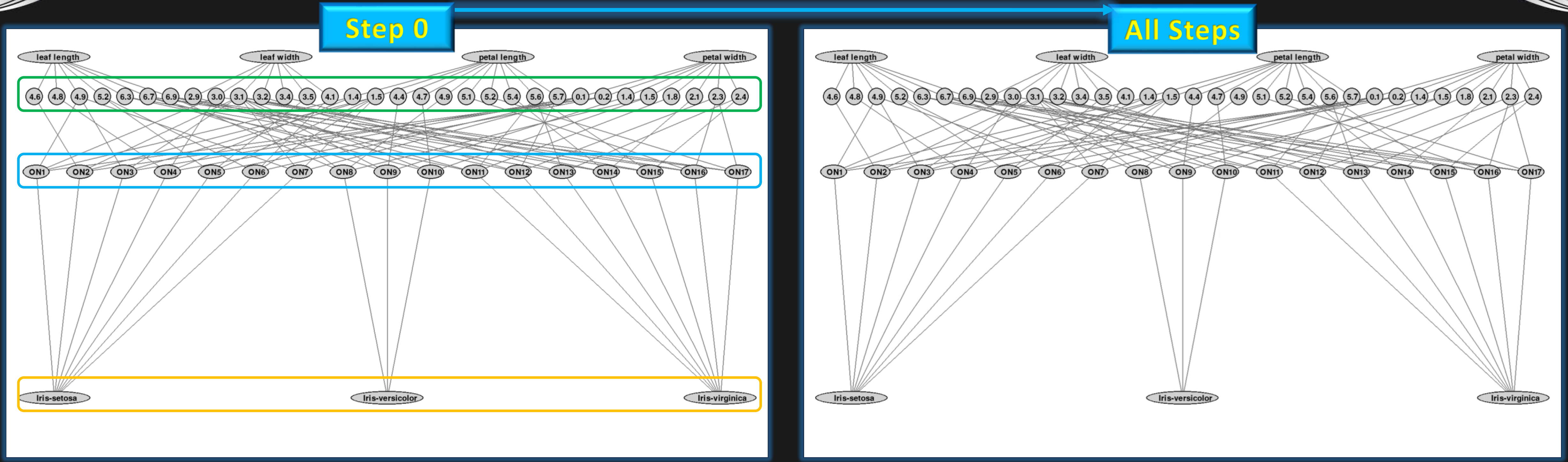
ASONN Construction Algorithm for any given training dataset follows the presented steps:

1. Transform training data into an Associative Graph Data Structure (AGDS), creating value, object, and class nodes.
2. Repeat until all objects are represented by at least one combination (representing a hypercuboid):
    1. Choose an object (training example) to initiate the construction of a new combination (hypercuboid) based on its correlation to other objects, creating also a few range neurons constituting this new combination.
    2. Repeat until adding the next value to the range-defining combination would result in representing an object of another class than the one represented by the constructed combination (hypercuboid):
        1. Expand ranges with all values that represent only features of objects of a class represented by the constructed combination (hypercuboid).
        2. Expand the range with the biggest expansion coefficient value that is calculated as a sum of seeds and weeds.
3. Remove all value and object nodes from the structure because they do not compose the ASONN model, but only range, combination, and class neurons.



Fast structure construction and weights calculation!

Detail Description of ASONN Construction

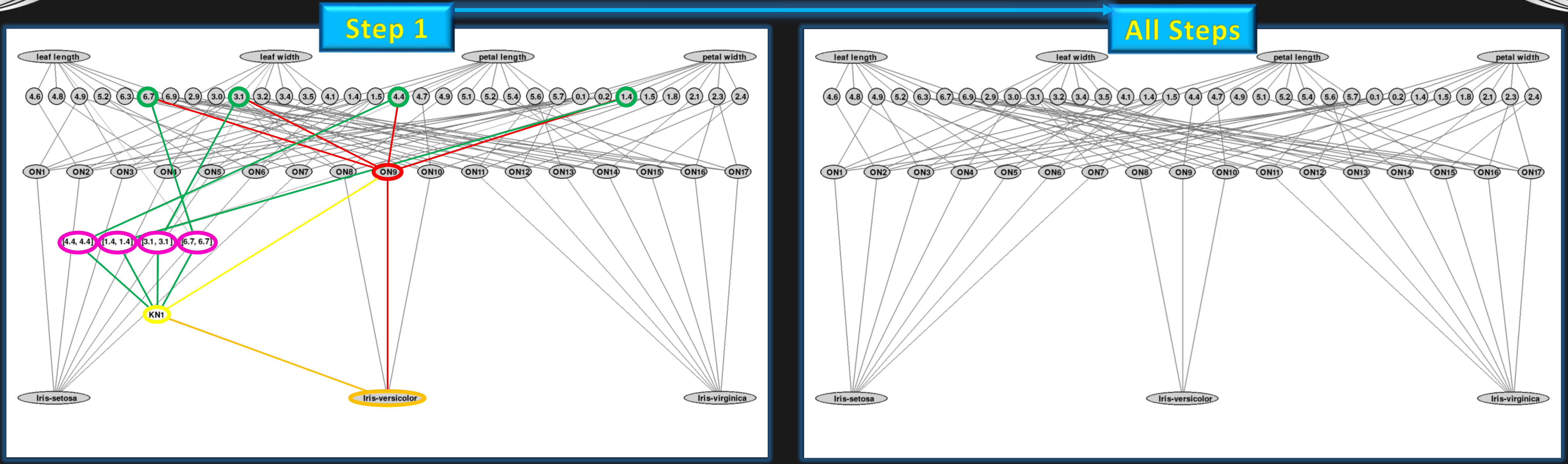


The ASONN construction process starts from the created AGDS structure for a given dataset, in which all nodes are transformed into neurons: Value Neurons (VNs), Object Neurons (ONs), and Class Neurons (CNs).

Next, the construction of hypercuboids is started, and every hypercuboid is constructed until at least one of the combination ranges can be expanded and the minimum number of discriminating features between these combinations is maintained (the detailed description available in the paper):

$$\left(dir^{-}_{RN^a_r} > 0 \lor dir^{+}_{RN^a} > 0\right) \land \left(discrim^{KN_k}_{min} \geq discrim^{user}_{min} \geq 1\right) = true. \quad (21)$$

Detail Description of ASONN Construction

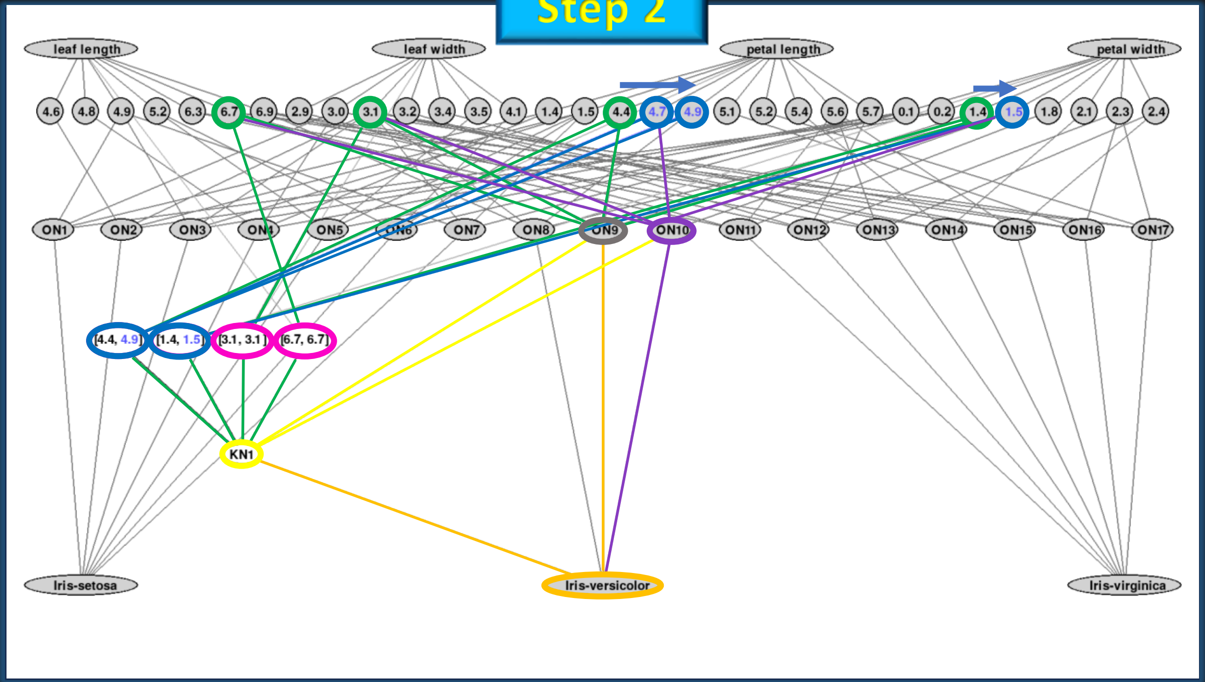


In the first step, we choose Object Neuron ON9 with the biggest correlation to all object neurons connected to the other classes than the class (CN – Iris-versicolor) to which the ON9 is connected.

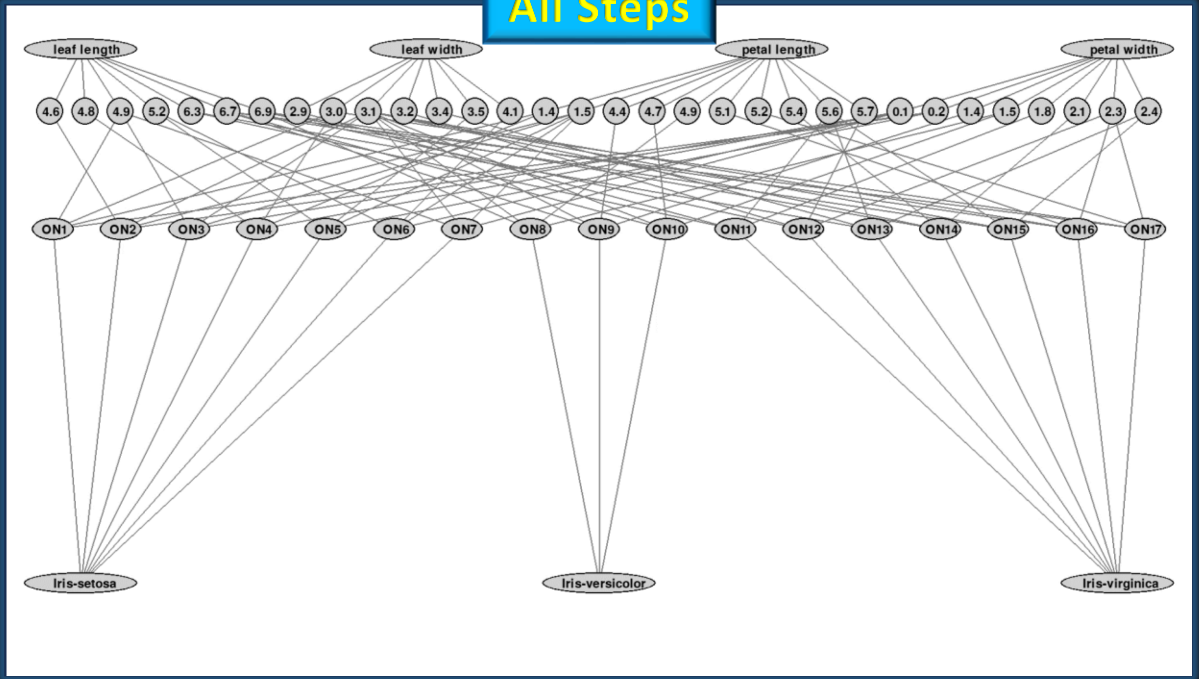
ON9 initializes a new combination neuron (KN1) and connects it to ON9' class (CN – Iris-versicolor), where KN1 is initially defined by the single-value ranges (represented by Range Neurons - RNs) taken from the Value Neurons (VNs) defining ON9.

Detail Description of ASONN Construction

Step 2



All Steps



Ranges can be extended by the values that exclusively belong to the same class as the class (Iris-versicolor) represented by the associated combination (KN1).

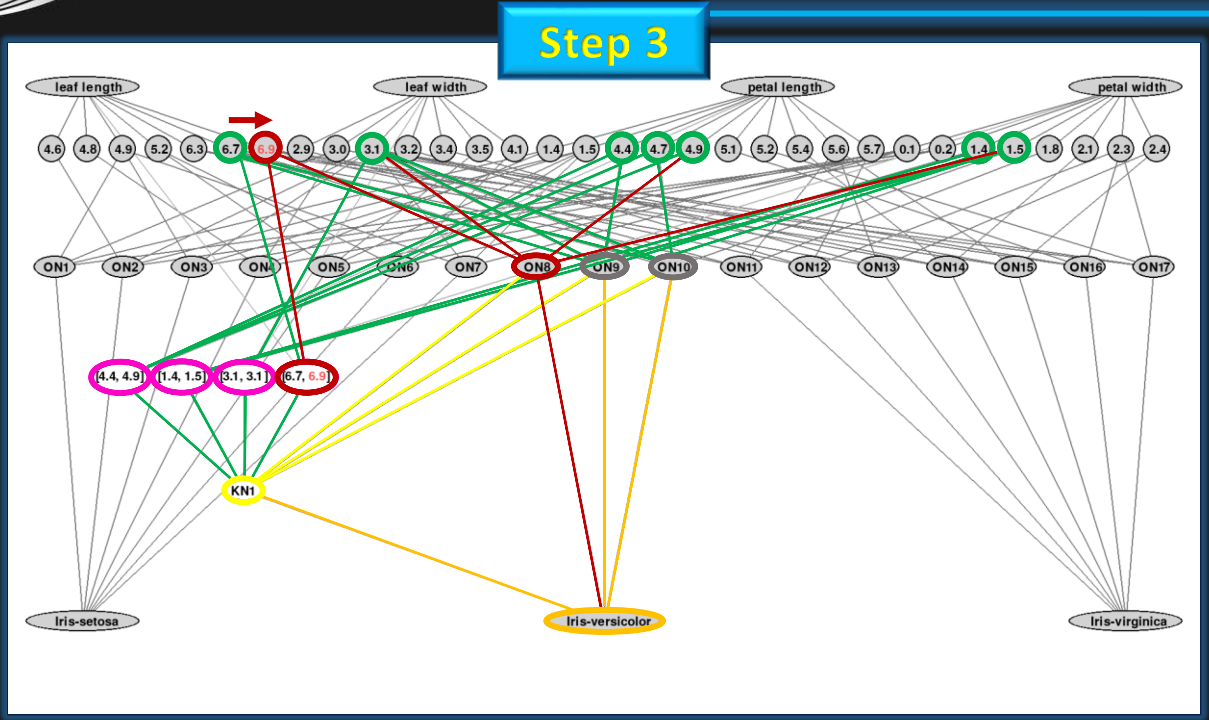
Range Neurons (RNs) are subsequently connected to neighbor Value Neurons that are exclusively connected only to Object Neurons that represent the same class (CN – Iris-versicolor) as the constructed combination (KN1) representing the growing hypercuboid. In such steps, we add only Seeds (no Weeds) to the combination.

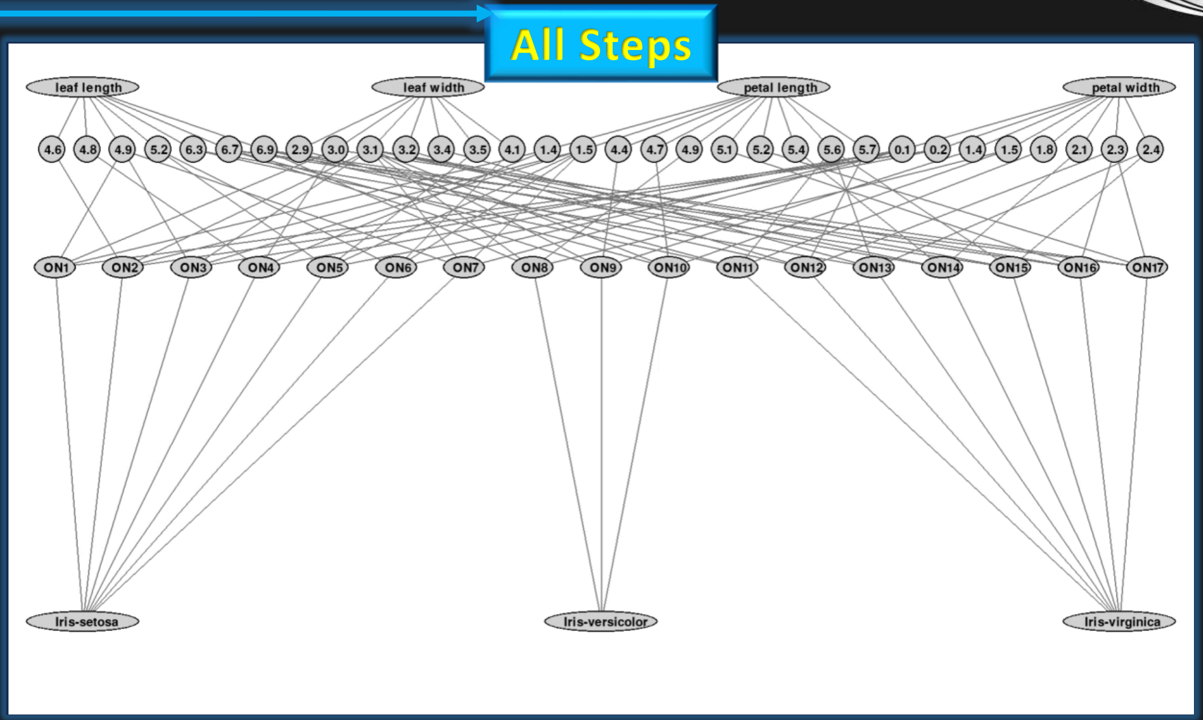
In a result, the next same-class Object Neurons (here ON10) can be added and connected to KN1.

Detail Description of ASONN Construction

Step 3

All Steps





Ranges can be also extended by the values that belong to the same class as the class (Iris-versicolor) represented by the associated combination (KN1), but also are associated with objects of other classes.

Range Neurons (RNs) are subsequently connected to such neighbor Value Neurons expanding represented ranges of the constructed combination (KN1). In such steps, we try to add as many Seeds and as few Weeds as possible to the combination. Expanded ranges and grown hypercuboid support generalization!

In a result, the next same-class Object Neurons (here ON8) can be added and connected to KN1.

Mathematical Background of ASONN Construction

$$dir^{-}_{VN^a_n} = \delta^{RN^a_{min}}_{VN^a_{val}} * \left(\sum_{ON^{same}} \delta^{repr}_{ON^{same}} * \delta^{cont}_{ON^{same}} - \sum_{ON^{diff}} \delta^{cont}_{ON^{diff}} \right)$$

$$dir^{+}_{VN^a_n} = \delta^{RN^a_{max}}_{VN^a_{val}} * \left(\sum_{ON^{same}} \delta^{repr}_{ON^{same}} * \delta^{cont}_{ON^{same}} - \sum_{ON^{diff}} \delta^{cont}_{ON^{diff}} \right)$$

where

$$\delta^{RN^a_{min}}_{VN^a_{val}} = \left(1 - \frac{VN^a_{val} - RN^a_{min}}{RN^a_{range}} \right)^2$$

$$\delta^{RN^a_{max}}_{VN^a_{val}} = \left(1 - \frac{RN^a_{max} - VN^a_{val}}{RN^a_{range}} \right)^2$$

$$RN^a_{range} = RN^a_{max} - RN^a_{min}$$

$$RN^a_{max} = \max_{VN^a_{val}} \{VN^a_{val} : \exists_{ON_m \leftrightarrow CN_c \leftrightarrow KN_k \leftrightarrow RN^a_r} ON_m \leftrightarrow VN^a_{val}\}$$

$$RN^a_{min} = \min_{VN^a_{val}} \{VN^a_{val} : \exists_{ON_m \leftrightarrow CN_c \leftrightarrow KN_k \leftrightarrow RN^a_r} ON_m \leftrightarrow VN^a_{val}\}$$

$$\delta^{repr}_{ON^{same}} = \left(\frac{1}{1 + q^{repr}_{ON^{same}}} \right)^2$$

$$q^{repr}_{ON^{same}} = \sum_{KN_k} \|\{ON_m : ON_m \leftrightarrow KN_k\}\|$$

Details available in the paper.

$$\delta^{cont}_{ON^{same}} = \left(\frac{1 + q^{VN^a}_{ON^{same}}}{\|RN^a_r : RN^a_r \leftrightarrow KN_k\|} \right)^2$$

$$\delta^{cont}_{ON^{diff}} = \left(\frac{1 + q^{VN^a}_{ON^{diff}}}{\|RN^a_r : RN^a_r \leftrightarrow KN_k\|} \right)^2$$

$$q^{VN^a}_{ON^{same}} = \sum_{RN^a_r \leftrightarrow KN_k} \|\{VN^a_n : RN^a_r \leftrightarrow VN^a_n \leftrightarrow ON_m \leftrightarrow KN_k\}\|$$

$$q^{VN^a}_{ON^{diff}} = \sum_{RN^a_r \leftrightarrow KN_k} \|\{VN^a_n : RN^a_r \leftrightarrow VN^a_n \leftrightarrow ON_m \not\leftrightarrow KN_k\}\|$$

In simple words, during the ASONN construction, we expand ranges that define combinations that represent hypercuboids encompassing and discriminating training examples of the same classes.

During the ASONN construction, we try to establish the most beneficial expansions that add the most Seeds and the fewest Weeds to the constructed combination.

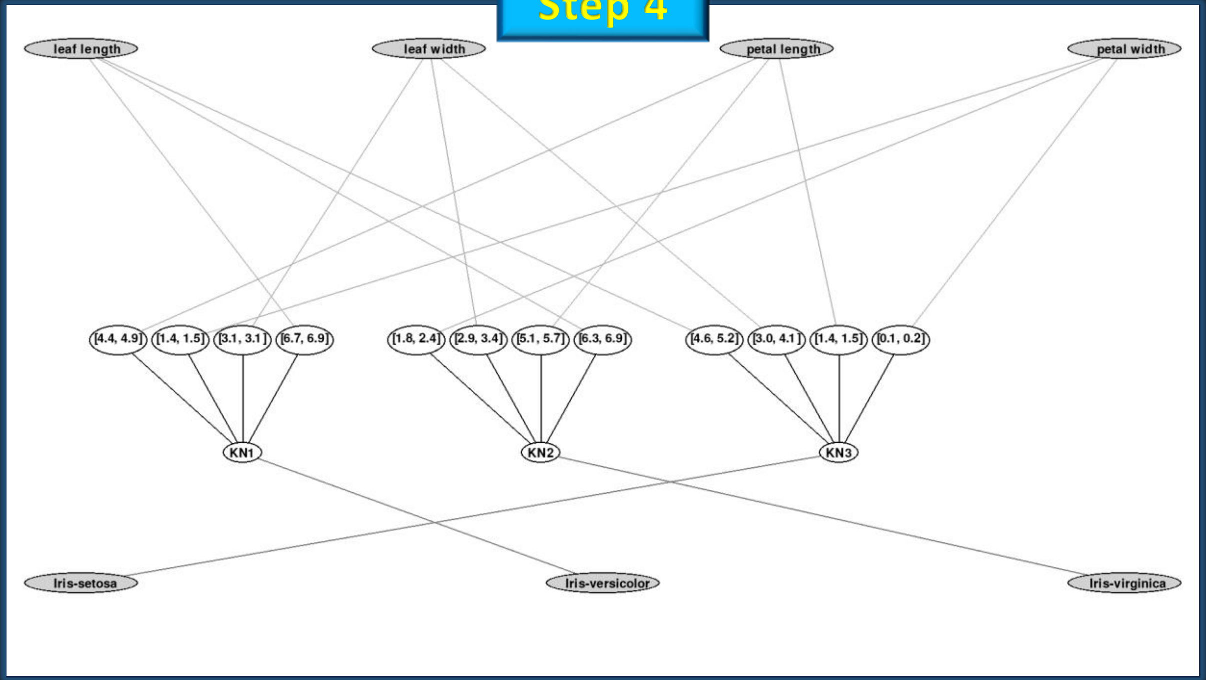
During the ASONN structure optimization process, we calculate coefficients (dir$^-$ and dir$^+$) for each potentially expanded range and choose the one (for all ranges) that is the most beneficial.

$$dir^{-}_{RN^a_r} = \frac{1}{\gamma_a} \sum_{VN^a_n < RN^a_{min}} dir^{-}_{VN^a_n}$$

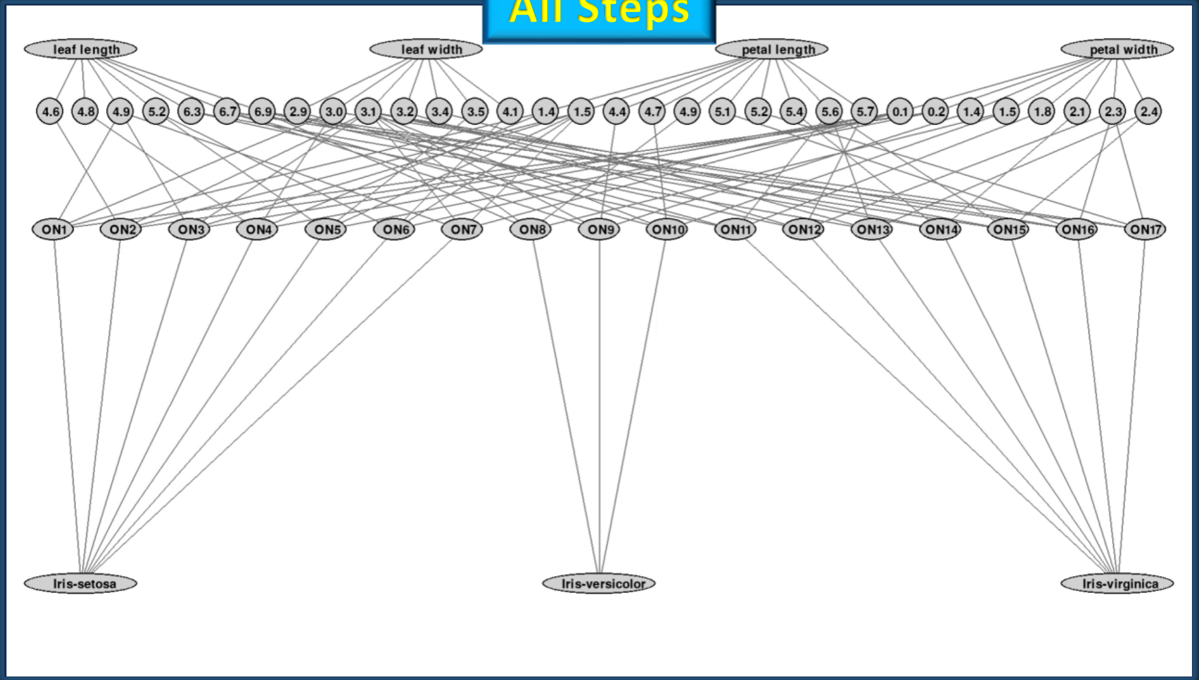
$$dir^{+}_{RN^a} = \frac{1}{\gamma_a} \sum_{VN^a_n > RN^a_{max}} dir^{+}_{VN^a_n}$$

Detail Description of ASONN Construction

Step 4



All Steps



When the training dataset is not updated over time (i.e., is static) and all objects (training examples) are represented by at least one combination neuron of ASONN, all value and object neurons together with their connections are removed, leaving the pure ASONN classifier structure consisting of range, combination, and class neurons.

Range neurons are additionally equipped with the Gaussian-cut-hat fuzzifying function to generalize outside the developed hypercuboids.

$$GCH_{RN^a}(x) = \begin{cases} 1 & if \quad RN^a_{Min} \le x \le RN^a_{Max} \\ e^{\alpha_{GCH}} & otherwise \end{cases} \qquad (22)$$

$$\alpha_{GCH} = \frac{1 - \left(\frac{2x - RN^a_{Max} - RN^a_{Min}}{RN^a_{Max} - RN^a_{Min}} \right)^2}{2} \qquad (23)$$

More details available in the associated paper.

Calculation of Connection Weights

✓ For the calculation of weights, we use different ratios calculated on the basis of the number of Seeds and Weeds established for the expanded ranges, as well as the number of connected objects and features:

Weights between Value Neurons

Weights between Value Neurons and Object Neurons

Weights between Range Neurons and Combination Neurons

Seeds inside the Range

Weeds inside the Range

Details available in the paper.

$$w_{VN_n^a,VN_{n+1}^a} = \left(1 - \frac{|v_n^a - v_{n+1}^a|}{R^a}\right)^s \qquad (1)$$

$$w_{VN_n^a,ON_i} = \frac{\frac{\|\{ON_j : VN_n^a \leftrightarrow ON_j \leftrightarrow CN_c \leftrightarrow ON_i\}\|}{\|\{ON_l : VN_n^a \leftrightarrow ON_l\}\|}}{\sum_{VN_m^a : VN_m^a \leftrightarrow ON_i} \frac{\|\{ON_j : VN_m^a \leftrightarrow ON_j \leftrightarrow CN_c \leftrightarrow ON_i\}\|}{\|\{ON_l : VN_m^a \leftrightarrow ON_l\}\|}} \qquad (2)$$

$$w_{RN^a \leftrightarrow KN_k} = \frac{\varphi_{RN_r^a}}{\sum_{\{RN_r^b : RN_r^b \leftrightarrow KN_k\}} \varphi_{RN_r^b}} \qquad (24)$$

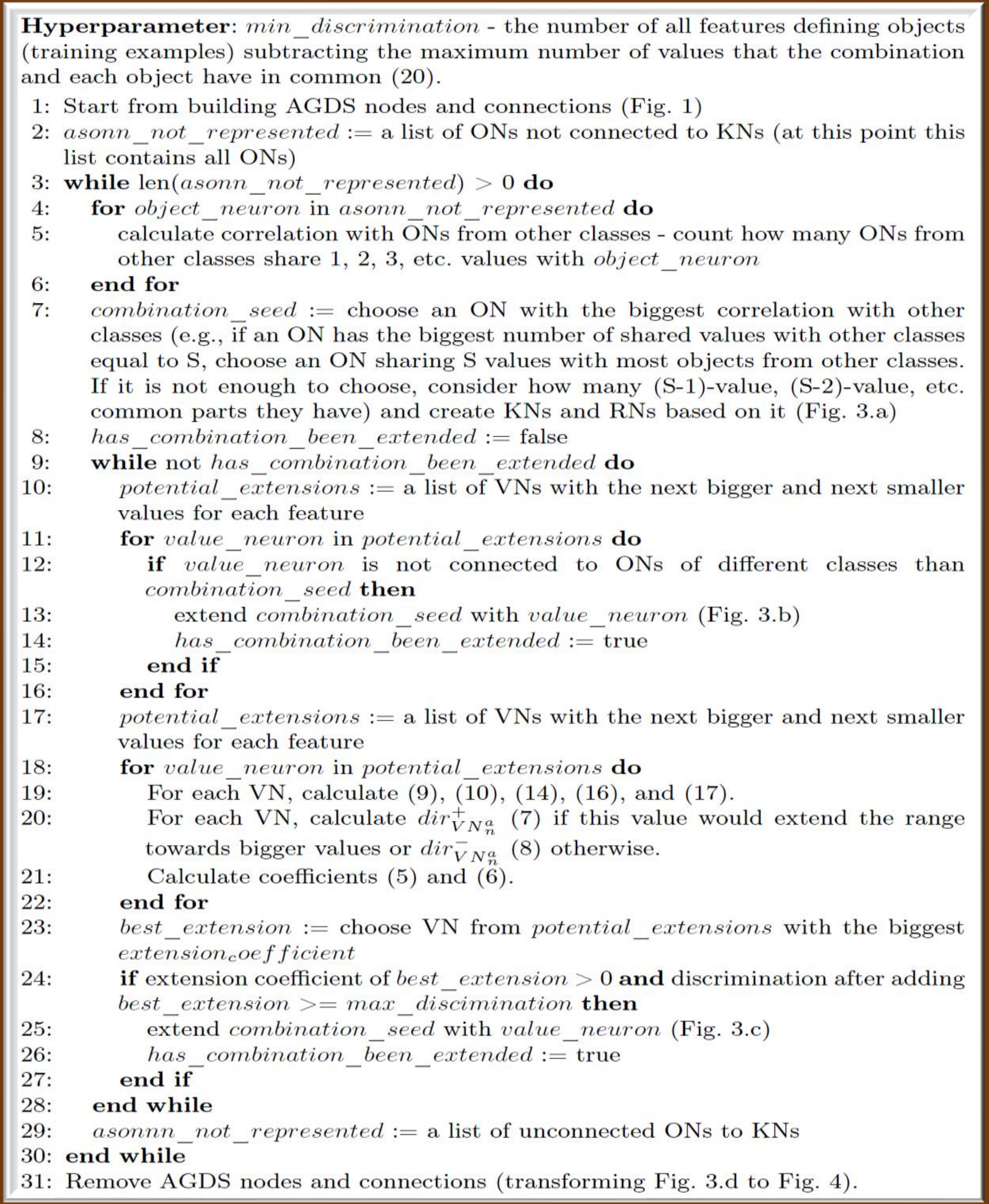
$$\varphi_{RN_k} = \left(1 - \frac{Weeds_{RN_k}}{\varphi_{ON^{diff}}^{discr}}\right) \frac{\varphi_{ON^{same}}^{repr} + Seeds_{RN_k}}{\varphi_{ON^{same}}^{repr} + AllSeeds_{KN}} \qquad (25)$$

$$Seeds_{RN_r^a} = \sum_{\{ON_i : ON_i \leftrightarrow CN_c \leftrightarrow KN_k \leftrightarrow RN_r^a\}} \|\{VN_n^a : RN_r^a \leftrightarrow VN_n^a \leftrightarrow ON_i\}\|^2$$

$$Weeds_{RN_r^a} = \sum_{\{\widetilde{ON}_i : \widetilde{ON}_i \leftrightarrow \widetilde{CN}_e \not\leftrightarrow KN_k \leftrightarrow RN_r^a\}} \left\|\left\{VN_n^a : RN_r^a \leftrightarrow VN_n^a \leftrightarrow \widetilde{ON}_i\right\}\right\|^2$$

Range & Combination Expansion Algorithm

- ✓ In the associated paper, there is presented a pseudocode of the algorithm for expansion of ranges and combinations.
- ✓ The source code of the implemented algorithm is available on [14]:



Hyperparameter: $min\_discrimination$ - the number of all features defining objects (training examples) subtracting the maximum number of values that the combination and each object have in common (20).

```
1:  Start from building AGDS nodes and connections (Fig. 1)
2:  $asonn\_not\_represented$ := a list of ONs not connected to KNs (at this point this list contains all ONs)
3:  **while** len($asonn\_not\_represented$) $> 0$ **do**
4:    **for** $object\_neuron$ in $asonn\_not\_represented$ **do**
5:      calculate correlation with ONs from other classes - count how many ONs from other classes share 1, 2, 3, etc. values with $object\_neuron$
6:    **end for**
7:    $combination\_seed$ := choose an ON with the biggest correlation with other classes (e.g., if an ON has the biggest number of shared values with other classes equal to S, choose an ON sharing S values with most objects from other classes. If it is not enough to choose, consider how many (S-1)-value, (S-2)-value, etc. common parts they have) and create KNs and RNs based on it (Fig. 3.a)
8:    $has\_combination\_been\_extended$ := false
9:    **while** not $has\_combination\_been\_extended$ **do**
10:     $potential\_extensions$ := a list of VNs with the next bigger and next smaller values for each feature
11:     **for** $value\_neuron$ in $potential\_extensions$ **do**
12:       **if** $value\_neuron$ is not connected to ONs of different classes than $combination\_seed$ **then**
13:         extend $combination\_seed$ with $value\_neuron$ (Fig. 3.b)
14:         $has\_combination\_been\_extended$ := true
15:       **end if**
16:     **end for**
17:     $potential\_extensions$ := a list of VNs with the next bigger and next smaller values for each feature
18:     **for** $value\_neuron$ in $potential\_extensions$ **do**
19:       For each VN, calculate (9), (10), (14), (16), and (17).
20:       For each VN, calculate $dir^{+}_{VN^{a}_{n}}$ (7) if this value would extend the range towards bigger values or $dir^{-}_{VN^{a}_{n}}$ (8) otherwise.
21:       Calculate coefficients (5) and (6).
22:     **end for**
23:     $best\_extension$ := choose VN from $potential\_extensions$ with the biggest $extension_coefficient$
24:     **if** extension coefficient of $best\_extension > 0$ **and** discrimination after adding $best\_extension >= max\_discimination$ **then**
25:       extend $combination\_seed$ with $value\_neuron$ (Fig. 3.c)
26:       $has\_combination\_been\_extended$ := true
27:     **end if**
28:   **end while**
29:   $asonn\_not\_represented$ := a list of unconnected ONs to KNs
30: **end while**
31: Remove AGDS nodes and connections (transforming Fig. 3.d to Fig. 4).
```

ASONN Classifier Construction

attributes

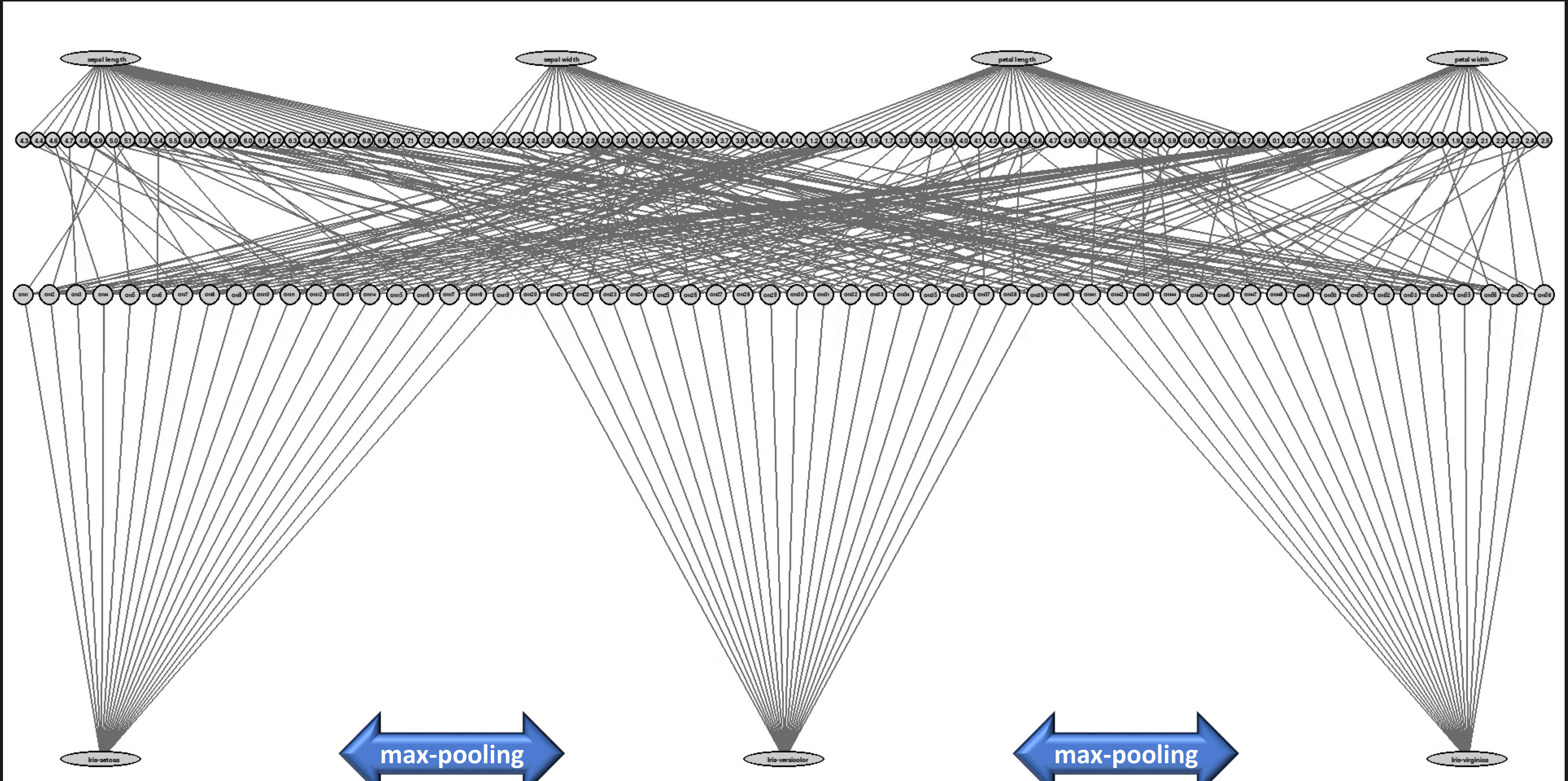
value neurons

object neurons

range neurons

combination neurons

class neurons



Starting from the Associative Graph Data Structure for a given dataset, the ASONN structure is developed in the fast construction process that creates range and combination neurons.

AGDS

The ASONN weights are quickly calculated not trained!

ASONN

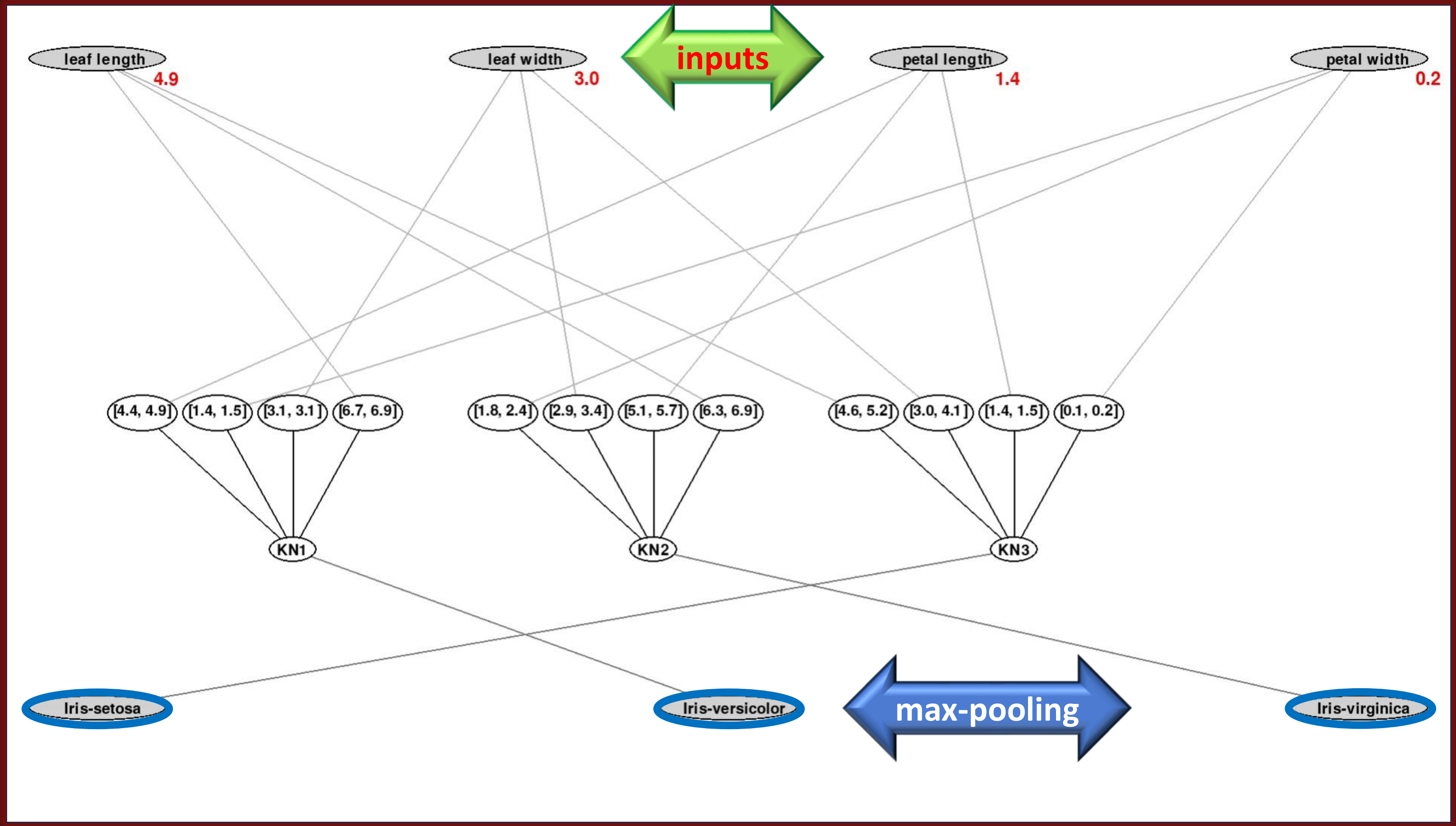
The final ASONN structure is small, sparse, consistent, optimized, and explainable!

The expansions of the ranges and combination neurons for the 60-example dataset is presented above.

ASONN Classification

The ASONN classification process is defined by the following steps:

1. Feed a test example (4.9, 3.0, 1.4, 0.2) to the ASONN input range neurons.
2. The range neurons representing the fed features produce outputs (in between 0 and 1) that stimulate the combination neurons.
3. The combination neurons calculate their excitement and compete with the other combination neurons (in max-pooling operation) for the strongest reaction that is measured by the class neurons.
4. Finally, a soft-max layer is implemented on the output to designate the winning class.

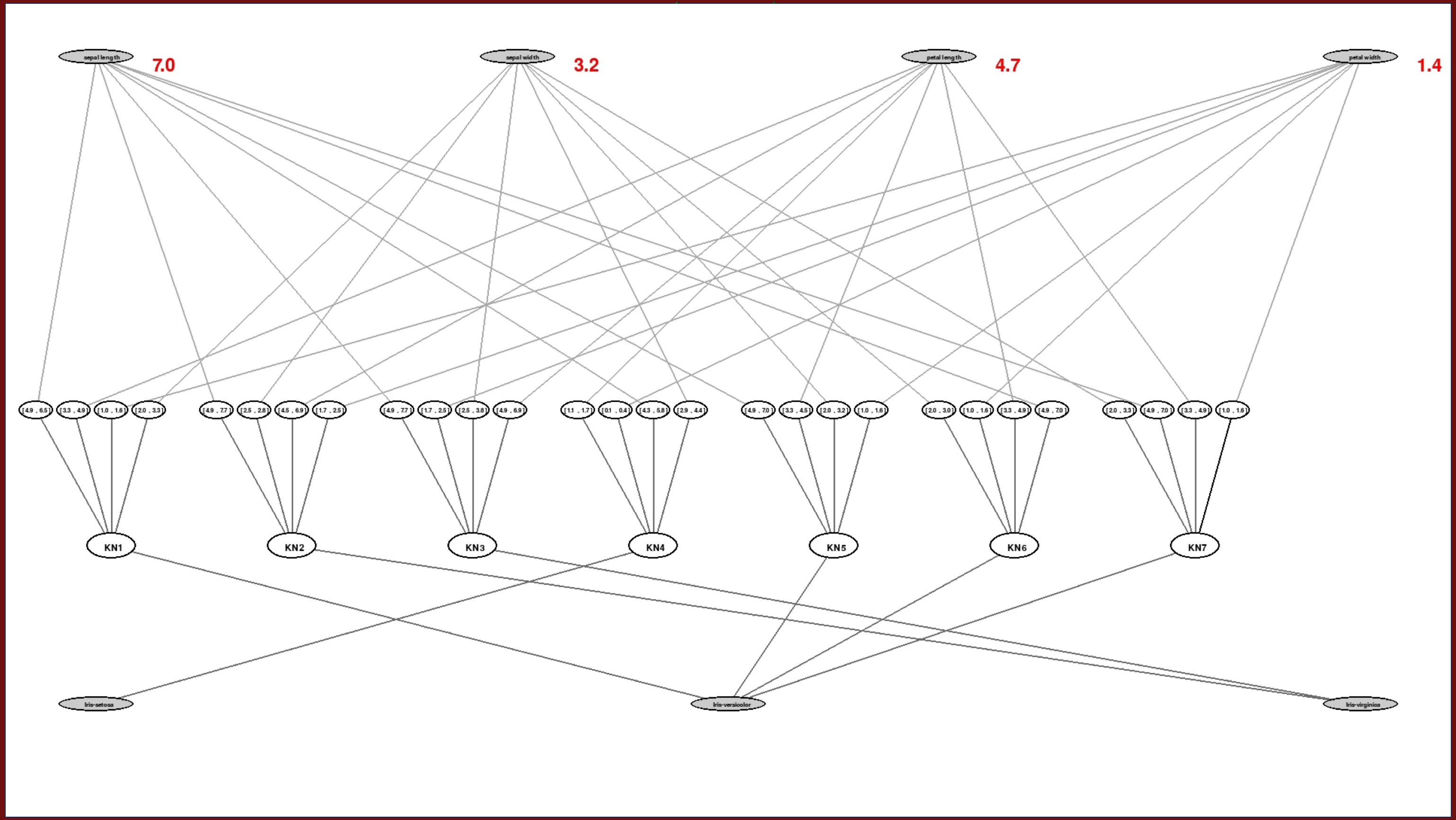


The ASONN construction and classification processes look unimaginably simple in comparison to many contemporary deep network models, but the sparse and relation-sensitive connections replace many deep layers that must be trained in a long-lasting optimization process.

ASONN Classification

The ASONN classification process is defined by the following steps:

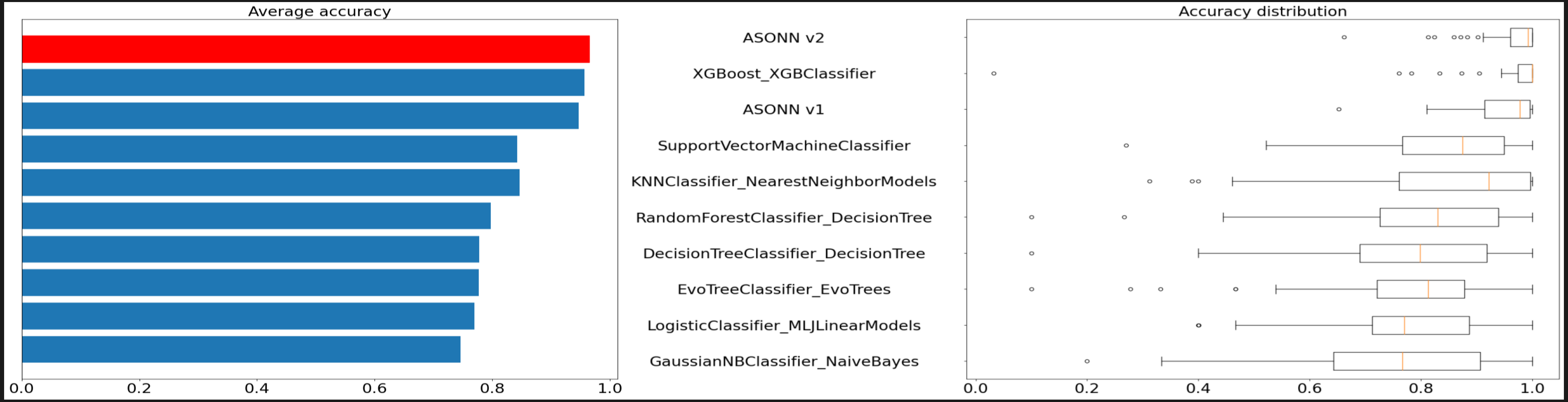
1. Feed a test example (7.0, 3.2, 4.7, 1.4) to the ASONN input range neurons.
2. Range neurons representing the fed features produce outputs (in between 0 and 1) that stimulate the combination neurons.
3. Combination neurons calculate their excitement and compete with the other combination neurons (in max-pooling operation) for the strongest reaction that is measured by the class neurons.
4. Finally, a soft-max layer is implemented on the output to designate the winning class.



The ASONN network and the classification process look unimaginably simple in comparison to many contemporary deep network models, but the sparse and relation-sensitive connections replace many deep layers that must be trained in a long-lasting optimization process.

Experimental Results

- ✓ The performance of the proposed ASONN network, constructed and adapted by the described algorithm, was tested on typical machine learning classification tasks and compared to other popular models.
- ✓ We used 52 representative datasets from the PMLB benchmark dataset [13], which have between 32 and 3200 instances and contain between 3 and 34 binary, categorical, and numerical features and define between 2 and 10 classes.
- ✓ We have removed datasets from comparisons for which some of the compared models failed to produce results.
- ✓ The train and test sets were chosen in a ratio of 70:30.



Thanks to the automatically adaptable sparse structure and attention to the most essential features, ASONN v2 classification usually delivered the best results.

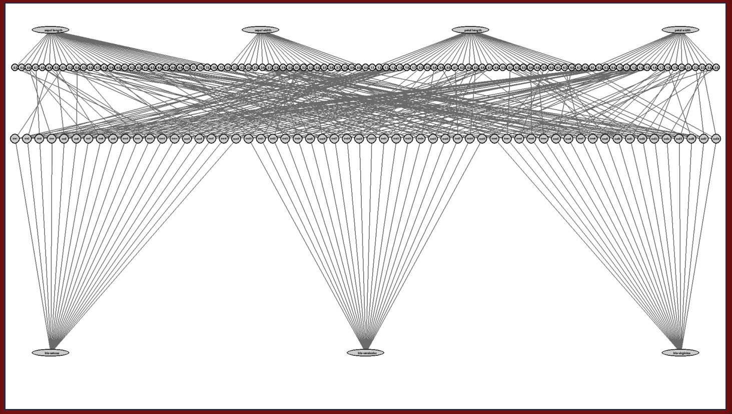
ASONN v1 and ASONN v2 differ mainly in the stop condition of the range extension algorithm.

Conclusions & Remarks

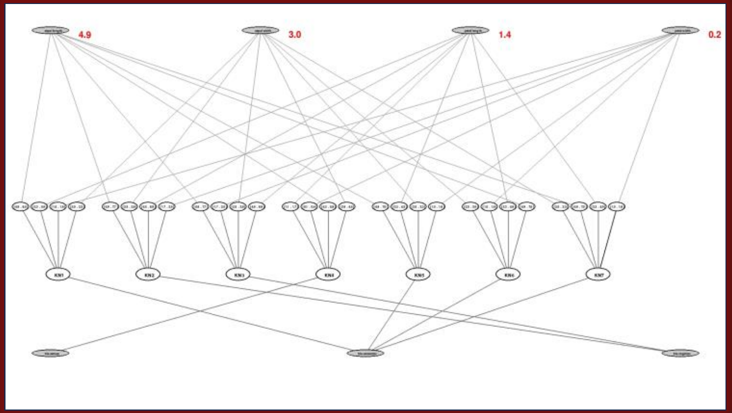
Associative Self-Optimizing Neural Networks (ASONN)
allow for the very fast development of classifiers that

- ✓ work based on hypercuboids created in different sub-hyperspaces, optimizing their performance;
- ✓ have very sparse structure of connections which work like hard-attention reproducing key data relationships;
- ✓ do not require optimizing of hyperparameters unlike many other models;
- ✓ can adapt their structures and parameters simultaneously and automatically;
- ✓ can automatically take into account different costs of features and prefer these features which are the cheapest (producing cost-effective solutions);
- ✓ are robust, predictable, trustful, explainable, and well-generalizing;
- ✓ achieve very high performance that is similar to the best-performing classifiers (e.g., XGBoost) on vectorized data which objects are defined by the subgroups of similar features that can be automatically extracted and used.

CONSTRUCTION



EVALUATION



Thank you!

Adrian Horzyk

AGH University of Krakow
Poland
horzyk@agh.edu.pl

Jakub Kosno

Independent Researcher
Poland

Daniel Bulanda

AGH University of Krakow
Poland

Janusz A. Starzyk

University of Information Technology and Management in Rzeszow, Poland and
Ohio University, Athens, USA

1. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning, MIT Press (2016).
2. Ali, S., Abuhmed, T., El-Sappagh, S., Muhammad, K., Alonso-Moral, J.M., Confalonieri, R., Guidotti, R., Del Ser, J., Díaz-Rodríguez, N., Herrera, F.: Explainable Artificial Intelligence (XAI): What we know and what is left to attain Trustworthy Artificial Intelligence. Information Fusion, 101805, Elsevier (2023).
3. Dwivedi, R., Dave, D., Naik, H., Singhal, S., Omer, R., Patel, P., Qian, B., Wen, Z., Shah, T., Morgan, G., et. al: Explainable AI (XAI): Core ideas, techniques, and solutions. ACM Computing Surveys, 55(9):1–33, ACM New York, NY (2023).
4. Hedström, A., Weber, L., Krakowczyk, D., Bareeva, D., Motzkus, F., Samek, W., Lapuschkin, W., and M-C Höhne, M.: Quantus: An Explainable AI Toolkit for Responsible Evaluation of Neural Network Explanations and Beyond. Journal of Machine Learning Research, 24(34):1–11 (2023).
5. Vaswani, Ashish, et al.: Attention is all you need. Advances in neural information processing systems 30 (2017), https://arxiv.org/abs/1706.03762.
6. Subutai, A., Scheinkman, L.: How can we be so dense? The benefits of using highly sparse representations. arXiv preprint arXiv:1903.11257 (2019).
7. Runge, K., Cardoso, C., de Chevigny, A.: Dendritic Spine Plasticity: Function and Mechanisms. Frontiers Synaptic Neuroscience (2020) Aug 28;12:36. doi: 10.3389/fnsyn. 2020.00036. PMID: 32982715; PMCID: PMC7484486.
8. Pchitskaya, E., Bezprozvanny, I.: Dendritic Spines Shape Analysis-Classification or Clusterization? Perspective. Front Synaptic Neurosci (2020) Sep 30;12:31. doi:10.3389/fnsyn.2020.00031. PMID: 33117142; PMCID: PMC7561369.
9. Kasabov, N.K.: Time-Space, Spiking Neural Networks and Brain-Inspired Artificial Intelligence, In: Springer Series on Bio- and Neurosystems, Vol 7., Springer (2019).
10. Horzyk, A: Associative Graph Data Structures with an Efficient Access via AVB+trees, In: 11th International Conference on Human System Interaction (HSI), IEEE Xplore, pp. 169 - 175 (2018).
11. Horzyk, A.: Artificial Associative Systems and Associative Artificial Intelligence. Academic Publishing House EXIT, Warsaw (2013).
12. Linoff, G.S., Berry, M.A.: Data Mining Techniques: For Marketing, Sales, and Customer Relationship Management, 3rd Edition (2011).
13. Olson, R.S., La Cava, W., Orzechowski, P., Urbanowicz, R., Moore, J.H.: PMLB: A Large Benchmark Suite for Machine Learning Evaluation and Comparison. Bio-Data Mining 10(1) (2017).
14. 14. https://github.com/jakubkosno/ASONNv2
15. Gale, T., Elsen, E., Hooker, S.: The State of Sparsity in Deep Neural Networks (2019). arXiv:1902.09574
16. Liu, S., Chen, T., Zhang, Z., Chen, X., Huang, T., Jaiswal, A., and Wang, Z.: Sparsity May Cry: Let Us Fail (Current) Sparse Neural Networks Together! (2023). arXiv preprint arXiv:2303.02141.
17. Mocanu, D.C., Mocanu, E., Stone, P., et al.: Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. Nat Commun 9, 2383 (2018). https://doi.org/10.1038/s41467-018-04316-3
18. Atashgahi, Z., Pieterse, J., Liu, S., et al.: A brain-inspired algorithm for training highly sparse neural networks. Mach Learn 111, 4411–4452 (2022).