Short thesis for the degree of doctor of philosophy (PhD)

Increasing the efficiency of fuzzy logic and neural network algorithms

by Tibor Gábor Tajti

Supervisor: Dr. István Fazekas

UNIVERSITY OF DEBRECEN
Doctoral school of Informatics

Debrecen, 2020

Contents

1 Increasing the efficiency of fuzzy logic and neural network algorithms

1.1 Introduction and motivation

Fuzzy logic and artificial neural networks are two fields of the area of modern artificial intelligence, providing various algorithms with the ability of learning and providing decision support for us.

Fuzzy logic has extended the Boolean logic to deal with many truth values. Fuzzy logic has been used in many applications such as machine control, knowledge-based systems, optimization problems, weather forecasting, risk assessment, medical diagnosis, and treatment plans, etc. Some of its applications can need the processing of very large formulas, e.g. system controlling and machine learning, where there can be a large amount of data to run our algorithms on, so the efficiency of our algorithms is essential. The short-circuit evaluation technique, which is used for binary logic by the interpreters and compilers of programming languages, can also be applied to fuzzy logic expressions. Extending this short-circuit evaluation with additional pruning techniques for the evaluation of fuzzy logic formulas can further improve performance, this motivated me to do this research.

One of the most widely used machine learning algorithms is the Artificial Neural Network or its Deep and Convolutional variants. Neural network algorithms are supervised machine learning algorithms, widely used in machine learning. Its major applications include classification, regression, pattern recognition, function approximation, intelligent control, learning from data.

Neural network and Convolutional neural network algorithms are among the best performing machine learning algorithms. One known issue of neural networks used for classification is that training data usually have binary output values, even when the training samples may belong to more than one class at a certain fuzzy level. There can be cases where these crisp (binary) class membership values can be considered misleading, so the correction of these values can lead to reducing the confounding effect of them. I have developed a simple method to correct the crisp class membership values via fuzzification.

The performance of the neural network algorithms may vary between multiple runs because of the stochastic nature of these algorithms. This stochastic behavior can result in worse-than-average accuracy for a single run, and in many cases, it is difficult to decide, whether we should repeat the learning, giving a chance to have a better result. Among the useful techniques to solve this problem, we can use the committee machine and the ensemble methods which in many cases give better than average or even better than the best individual result. There are many well-known techniques to do that, I also had intuitions to propose new methods for ensemble learners.

Enhancements made in machine learning algorithms can expand the range of problems which we can run on the presently available computers. Better efficiency also can help to make them usable on lower performance hardware. This can make the application of machine learning more affordable.

1.2 Primary goals

Short circuit evaluation (or short-cut evaluation) is widely used in programming languages for efficient evaluation of logical expressions. Parts of the expressions can be cut from the evaluation if they do not have an effect on the final value of the expression. In the case of large expressions, this can be very effective. I performed research to develop fast evaluation algorithms for fuzzy logic expression trees. My goal was to find techniques to make short-cuts in the recursive evaluation algorithm in addition to the trivial short-cut possibilities, which are used by the interpreters or compilers of programming languages to evaluate boolean logic expressions. My research and development was to achieve fast evaluations for large formula trees of Gödel fuzzy logic [1], Product logic [2], and Łukasiewicz formulae [3, 4] by pruning from the evaluation nodes and their children, if the values of them do not affect the final result of the formula.

My goal was also to develop a flexible expression tree generator that can be used to generate formula trees for the chosen fuzzy logic type, allowing to set the shape of the generated trees and also providing a parameter to set the value distribution for the leaf nodes.

The planned performance evaluation also required a flexible framework from which the formula generator and the multiple variants of the fast evaluation algorithm can be executed to do the measurements.

Committee machines are very efficient in machine learning. I performed research on advanced techniques for Neural Network and Convolutional Neural Network algorithms using three different models for performance evaluation. I have examined the usability of fuzzification of the crisp (binary) class membership values during training to correct noisy or possibly misclassified training samples in order to achieve better accuracy. The main goal was the correction of the training data class membership values, where the given crisp class membership values can be misleading for some samples. I also conducted research on developing new committee machine (ensemble) algorithms and variations. I have researched the possible use of new variants of committee machine simple voting functions, and new meta voting functions which can use the single voting functions as their input.

1.3 Theses and results

1.3.1 Thesis I: Fast evaluation of large fuzzy logic formula trees

I developed new algorithms for the evaluation of fuzzy logic expression trees of Gödel type fuzzy logic [1], Product fuzzy logic [2] and Łukasiewicz fuzzy logic [3, 4] formulas (see subchapters 5.1, 5.2, 5.3 of the dissertation) [5, 6]. The algorithms work by limiting the evaluation of the actual node by lower and upper limits to set the interval in which we are interested in the exact value of the node. By this technique, a high ratio of the nodes in a large formula tree can be pruned from the evaluation, without affecting the final result of the evaluation. This will lead to an optimized algorithm needing less performance to evaluate the formula tree.

Here we only show one of the proposed fast evaluation algorithms, for the evaluation of Łukasiewicz fuzzy logic formula trees.

```
function PruneEval1(node, lower, upper):
  if lower >= upper:
    return v = lower # cut

  if nodetype == NODE_TYPE_NEG:
    v = 1 - PruneEval1(child, 1-upper, 1-lower)

  elif nodetype == NODE_TYPE_AND:
    v1 = PruneEval1(child1, lower, 1)
    v2 = PruneEval1(child2, min(1+lower-v1,1), min(1+upper-v1,1))
    v  = max(v1+v2-1, 0)

  elif nodetype == NODE_TYPE_OR:
    v1 = PruneEval1(child1, 0, upper)
    v2 = PruneEval1(child2, max(lower-v1,0), max(upper-v1,0))
    v  = min(v1+v2, 1)

  elif nodetype == NODE_TYPE_IMP:
    v1 = PruneEval1(child1, 1-upper, 1)
    v2 = PruneEval1(child2, max(lower-1+v1,0), max(upper-1+v1,0))
    v  = min(1-v1+v2, 1)

  else: # NODE_TYPE_LEAF
    v = node value

  return v
end function

value_of_the_expression = PruneEval1(root, 0, 1)
```

The algorithm starts by calling it with the input formula tree with its root and two parameters, *lower* and *upper*, then it computes the value y of the expression if $lower \leq y \leq upper$. The interesting interval is represented by the numbers *lower* and *upper*, as its lower and upper limits, for the value v of a node such that this value has an influence on the result of the main formula if v is in its interesting interval. This interval depends on the already analyzed part of the expression(tree). The cuts during the evaluation are performed by adjusting the parameters *lower* and *upper* dynamically during the run for each node of the formula tree. More precisely, the role of *lower* and *upper* is as follows.

While evaluating the children of the current node we must continue the evaluation only if their value can be between the actual limits *lower* and *upper* provided for their evaluation. Intuitively, when $lower >= upper$, there is no interval between them, thus the given node or sub-tree can be cut, the value of the main formula does not depend on the value(s) of this node or sub-tree. We do the recursive evaluation of the formula tree with simple recursive function calls. The main call can be seen in the last line of the algorithm.

Expression tree generation

For analysis purposes, I developed a framework to produce a large number of formula trees on which the evaluation algorithms can be executed. The framework has been created for general purposes allowing to set parameters for the distribution of values in the $[0, 1]$ interval,

the shape of the tree, the allowed maximal number of child nodes for the node types, and the number of nodes [5, 7], as described below.

Since the fuzzy logic types are three or more valued, many-valued, or have infinitely many-valued distributions, the ability to generate formula trees for such value distributions has been added to the framework. It can even generate two-valued formula trees for experimental purposes. Also, it can be used with random values generated from the [0,1] interval using the available accuracy provided by the machine and the programming environment.

The parameter setting to determine the shape of the formula tree has been added to be able to set a simulation environment for various needs since real-life cases show that formula trees can be very variable. In some cases, problems can be described by balanced or roughly balanced trees, like an optimal decision tree, which should have a lower depth for faster decisions. In other cases, the tree can be narrow and deep, e.g. in a communication network, or a wireless sensor network.

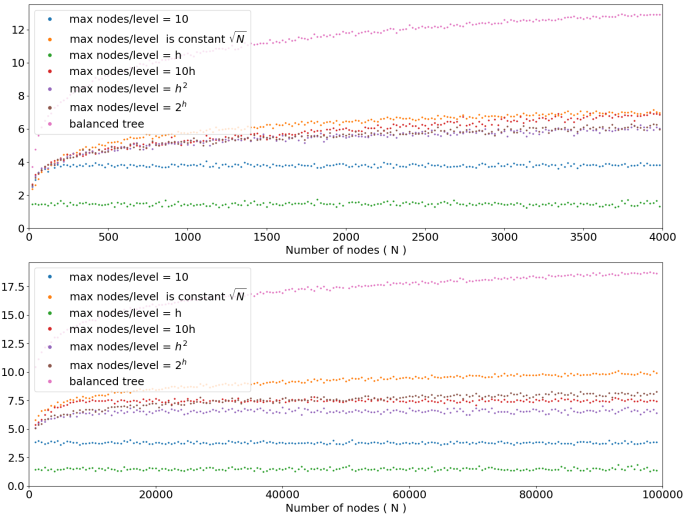
The formula shapes that can be used by the expression tree generator:

- max 10 nodes at each tree level;
- max constant $\sqrt{N}$ nodes at each tree level, where N is the required number of nodes of the tree, given as parameter;
- max h nodes at the tree level of depth h ($h = 1$ for the root);
- max $10h$ nodes at the tree level of depth h;
- max h^2 nodes at tree level h;
- no limit is used (i.e., the natural limit 2^h nodes at depth level h for a binary tree);
- balanced tree, that is, the depth level of the leaves may differ by no more than1.

The last two tree shapes are the most condensed ones, their depth can be thought of as minimal. This is an extremal class in the experiment.

We note that the last two shapes are very similar with the only difference that when generating the tree with the limit 2^h nodes at depth level h, then it will be naturally close to the balanced tree.

The flexible setting ability for the maximal number of child nodes for each node type is provided because of the multiple available fuzzy logic types. The Gödel type logic can by default work with more than two nodes for the MIN and MAX nodes. In the Product logic there is no default extension to have more than two child nodes for the AND and OR nodes but with this parameter, it is possible to experiment with such extensions. The Łukasiewicz logic has weak and strong variants of the conjunction and disjunction operators. We have chosen the strong variants in our research, which are binary operators.



Executing two batches of formula tree generation, above we show the minimum tree depth (level of leaf nodes) for tree sizes up to 4000 nodes, below tree sizes up to 100000 nodes are shown.

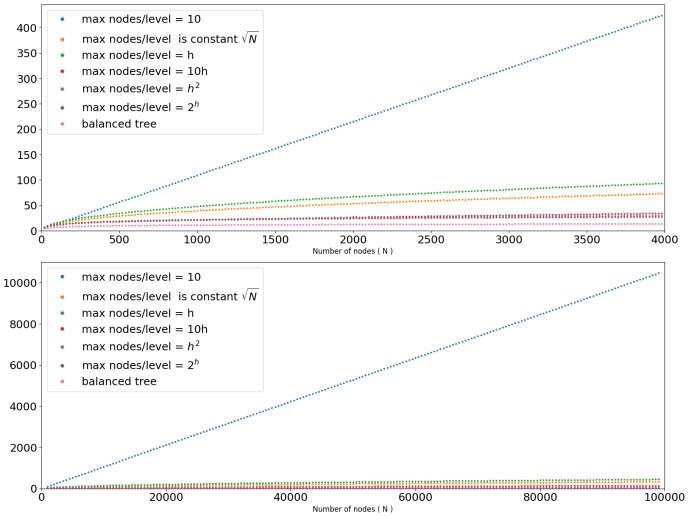
As shown in the figures above and below, the narrowest tree shapes have the smallest minimum depths and the greatest maximum depths, while the widest shapes in the figure have the smallest maximum depths and their average minimum depths are among the highest. This is not a contradiction, leaf nodes can only be closer to the root if other leaf nodes are at a greater distance.

The figure above shows the minimum tree depths as a function of the tree size for different tree shapes.

The minimum depth is very important because short-cuts cannot be made without evaluating at least one leaf node.

The figure below shows the maximum level of leaf nodes on average for the expression trees generated in various shapes.

The maximal depth of expression trees is important as well since there can be threads in the recursive function call flow, where we reach the maximum depth, at which we find the leaf node with the needed value.



Maximum tree depth, above we show the maximum level of leaf nodes for tree sizes up to 4000 nodes, below sizes up to 100000 nodes are shown.

As one can see from the above two figures, the narrowest tree shapes have the lowest minimum depths and the highest maximum depths, while the widest shapes on the figure have the lowest maximum depths and their average minimum depths are among the highest ones. There is no contradiction, there can be leaf nodes closer to the root node only if other leaf nodes will be farther.

Also, we must note for the reader that depths on the figures can be not only integral numbers because many experiments were executed with the same parameters including the number of nodes and we show the average results.

Fast evaluation algorithms

The proposed algorithms for fast evaluation of Gödel fuzzy logic, Product fuzzy logic, and Łukasiewicz fuzzy logic expression trees are shown and described in subchapters 5.1, 5.2, 5.3 of the dissertation. The introduction of the improvements will be shown below.

We use a technique inspired by alpha-beta pruning. Despite the similarity, we will use the terms *lower* and *upper* for the known lower and upper limits of the interval in which in the actual function call the exact value of the node is interesting. If the interval is zero-length then the value of the node can be omitted from the evaluation.

The fast evaluation function must be called with the root node as the first parameter, 0 for *lower* parameter, and 1 for *upper* parameter. The evaluation works recursively, it will

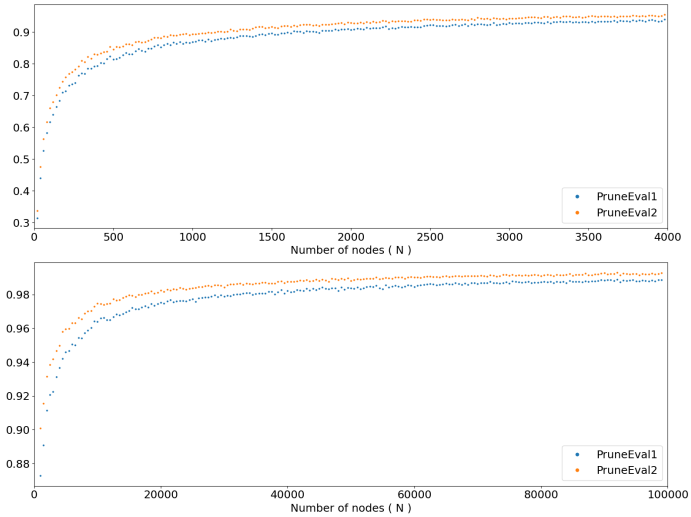
calculate the value of a node reaching its children nodes via recursive function calls, making short-cuts where possible. The *lower* parameter when calling the function for a given node means that for the evaluation of that node it does not matter that the value of it is less than *lower* or equal to *lower*. Similarly, the *upper* parameter means that if we know that the value of the node is at least the value of this parameter then the exact value is not needed.

Pruning the evaluation, i.e. short-cut will be made if the value of the *lower* parameter is higher than or equal to the value of the *upper* parameter, because that means that the value of that node has no effect on the value of the formula tree.

For each of the three examined fuzzy logic types two variants of the pruning algorithms are presented in chapter 5 of the dissertation, the first of which if a simpler algorithm, the second is a more advanced one.

Performance of the fast evaluation algorithms

Some major characteristics of the results will be shown below, the detailed presentation of them can be seen in chapter 6 of the dissertation.



The ratio of the pruned nodes executed on Łukasiewicz type fuzzy logic expression trees nodes. See the result for tree sizes with up to 4000 nodes above and up to 100000 nodes below

The figure shows the ratio of the nodes omitted from the evaluation by the pruning algorithms on Łukasiewicz logic formula tree. About 90% of the nodes were pruned with tree sizes of a few thousands of nodes. For 100k nodes, this ratio was as high as about 99%.

1.3.2 Thesis II: Improvements for neural network classifiers

I proposed simple additions that can be used for machine learning classifiers to possibly achieve better performance.

The proposed fuzzification of training data output class membership values can be used with standalone learners, and with multiple (ensemble) learners as well for better results [8]. This can be useful for datasets where the given training data have crisp (binary) class membership values although the "real" class membership value can be fuzzy. Such datasets can be, e.g., images containing male and female faces, which might be misleading to have binary class membership values for each image, or weather data where e.g. the class "rain" could be fuzzy class as well. In addition to possible performance improvement, this technique has an additional advantage as well. If the algorithm gives fuzzy class membership values then it has an additional information for the classification.

Another proposed addition is defining new variants for committee machine voting functions which in some cases might have better performance compared to the well-known voting functions [9].

These additions can be used separately or together as well.

We note that our experiment was done using convolutional neural network classifiers, however, these techniques might be used for every classifier which can produce fuzzy output values, as well.

The experiments ran on personal computers equipped with NVIDIA and AMD GPUs using Tensorflow from Python programs. Our simple framework was based on a file interface that enables us to run the machine learning on multiple machines, and then later collected and processed the output files generated by the learners.

For the research, two convolutional neural network learning algorithms with different strengths have been selected as the basis of the modifications.

The problem set given to the learning algorithms was the well-known MNIST database of handwritten digits [10]. We plan to perform research on other problem sets as well.

The results may vary given the stochastic nature of the algorithms, so a large number of experiments with different parameters were performed, and average results were analyzed. Since multiple learners have proven to be more successful when we combine their results through voting, we might expect better results in setting the fuzzified class membership together as well. In this experiment ensemble learners in different group sizes were executed. We will show the results of this research. In these experiments, we have measured the standalone test results of the learners, as well as the results of the committee-machine voting functions. When we talk about committee machine voting we can choose from many voting functions, e.g. fuzzy averaging, plurality(or majority) voting, etc. In our research, we have compared the results of some of the most well-known voting functions with our newly defined ones.

For the analyses, we used the Python Numpy and Pandas frameworks.

The algorithms were run with different epoch counts to see the behavior of our proposed algorithm variations not only with the statistically best settings.

1.3.3 Neural network algorithms used in the experiments

For the performance evaluation both in the case of Thesis II/a and Thesis II/b the following neural network algorithms were used as the basis of our proposed advanced techniques.

Neural network algorithm 1

Our first base algorithm was selected to be a simple multi-layer perceptron architecture with 784, 1024, 128, 10 neurons in the successive layers. The activation function of the hidden layers was the relu, the output activation function was the softmax. This algorithm can be considered a weak learner, compared to the current state-of-the-art algorithms.

Neural network algorithm 2

The second neural network algorithm for our experiments was a modified variant of a convolutional neural network [11]. This algorithm can be considered a strong one, although not as much as the current state-of-the-art algorithms.

Neural network algorithm 3

The third selected algorithm to add our improvements to was a modified algorithm of [12], using the parameters for the fuzzification. The number of epochs we executed our algorithm was set from 15 to 20, to eliminate the effect of a statistically chosen best epoch count for a specific data set.

1.3.4 Thesis II/a

Neural network and convolutional neural network algorithms are powerful machine learning tools for solving classification or other problems. Their performance can certainly be influenced by the quality of the training data. One common problem is that training data usually has binary output values, even when the training samples may belong to more than one class at a certain fuzzy level. These data come usually labeled so that each sample has one or more labels, each of which means the crisp `True` membership in the class behind that label, and crisp `False` membership value for the other classes in the same category. There can be cases where these crisp class membership values can be considered misleading, so the correction of these values can lead to reducing the confounding effect of them. Modification of training data is often useful for regularization. This can be done by e.g. making distortion, adding noise, using data augmentation, or adversarial training.

The proposed fuzzification technique might be applied to other classifying algorithms as well, in case they are able to give fuzzy membership values in their output. Research on other algorithms in order to apply the fuzzification technique on them can be future research, in the current research we conducted our measurements with neural network algorithms.

We define simple methods that can be used to modify the target output values given for the training patterns during the training process to get fuzzy output values from the crisp (binary) values of the training data set. This class membership fuzzification is done so that the knowledge gained during the learning process will be used to correct the inaccurate output class membership values of the training patterns.

Three versions of this algorithm are presented in the dissertation. The first of them (Algorithm 7) is for single learners, the second version (Algorithm 8) is for multiple learners the result of which can be used with committee machine voting functions, the third variant (Algorithm 9) is a simple modification to handle the parameters of the fuzzification for multiple learners. Here only the variant Algorithm 8 will be shown.

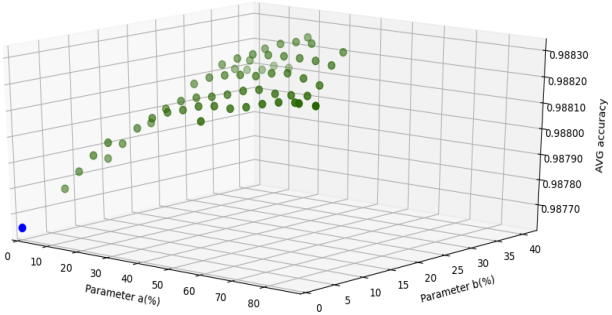
```
procedure FuzzyTrainingEnsemble(models,train_X,train_Y,FA,FB,FC):
  epoch = 0
  fuzzy_Y = train_Y
  while epoch<MAX_EPOCHS and CheckEarlyStopCondition()==False:
    for model in models:
      model.fit(train_X, fuzzy_Y, epochs=1)
      out = model.predict(train_X)
      if epoch >= START_FUZZY:
        fuzzy_Y = FA*fuzzy_Y + FB*out + FC*train_Y
    epoch = epoch + 1
end procedure
```

The algorithm must be called with the learners' models, the training inputs and outputs, and the parameters for the fuzzification for the training of a learning model. The parameter FA is the weight for the momentum which means the importance of the actual (current) class membership values (the $fuzzy\_Y$ vector). This affects the change from original values towards the desired values giving the momentum for the actual knowledge. The parameter FB is the weight for the current knowledge (the *out* vector), which means the courage to change. The parameter FC is the weight used for the $train\_Y$ vector, which means the importance of the original target output data. The sum of the parameters FA, FB, and FC must be 1.0. In the above-presented algorithm, it is a simple condition to have some epochs before the first correction. This, of course, can be changed to an adaptive condition to achieve better performance, however, for our measurement, it is more important to know the number of correction operations. When the learning starts, the initial values in the $fuzzy\_Y$ vector are the same as given in the $train\_Y$ vector.

Performance of the proposed neural network class membership value fuzzification

The performance evaluation of my proposed training data class membership value fuzzification is shown and described in detail in chapter 8.1 of the dissertation. The accuracy of the predictions on test data with three different neural network algorithms for our experiment will be shown below.

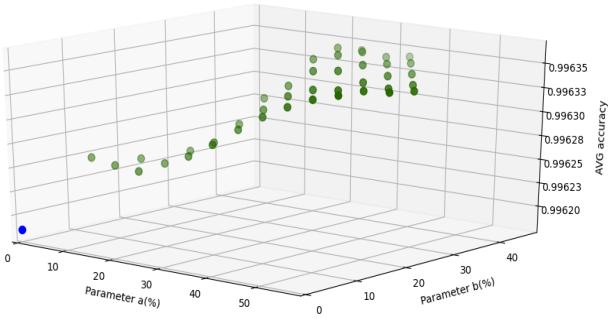
First, we show evidence of the efficiency of our proposed fuzzification algorithm using the Neural network algorithm 1 thousands of learners learned 20 epochs in ensembles of 10 learners. The ensembles had different parameters for fuzzification, including parameters that keep the original class membership values ($FA = 0, FB = 0, FC = 1$). Input data distortions were applied during the training.



The average performance results of our algorithms on test data, using different parameters for the fuzzification of the training data class membership values.

The figure above shows the average individual accuracy on test data for the FA and FB parameters (parameter values shown in %), the third parameter $FC = 1 - (FA + FB)$. The value with $FA = 0, FB = 0$ coordinates shows the average result without fuzzification. We can see that with values of parameter FB around 40% we had better accuracy, especially when the value of parameter FA was close to 20%. This means that the stronger fuzzification with a stronger momentum factor resulted in better accuracy. The difference was significant.

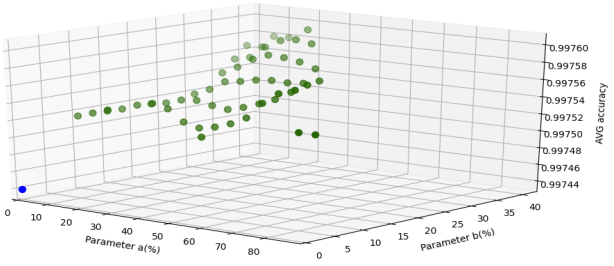
The second experiment was performed using the Neural network algorithm 2. Thousands of learners learned different number of epochs and with different parameters for fuzzification, including parameters that keep the original class membership values ($FA = 0, FB = 0, FC = 1$).



The average accuracy results of our algorithms on test data using different parameters for the fuzzification of the training data class membership values.

The figure above shows the average accuracy of the algorithm on test data for the various FA and FB parameters ($FC = 1-(FA+FB)$). The result with coordinates $FA = 0, FB = 0$ shows the average result without fuzzification. The stronger fuzzification resulted in better accuracy, especially when the value of the FA parameter setting the factor of the current knowledge (momentum) was between 20% and 30%. The difference between the individual accuracy of the algorithms with and without fuzzification was significant.

In the third experiment, we executed the learning sessions using the Neural network algorithm 3 as basis of our modifications.



The results of our algorithms average accuracy on test data using different parameters for the fuzzification of the training data class membership values.

This figure shows the results of thousands of learning sessions which were executed with different FA, FB and FC parameters. The point with $FA = 0, FB = 0$ coordinates shows the

average result when class membership values of training data were not corrected. We can see that the results were higher with lower FA and FB parameter values ($FC = 1-(FA+FB)$). For such parameters the FC parameter is higher, so only minor corrections on the training data class membership values can be made.

Similar to the previous experiments the performance of fuzzification was significant with this strong algorithm, as well. Although the algorithm without performing fuzzification has a good performance, adding our method to correct the class membership of the training data improved the accuracy of the individual learners. The stronger fuzzification, i.e. larger value for the factor of the actual output of the learner, resulted in higher accuracy.

1.3.5 Thesis II/b

I described some well-known committee machine voting functions in subchapter 3.11 of the dissertation. Their good performance motivated us to develop our new ones. We defined the following new committee machine voting functions which we will compare with some of the well-known voting functions. Some of them belong to the locally weighted average voting functions.

- **Fuzzy average voting weighted by confidence (V2 in the dissertation):**

  Fuzzy average voting can be weighted by confidence [13]. Here I propose a simple function with getting confidence from the class membership values. This method obviously needs less performance compared to other more advanced methods. Class membership values closer to 0 or 1 will have a stronger weight, we transform the output of the individual learners before calculating the fuzzy average so that the values which are considered uncertain (not close to 0 or 1) will be less important by multiplying with a lower weight. Given the network output $o_i[j]$ for each i learner for each j class we calculate the combined result with the following formula with N learners:

  $$o[j] = \frac{1}{N}\sum_{i=1}^{N}((o_i[j]-0.5)*(2*o_i[j]-1)^2+0.5)$$

  Then we get the winner class from this weighted average:

  $$l = argmax(o)$$

- **Fuzzy average voting weighted by 1-difference from the combined output (V3 in the dissertation):**

  Knowing the outputs of N learners, we can base another weighted average method based on the better performance of the fuzzy average voting compared to the individual learners. Starting with the calculation of the fuzzy average, individual predictions will be multiplied by a weight which is the difference from the ensemble prediction subtracted from 1. Let $o[j]$ for each j class be calculated as usual for the fuzzy voting. Then we calculate the new variant with N learners as follows:

  $$o'[j] = \frac{1}{N}\sum_{i=1}^{N}((o_i[j]-0.5)*(1-|o_i[j]-o[j]|)+0.5)$$

We can find the winner class from the weighted average:

$$l = argmax(o')$$

- **Fuzzy average voting weighted by the reciprocal value of the number of failed training samples (V4 in dissertation):**

  Let f_i be the number of failed (misclassified) samples for each learner i on the training dataset if it is not equal to 0, otherwise, we use a value between 0 and 1, e.g. 0.5.

  The reciprocal value of f_i will be used as the weight for the learner i.

$$o[j] = \left(\frac{1}{N}\sum_{i=1}^{N}\frac{o_i[j]}{f_i}\right)\left(\frac{1}{N}\sum_{i=1}^{N}f_i\right)$$

  From this weighted average we get the winner class:

$$l = argmax(o)$$

Meta-voting variants

Fuzzy average or plurality vote by combining selected voting functions by calculating the fuzzy average or the plurality of votes on the classes of the results of the selected voting functions.

For analysis purposes, we define three meta voter variants.

- V8: Plurality voting from the results of V1,V2,V3,V4,V5,V6,V7
- V9: Plurality voting from the results of V1,V2,V3,V4,V7
- V10: Fuzzy average voting from the results of V1,V2,V3,V4,V7

For the above three meta voting functions we calculate the results of the required voting functions first, then we combine them as it was described earlier for the voting functions calculated from the results of the individual learners.

Performance evaluation of the proposed voting functions

voting function	MIN	AVG	MAX
min(accuracy)	0.981700	0.986263	0.990600
avg(accuracy)	0.983600	0.988148	0.990900
max(accuracy)	0.983800	0.989761	0.991900
V1–fuzzy average	0.987600	0.992755	0.994800
V2–weighted by confidence	0.987600	0.992769	0.994700
V3–weighted by diff from V1	0.987600	0.992752	0.994800
V4–weighted by 1/training fails	0.987600	0.992761	0.994700
V5–plurality voting	0.981100	0.992449	0.994800
V6–borda voting	0.982000	0.992537	0.994600
V7–geometric mean voting	0.987500	**0.992790**	0.994800
V8–meta–plurality (V1-V7)	0.987600	0.992764	0.994800
V9–meta–plurality (V1-V4,V7)	0.987600	0.992765	0.994800
V10–meta–fuzzy avg (V1-V4,V7)	0.987700	0.992776	0.994800

Minimum, maximum, and average accuracy for ensemble learning voting experiment with the modified version of Neural network algorithm 1

The table above shows the results where $5 - 20$ voters computed their predictions in each turn. Their predictions were then combined using the voting functions V1-V10. The best individual result was 81 misclassification on 10000 test samples, the worst individual result was 183 failed samples and on average they performed only 118.52 fails as individual learners. The well-known fuzzy voting performed 72.45 misses on average. There were no big differences among the voting functions, the best average result (72.1 fails on average) was achieved by the geometric mean (V7) voting function.

voting function	**MIN**	**AVG**	**MAX**
min(accuracy)	0.992700	0.995422	0.996900
avg(accuracy)	0.995188	0.996306	0.997260
max(accuracy)	0.995600	0.997043	0.998000
V1–fuzzy average	0.996000	0.997351	0.998400
V2–weighted by confidence	0.995900	0.997360	0.998300
V3–weighted by diff from V1	0.996000	0.997346	0.998300
V4–weighted by 1/failures	0.995500	0.997321	0.998300
V5–plurality voting	0.995500	0.997280	0.998400
V6–borda voting	0.995600	0.997296	0.998400
V7–geometric mean voting	0.996000	**0.997367**	0.998300
V8–meta–plurality (V1-V7)	0.995900	0.997353	0.998400
V9–meta–plurality (V1-V4,V7)	0.996000	0.997356	0.998400
V10–meta–fuzzy avg (V1-V4,V7)	0.996100	0.997350	0.998300

Minimum, maximum, and average accuracy for ensemble learning voting experiment with the modified version of Neural network algorithm 2

The table above shows the accumulated results of the tested voting functions with the minimum, average, and maximum number of the failed samples of the individual learners included. The best individual result was 20 fails on 10000 test samples, the worst was 73 failed samples and on average they performed as low as 36.94 fails from 10000 samples as individual learners. The well-known fuzzy voting performed 26.49 fails on average. There were no big differences among the voting functions, the best result was given by the product voting (i.e. geometric mean–V7) from committee results on training failures.

voting function	**MIN**	**AVG**	**MAX**
min(accuracy)	0.996200	0.996878	0.997700
avg(accuracy)	0.996840	0.997418	0.997886
max(accuracy)	0.997100	0.997914	0.998500
V1–fuzzy average	0.997300	0.998126	0.998700
V2–weighted by confidence	0.997300	0.998122	0.998700
V3–weighted by diff from V1	0.997200	0.998128	0.998700
V4–weighted by 1/failures	0.997300	0.998126	0.998700
V5–plurality voting	0.997000	0.998044	0.998700
V6–borda voting	0.997000	0.998044	0.998700
V7–geometric mean voting	0.997100	0.998126	0.998700
V8–meta–plurality (V1-V7)	0.997200	0.998125	0.998700
V9–meta–plurality (V1-V4,V7)	0.997200	0.998128	0.998700
V10–meta–fuzzy avg (V1-V4,V7)	0.997200	**0.998129**	0.998700

Minimum, maximum, and average accuracy for ensemble learning voting experiment with the modified version of Neural network algorithm 3

The table above shows the results where $5-20$ voters voted in each turn using the above-defined voting functions. The best individual result was 15 fails from 10000 test samples, the worst individual result was 38 failed samples and on average they performed only 25.82 fails as individual learners. The well-known fuzzy voting performed 18.74 on average. There were no big differences among the voting functions, the best result (18.71 fails) came from our meta fuzzy voting function (V10).

2. Fuzzy logika és neurális hálózat algoritmusok hatékonyságának növelése

2.1. Bevezetés és motiváció

A fuzzy logika és a mesterséges neurális hálózatok a modern mesterséges intelligencia két területe, amelyek különféle algoritmusokat biztosítanak a tanulás képességével, és döntési támogatást nyújtanak számunkra.

A Fuzzy logika kibővítette a bináris logikát, hogy több igazságértékkel rendelkezzen. A fuzzy logikát számos alkalmazásban használják, például gépi vezérlés, tudásalapú rendszerek, optimalizálási problémák, időjárás-előrejelzés, kockázatértékelés, orvosi diagnózis, stb. Néhány alkalmazásának nagyon nagy formulák feldolgozására lehet szüksége, pl. gépi vezérlés és gépi tanulás, ahol nagy mennyiségű adat állhat rendelkezésre az algoritmusaink futtatásához, ezért jó hatékonyságuk elengedhetetlen. A rövidzár kiértékelési technika alkalmazható fuzzy logikai kifejezésekre is. A fuzzy logikai formulák rövidzár-kiértékelésének kiterjesztése tovább javíthatja a teljesítményt, ez motivált erre a kutatásra.

Az egyik legelterjedtebb gépi tanulási algoritmus a Mesterséges neurális hálózat vagy annak mély és konvolúciós változatai. A neurális hálózati algoritmusok felügyelt gépi tanulási algoritmusok, amelyeket széles körben használnak a gépi tanulásban. Fő alkalmazásai közé tartozik az osztályozás, a regresszió, a mintafelismerés, a függvény-közelítés, az intelligens vezérlés, az adatokból való tanulás.

A neurális hálózat és a konvolúciós neurális hálózat algoritmusai a legjobban teljesítő gépi tanulási algoritmusok közé tartoznak. Az osztályozáshoz használt neurális hálózat egyik ismert kérdése, hogy a tanulási adatok általában bináris kimeneti értékekkel rendelkeznek, még akkor is, ha a minták fuzzy módon egynél több osztályba tartozhatnak. Vannak olyan esetek, amikor ezek a bináris osztálytagsági értékek félrevezetőnek tekinthetők, így ezen értékek korrekciója csökkentheti ezek zavaró hatását. Egy egyszerű módszert dolgoztam ki az éles (bináris) osztálytagsági értékek fuzzifikációval történő kijavítására.

A neurális hálózati algoritmusok teljesítménye több futtatás között változhat, mivel ezek az algoritmusok sztochasztikus jellegűek. Ez a sztochasztikus viselkedés az átlagosnál rosszabb pontosságot eredményezhet egyetlen futás során, és sok esetben nehéz eldönteni, meg kelle ismételnünk a tanulást, esélyt adva a jobb eredmény elérésére. A probléma megoldására szolgáló hasznos technikák között alkalmazhatjuk a bizottság-gépet és az együttes módszereket, amelyek sok esetben az átlagnál jobb, sőt gyakran a legjobb egyéni eredménynél is jobb eredményt adnak. Számos jól ismert technika létezik erre, kutatásom eredményeképpen új módszereket javasoltam az együttes tanuló algoritmusok számára.

A gépi tanulási algoritmusok fejlesztései kibővíthetik a jelenleg elérhető számítógépeken futtatható problémák körét. A jobb hatékonyság hozzájárulhat ahhoz, hogy alacsonyabb teljesítményű hardvereken is használhatók legyenek. Ez hatékonyabbá teheti a gépi tanulás alkalmazását.

2.2. Elsődleges célok

A rövidzár-kiértékelést széles körben használják a programozási nyelvekben a logikai kifejezések hatékony értékeléséhez. A kifejezések egyes részei kihagyhatók az értékelésből, ha azok nem befolyásolják a kifejezés végső értékét. Nagy kifejezések esetén ez nagyon hatékony lehet. Kutatást végeztem a fuzzy logic kifejezésfák gyors kiértékelési algoritmusainak kifejlesztésére. Célunk az volt, hogy megtaláljuk azokat a technikákat, amelyek a rekurzív kiértékelési algoritmusban rövidítéseket hoznak létre a triviális rövidítési lehetőségek mellett, amelyeket a programozási nyelvek interpreterei vagy fordítói használnak logikai kifejezések kiértékelésére. Kutatásom és fejlesztésem arra irányult, hogy a Gödel fuzzy logika, a Product fuzzy logika és a Łukasiewicz fuzzy logika kifejezések nagy méretű fáinak kiértékelésére hatékony algoritmus(oka)t alkossak úgy, hogy az értékelésből kihagyjuk azokat a csúcsokat és gyermekeiket, amelyek értéke nem befolyásolja a kifejezés végeredményét [14, 5, 6].

Célom egy olyan rugalmas kifejezésfa-generátor kifejlesztése is volt, amely felhasználható formulafák előállítására a kiválasztott fuzzy logikai típushoz, lehetővé téve a generált fák alakjának beállítását, valamint paramétert biztosítva a levélcsúcsok értékeloszlásának beállításához.

A tervezett teljesítményértékeléshez rugalmas keretrendszerre is szükség volt, amely segítségével a kifejezésgenerátor és a gyors kiértékelő algoritmus több változata futtatható a mérések elvégzéséhez.

A bizottság-gépek nagyon hatékonyak a gépi tanulásban. Kutatást folytattam a neurális hálózat és a konvolúciós neurális hálózat algoritmusok fejlett technikáival kapcsolatosan, három különböző modell felhasználásával a teljesítmény értékeléséhez. Megvizsgáltam a bináris osztálytagsági értékek fuzzifikációjának alkalmazhatóságát a tanulás során [8], a zajos vagy esetleg rosszul besorolt tanulási minták kijavítására a jobb pontosság érdekében. A fő cél a tanulási adatok osztálytagsági értékeinek korrekciója volt, ahol az adott bináris osztálytagsági értékek félrevezetőek lehetnek egyes minták esetében.

Kutatást folytattam új bizottság-gépi (együttes) algoritmusok és variációk kifejlesztésére is [9]. Vizsgáltam a bizottság-gépek egyszerű szavazási funkcióinak új változatainak és új meta szavazási funkciónak a lehetőségét, amelyek az egyszerű szavazási funkciókat használhatják bemenetként [9].

2.3. Tézisek és eredmények

2.3.1. Tézis I: A nagy fuzzy logikai kifejezésfák gyors értékelése

Új algoritmusokat dolgoztam ki a Gödel típusú fuzzy logika, a Product fuzzy logika és Łukasiewicz fuzzy logika formulák kifejezés-fáinak kiértékelésére (lásd a disszertáció 5.1, 5.2, 5.3 alfejezetei) [5, 6]. Az algoritmusok úgy működnek, hogy az aktuális csúcs kiértékelését alsó és felső határokkal korlátozzák, hogy beállítsák azt az intervallumot, amelyben a csúcs pontos értéke érdekes. Ezzel a technikával egy nagy kifejezésfa csúcsjai magas arányban leválghatók a kiértékelésből anélkül, hogy ez befolyásolná az értékelés végeredményét. Ez ahhoz vezet, hogy az optimalizált algoritmus kevesebb teljesítményt igényel a kifejezésfa értékeléséhez.

A javasolt algoritmusok közül itt csak egyet mutatok be (PruneEval1 a Łukasiewicz formula fák kiértékelésére).

```
function PruneEval1(node, lower, upper):
  if lower >= upper:
    return v = lower # cut

  if nodetype == NODE_TYPE_NEG:
    v = 1 - PruneEval1(child, 1-upper, 1-lower)

  elif nodetype == NODE_TYPE_AND:
    v1 = PruneEval1(child1, lower, 1)
    v2 = PruneEval1(child2, min(1+lower-v1,1), min(1+upper-v1,1))
    v  = max(v1+v2-1, 0)

  elif nodetype == NODE_TYPE_OR:
    v1 = PruneEval1(child1, 0, upper)
    v2 = PruneEval1(child2, max(lower-v1,0), max(upper-v1,0))
    v  = min(v1+v2, 1)

  elif nodetype == NODE_TYPE_IMP:
    v1 = PruneEval1(child1, 1-upper, 1)
    v2 = PruneEval1(child2, max(lower-1+v1,0), max(upper-1+v1,0))
    v  = min(1-v1+v2, 1)

  else: # NODE_TYPE_LEAF
    v = node value

  return v
end function

value_of_the_expression = PruneEval1(root, 0, 1)
```

Az algoritmus úgy indul, hogy meghívjuk a bemeneti formulafával, amelyet a gyökércsúccsal adunk meg és két paraméterrel: *lower* és *upper*. Kiszámítja a kifejezés y értékét, ha $lower \leq y \leq upper$. Az érdekes intervallumot a *lower* és *upper* értékek képviselik, alsó és felső határként, egy csomópont v értékéhez úgy, hogy ez a v érték akkor befolyásolja az egész kifejezés eredményét, ha a v az érdekes intervallumban van. Ez az intervallum a kifejezés(fa) már elemzett részétől függ. Az értékelés során a vágásokat úgy hajtjuk végre, hogy a *lower* és *upper* paramétereket dinamikusan állítjuk be a futás során a formulafa minden csomópontjára. Pontosabban, a *lower* és *upper* szerepe a következő.

Az aktuális csomópont gyermekeinek értékelése során csak akkor kell folytatnunk a kiértékelést, ha értékük a kiértékelésükhöz megadott tényleges *lower* és *upper* határ között lehet. Intuitív módon, amikor $lower >= upper$, akkor nincs intervallum közöttük, így az adott csomópont levágható, a fő kifejezés értéke nem függ e csomópont értékeitől.

A kifejezés rekurzív kiértékelését egyszerű rekurzív függvényhívással végezzük.

A fő hívás az algoritmus utolsó sorában látható.

Kifejezésfák generálása

A kutatáshoz kidolgoztam egy egyszerű keretrendszert nagyszámú kifejezésfa előállításához, amelyeken az értékelési algoritmusok futtathatók. A keretrendszert általános célokra hoztam létre, lehetővé teszi paraméterekkel az értékek eloszlásának szabályozását a $[0, 1]$ in-

tervallumban, a fa alakjának, a csúcstípusok számára megengedett maximális gyermekcsúcsok számának és a csúcsok számának megadását [5, 7] az alábbiak szerint.

Mivel a fuzzy logika típusok három vagy több értékű, sokértékű, végtelenül sokértékű disztribúcióval rendelkeznek, a keretrendszerbe beépítettem a képességet, hogy kifejezésfákat generáljunk az ilyen értékeloszlásokhoz. Akár kétértékű kifejezésfákat is létrehozhat kísérleti célokra. Használható a [0,1] intervallumból generált véletlenszerű értékekkel is, a gép és a programozási környezet által biztosított pontossággal.

A kifejezésfa alakjának meghatározására szolgáló paraméterbeállítás lehetővé teszi, hogy a szimulációs környezetet különféle igényekhez lehessen igazítani, mivel a valós élet esetei azt mutatják, hogy a kifejezésfák nagyon változékonyak lehetnek. Bizonyos esetekben a problémákat kiegyensúlyozott vagy nagyjából kiegyensúlyozott fák írhatják le, például egy optimális (fordított) döntési fa, amelynek csekély mélységűnek kell lennie a gyorsabb döntésekhez. Más esetekben a fa lehet keskeny és mély, például egy kommunikációs hálózatban vagy egy vezeték nélküli szenzor hálózatban.

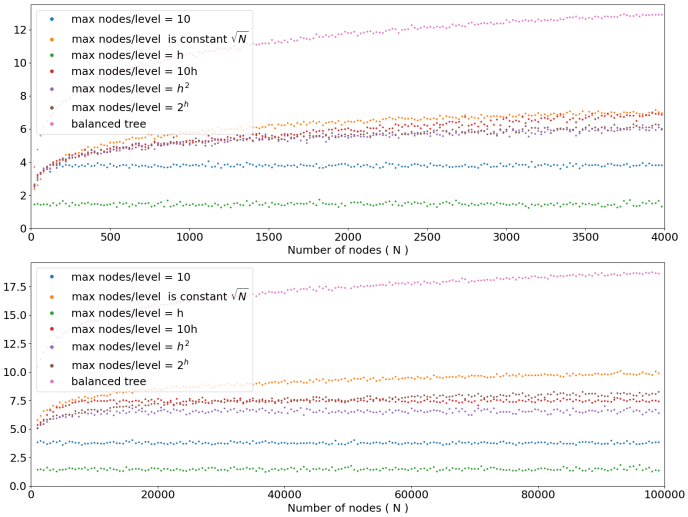
A kifejezésfa-generátor által használható szabályok:

- max 10 csúcs a fa minden szintjén;
- max $\sqrt{N}$ elem a fa minden szintjén, ahol N a fa mérete, operátor csúcsokat és levélcsúcsokat beleértve;
- max h csúcs a fa h. szintjén ($h = 1$ a gyökér esetén);
- max $10h$ csúcs a fa h. szintjén ;
- max h^2 csúcs a fa h. szintjén ;
- nincs korlát, (azaz a 2^h csúcs a fa h. szintjén bináris fa esetén);
- kiegyensúlyozott fa, azaz a levélcsúcsok szintjei közötti különbség maximum 1 lehet.

Az utolsó két faalak a leginkább kompakt, mélységük minimálisnak tekinthető. Ez egy extrém osztály a kísérletben.

Megjegyezzük, hogy az utolsó két alakzat nagyon hasonló, azzal az egyetlen különbséggel, hogy ha a fát a 2^h határértékkel hozzuk létre, akkor a csúcsok természetesen közel állnak a kiegyensúlyozott fához.

Az egyes csúcstípusokhoz tartozó gyermekcsúcsok maximális számának rugalmas beállítási képessége a több választható fuzzy logikai típus miatt biztosított. A Gödel típusú logika alapértelmezés szerint kettőnél több csúcsral működhet a MIN és MAX csúcsoknál. A Product logikában nincs alapértelmezett kiterjesztés, amely kettőnél több gyermek csúcsot tartalmazna az AND és OR csúcsok számára, de ezzel a paraméterrel kísérletet lehet folytatni ilyen kiterjesztésekkel. A Łukasiewicz gyenge és erős variánsokkal rendelkezik a konjunkció és a diszjunkció operátorok tekintetében. Kutatásunk során az erős változatokat választottuk, amelyekben a konjunkció és a diszjunkció bináris operátorok.

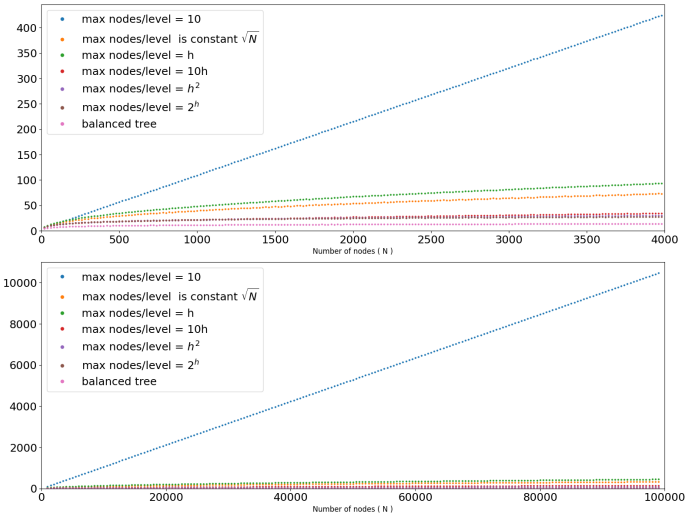


A formula fák minimális mélysége, a levélcsúcsok minimális szintjét ábrázoltuk, fent 4000 csúccsal rendelkező fa méretig, lent pedig pedig 100000 csúcsig.

Mint az a fenti és az alább bemutatott ábrákon látható, a legkeskenyebb faalakoknál a legkisebb a minimális mélység és a legnagyobb a maximális mélység, míg az kifejezésfák legszélesebb változatainál a legkisebb a legnagyobb mélység, és az átlagos legkisebb mélységük a legmnagyobbak között van. Ez nem ellentmondás, csak akkor lehetnek levélcsúcsok a gyökérhez közelebb, ha más levélcsúcsok nagyobb távolságra kerülnek.

A minimális mélység nagyon fontos a kiértékelés hatékonysága szempontjából, mert a vágásokat nem lehet elvégezni legalább egy levélcsúcs kiértékelése nélkül. A fenti ábra a csúcsok átlagos minimális szintjét mutatja.

Az alábbi ábra mutatja a levélcsúcsok maximális szintjének átlagos értékeit, hiszen fontos a kifejezés-fák maximális mélysége is, ugyanis a rekurzív függvényhívási folyamatban lehetnek szálak, ahol elérjük a maximális mélységet, amelynél megtaláljuk a kiértékeléshez szükséges levélcsúcsot.



A kifejezésfák maximális mélysége, a levélcsúcsok maximális szintjét mutatjuk fent 4000 csúccsal rendelkező fa méretig, lent pedig pedig 100000 csúcsig.

Gyors kiértékelési algoritmusok

A Gödel fuzzy logika, a Product fuzzy logika és a Łukasiewicz fuzzy logika formula fák gyors kiértékelésére javasolt algoritmusokat a disszertáció 5.1, 5.2, 5.3 alfejezetei mutatják be és írják le. Az algoritmus főbb tulajdonságait az alábbiakban mutatjuk be.

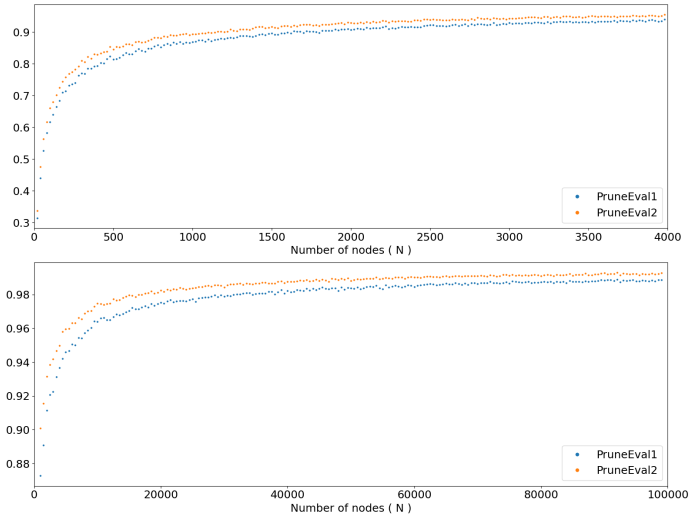
Egy az alfa-béta vágás által inspirált technikát alkalmazunk. A hasonlóság ellenére az alfa és béta helyett az alsó (lower) és felső (upper) limit kifejezéket fogjuk használni annak az intervallumnak az ismert alsó és felső határaihoz, amelyben a tényleges függvényhívásban érdekes a csúcs pontos értéke. Ha az intervallum nulla méretű, akkor a csúcs értéke kihagyható az értékelésből.

A gyors kiértékelési függvény meghívásakor a gyökércsúcs az első paraméter, ezután 0 értéket adunk a *lower* (alsó limit) paraméternek és 1 értéket az *upper* (felső limit) paraméternek. Az értékelési folyamat rekurzív, a csúcs értékét a gyermek csúcs(ok)ig elérő rekurzív függvényhívások segítségével határozzuk meg, vágást végezve, ahol lehetséges. Az alsó limit paraméter a függvény hívásakor az adott csúcs esetében azt jelenti, hogy az értékelésnél az adott csúcs esetében nem számít, hogy az értéke kisebb ennél az értéknél, vagy egyenlő vele. Hasonlóképpen a felső limit paraméter azt jelenti, hogy ha tudjuk, hogy a csúcs értéke legalább ennek a paraméternek az értéke, akkor a pontos értékre nincs szükség.

Az értékelés vágása, azaz rövid-zár történik, ha az alsó limit paraméter értéke nagyobb vagy egyenlő a felső paraméter értékével, mert ez azt jelenti, hogy az adott csúcs értéke nincs hatással a formulafa értékére.

A három vizsgált fuzzy logikai típus mindegyikéhez a vágással való kiértékelés algoritmusának két változatát mutatjuk be a dolgozat 5. fejezetében, amelyek közül az első egy nagyon egyszerű algoritmus, a második egy fejlettebb.

Az eredmények néhány főbb jellemzőjét az alábbiakban mutatjuk be, ezek részletes bemutatása a disszertáció 6. fejezetében található.



A Łukasiewicz típusú fuzzy logikai kifejezésfákon végrehajtott kiértékelésnél kihagyott csúcsok aránya. Lásd fent az eredményt azoknál a legfeljebb 4000 csúcsos fák, és lent a legfeljebb 100000 csúcsot tartalmazó fák esetén

A fenti ábrán látható, hogy a csúcsok mekkora hányadát hagyták ki a kiértékelésből a Łukasiewicz logikai kifejezésfa gyorskiértékelő algoritmusai.

A metszési arány a különböző, eltérő tulajdonságú fáknál különböző lehet. A kifejezésfák értékeloszlása és alakja erősen befolyásolja a vágási arányt. Ezt a viselkedést a disszertáció 6. fejezete mutatja be és írja le.

2.3.2. Tézis II: Fejlesztések a neurális hálózat osztályozókhoz

Javasoltam egyszerű kiegészítéseket, amelyek a gépi tanulás osztályozói számára használhatók a jobb teljesítmény elérése érdekében [8].

A tanulási adatok kimeneti osztálytagsági értékeinek javasolt fuzzifikálása önálló tanulókkal és több (együttes) tanulóval is használható. Ez hasznos lehet olyan adatok esetén, ahol

a tanulási adatok adott bináris osztálytagsági értékekkel rendelkeznek, bár a "valós" osztálytagsági érték fuzzy lehet. Ilyen adatkészlet lehet pl. férfi és női arcokat tartalmazó képek, amelyek megtévesztőek lehetnek, ha mindegyik kép bináris osztálytagsági értékekkel rendelkezik, vagy időjárási adatok, ahol például az "eső" osztály fuzzy osztályú is lehet. A lehetséges teljesítményjavítás mellett ez a technika további előnnyel is járhat. Ha az algoritmus fuzzy osztálytagsági értékeket ad, az plusz információval bír az osztályozáshoz.

Egy másik javasolt kiegészítés a bizottság-gépi szavazási funkciók új változatainak definiálása, amelyek bizonyos esetekben jobb teljesítményt nyújthatnak a jól ismert szavazási funkciókhoz képest [9].

A fenti kiegészítések külön-külön vagy együtt is alkalmazhatók.

A kísérleteket neurális hálózat osztályozókkal végeztem, azonban ezek a technikák alkalmazhatók minden olyan osztályozóhoz, amely fuzzy output értékeket is képes előállítani.

A kísérletek NVIDIA, illetve AMD GPU-val felszerelt személyi számítógépeken futottak, a Tensorflow keretrendszerrel, Python programok felhasználásával. Egyszerű keretrendszerünk fájl interfészen alapult, amely lehetővé teszi a gépi tanulás futtatását több gépen, majd később összegyűjti és feldolgozza a tanulók által generált kimeneti állományokat.

A kutatáshoz egy egyszerű többrétegű perceptron (MLP) és két konvolúciós neurális hálózat tanulási algoritmust választottam különböző erősséggel a módosítások alapjául.

A tanulási algoritmusok problémaköre a kézírásos számok jól ismert MNIST adatbázisa volt [10]. Tervezem, hogy kutatásokat folytatok más feladatokon is.

Az eredményeket befolyásolja a tanuló algoritmusok sztochasztikus természete, így a kísérletekhez nagyszámú futtatást végeztem különböző paraméterekkel, és az eredményt analizáltam. Mivel több tanuló sikeresebbnek bizonyulhat, amikor eredményeiket a szavazáson keresztül kombináljuk, jobb eredményekre számíthatunk, ha a fuzzifikált osztálytagságot is közösen állítják elő. Ebben a kísérletben különböző csoportméretű együttes tanulókat futtattam. A kutatás eredményeit a disszertáció 8.2-es alfejezetében mutatom be. Ezekben a kísérletekben megmértem a tanulók önálló teszt eredményeit, valamint a bizottságok szavazásának eredményeit. Amikor bizottság-gépi szavazásról beszélünk, számos szavazó függvény közül választhatunk, pl. fuzzy átlag, többségi szavazás, stb. Kutatásom során összevetettem néhány legismertebb szavazási funkció eredményeit az újonnan definiáltakkal.

Az elemzésekhez a Python Numpy és a Pandas keretrendszert használtam.

Az algoritmusok különböző tanulási ciklus számban futottak, hogy a viselkedés analizálása ne csak a statisztikailag legjobb beállításokkal történjen.

Algoritmusok a javasolt fejlesztések bemutatásához

A teljesítményértékeléshez mind a II/a, mind a II/b tézis esetében a következő neurális hálózati algoritmusokat használtam, amelyekre a javasolt fejlett technikákat alkalmaztam.

1. Neurális hálózat algoritmus

Az első alapalgoritmusunk egy egyszerű, többrétegű perceptron architektúra lett, amelynek 784, 1024, 128, 10 neuronja volt az egymást követő rétegekben. A rejtett rétegek aktiválási funkciója a relu, a kimeneti aktiválási funkció pedig a softmax volt. Ez az algoritmus gyenge tanulónak tekinthető, a jelenlegi legmodernebb algoritmusokhoz képest.

2. Neurális hálózat algoritmus

Kísérleteim második neurális hálózati algoritmusa egy konvolúciós neurális hálózat módosított változata volt [11]. Ez az algoritmus erősnek tekinthető, bár nem annyira, mint a jelenlegi "state-of-the-art" algoritmusok.

3. Neurális hálózat algoritmus

A harmadik választott algoritmus, amelyre alkalmaztam a fejlesztéseimet, szintén egy konvolúciós neurális hálózat [12] módosított algoritmusa volt, a fuzzifikáció paramétereinek felhasználásával. Az algoritmus által végrehajtott tanulási ciklusok számát 15-től 20-ig állítottam be, hogy kiküszöböljem egy adott adatsor statisztikailag kiválasztott legjobb ciklusszámának hatását.

Tézis II/a

A neurális hálózat és a konvolúciós neurális hálózat algoritmusok hatékony gépi tanulási eszközök osztályozás vagy más problémák megoldására. Teljesítményüket minden bizonnyal befolyásolhatja a tanulási adatok minősége. Az egyik gyakori probléma az, hogy a tanulási adatok általában bináris kimeneti értékekkel rendelkeznek, akkor is, ha a tanulási minták fuzzy módon egynél több osztályba tartozhatnak. Ezeket az adatokat általában úgy címkézik, hogy minden mintának egy vagy több címkéje van. "Igaz" az osztályba tartozás a címke mögött álló osztályban, és "Hamis" tagsági érték tartozik a többi osztályhoz ugyanabban a kategóriában. Vannak olyan esetek, amikor ezek a bináris osztálytagság értékek félrevezetőnek tekinthetők, ezért ezen értékek korrekciója csökkentheti az említett zavaró hatást. A tanulási adatok módosítása gyakran hasznos a regularizációhoz. Ez történhet például torzítással, zaj hozzáadásával, adat augmentációval vagy kontradiktórius tanítással.

A javasolt fuzzifikációs technika alkalmazható más osztályozó algoritmusokra is, abban az esetben, ha képesek fuzzy tagsági értékeket megadni a kimenetükben. Más algoritmusok kutatása a fuzzifikációs technika rajtuk való alkalmazása érdekében jövőbeli kutatás lehet, a jelenlegi kutatásban neurális hálózati algoritmusokkal végeztem méréseimet.

Olyan egyszerű módszereket határoztam meg, amelyek felhasználhatók a tanulási mintákhoz megadott osztálytagsági értékek módosítására a tanulás során, hogy fuzzy kimeneti értékeket kapjunk a tanulási adatkészlet bináris értékeiből. Ezt az osztálytagsági fuzzifikációt úgy hajtjuk végre, hogy a megszerzett ismereteket a tanulási folyamat során felhasználjuk a tanulási minták pontatlan kimeneti osztálytagsági értékeinek kijavítására.

Az algoritmus három változatát mutatom be a disszertáció 7.1-es alfejezetében. Az első közülük (Algorithm 7) egyedülálló tanulóknak szól, a második verzió (Algorithm 8) több tanulóval együtt alkalmazhtó, amelynek eredménye a bizottság-gépi szavazási funkciókkal is használható, a harmadik változat (Algorithm 9) egyszerű módosítás több tanuló számára, a kezdeti fuzzifikáció paraméterek módosításával. Itt csak a második változatot (Algorithm 8) mutatom be.

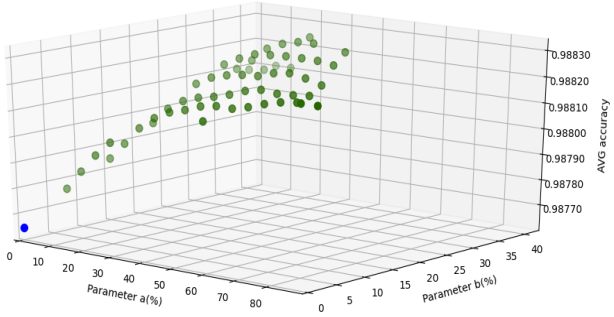
```
procedure FuzzyTrainingEnsemble(models,train_X,train_Y,FA,FB,FC):
  epoch = 0
  fuzzy_Y = train_Y
  while epoch<MAX_EPOCHS and CheckEarlyStopCondition()==False:
    for model in models:
      model.fit(train_X, fuzzy_Y, epochs=1)
      out = model.predict(train_X)
      if epoch > START_FUZZY:
        fuzzy_Y = FA*fuzzy_Y + FB*out + FC*train_Y
    epoch = epoch + 1
end procedure
```

Az algoritmust a tanulók modelljeivel, a képzési inputokkal és outputokkal, valamint a tanulási modell képzéséhez szükséges fuzzifikációs paraméterekkel kell meghívni. Az FA paraméter az impulzus súly, ami a tényleges (jelenlegi) módosított osztálytagsági értékek fontosságát jelenti (a $fuzzy\_Y$ vektor). Ez befolyásolja az eddig kialakított értékekről a kívánt értékekre való váltást, megadva a fontosságát az eddig kialakított tudásnak. A FB paraméter az aktuális friss tudás súlya (az *out* vektoré), ami a változtatás bátorságát jelenti. A FC paraméter a $train\_Y$ vektorhoz használt súly, ami az eredeti cél kimeneti adatok fontosságát jelenti. A FA, FB és a FC paraméterek összege 1 (100%). A fent bemutatott algoritmusban egyszerű feltétel, hogy hány ciklus legyen az első javítás előtt. Ez természetesen okosabb módon is változtatható lehet a jobb eredmény elérése érdekében, azonban mérésünk szempontjából fontosabb a korrekciós műveletek számának ismerete. Amikor a tanulás megkezdődik, a $fuzzy\_Y$ vektor kezdőértékei megegyeznek a $train\_Y$ vektorban megadottakkal.

A javasolt neurális hálózat tanulási adat fuzzifikáció teljesítménye

A javasolt tanulási adat osztálytagsági érték fuzzifikáció teljesítményértékelését a disszertáció 8.1-es fejezete mutatja be és írja le. Az alábbiakban bemutatjuk a kísérlet eredményét három különböző neurális hálózati algoritmus felhasználásával, a tesztadatokra vonatkozó előrejelzések pontosságára.

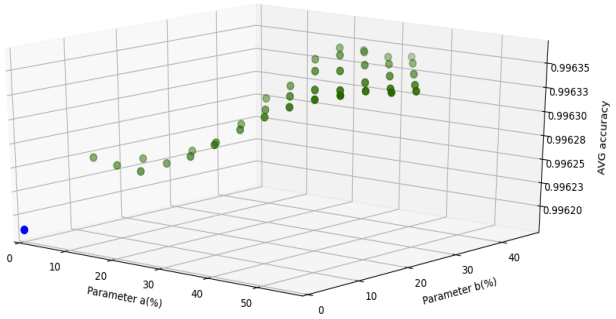
Először a korábban leírt 1. Neurális hálózat algoritmus teljesítményét vizsgáljuk, a hozzáadott fuzzifikáció alkalmazásával. Tanulók ezrei tanultak 20 ciklust 10 tanulónként együttműködve. A tanuló algoritmusok különböző paraméterekkel rendelkeztek a fuzzifikációhoz, beleértve azokat a paramétereket is, amelyek megtartják az eredeti osztálytagsági értékeket ($FA = 0, FB = 0, FC = 1$).



1. Neurális hálózat algoritmus fuzzyfikált változatának hatékonysága, átlagos pontosság a tesztadatokon, különböző paramétereket használva a tanulási adatok osztálytagsági értékeinek fuzzifikálásához.

A fenti ábra a különböző FA és FB paraméterek (% -ban ábrázolva) esetén a teszt adatokon mért átlagos egyéni pontosságot mutatja ($FC = 1 - (FA + FB)$). Az $FA = 0, FB = 0$ koordinátákkal szereplő érték az átlagos eredményt mutatja fuzzifikáció nélkül. Láthatjuk, hogy ha az FB paraméter értéke 40% körül volt, akkor pontosabbak voltak az eredmények, különösen akkor, ha az FA paraméter értéke 20% körül volt. Ez azt jelenti, hogy erősebb fuzzifikáció és közepes impulzus tényező jobb pontosságot eredményezett. A különbség jelentős volt a fuzzifikáció nélküli futás pontosságához képest.

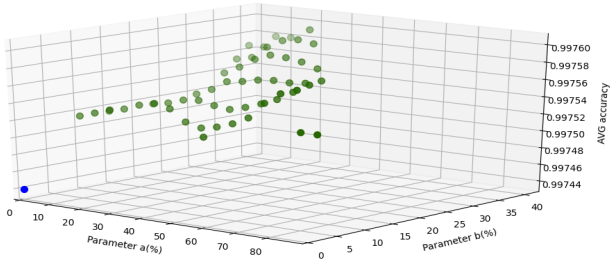
A második kísérlet a 2. Neurális hálózat algoritmussal került végrehajtásra. Tanulók ezrei tanultak különböző ideig (tanulási ciklusszámig) és különböző paraméterekkel a fuzzifikáció eredményességének vizsgálatához, beleértve azokat a paramétereket is, amelyek megtartják az eredeti osztálytagsági értékeket ($FA = 0, FB = 0, FC = 1$).



Algoritmusunk átlagos pontossági eredményei tesztadatokon, különböző paramétereket használva a tanulási adatok osztálytagságainak fuzzifikálásához a 2. Neurális hálózat algoritmus módosításával.

Az ábra bemutatja az osztályba tartozási adatok fuzzifikációs módszerének hatékonyságát, a 2. Neurális hálózat algoritmus átlagos pontosságán, a különféle FA és FB paraméterekkel ($FC = 1 - (FA + FB)$) folytatott tanulók tesztadatain. A $FA = 0, FB = 0$ koordinátákkal ellátott eredmény az átlagos eredményt mutatja fuzzifikáció nélkül. Az erősebb fuzzifikáció jobb pontosságot eredményezett, különösen akkor, ha a jelenlegi tudás (impulzus) tényezőjét meghatározó FA paraméter értéke 20% és 30% között volt. A fuzzifikációval és anélkül végzett algoritmusok egyéni pontossága között jelentős volt a különbség.

A harmadik kísérletben a tanítást a 3. Neurális hálózat algoritmus módosított változata segítségével hajtottam végre.



A fuzzifikációs algoritmus eredményessége a tesztadatokon a 3. Neurális hálózat algoritmuson, különböző paramétereket használva a tanulási adatok osztálytagsági értékeinek fuzzifikálásához.

Az ábra több ezer olyan tanuló eredményét mutatja, amelyeket különböző FA, FB és FC paraméterekkel futtattunk. Az $FA = 0, FB = 0$ koordinátákkal rendelkező pont azt az átlagos eredményt mutatja, amikor a tanulási adatok osztálytagsági értékeit nem korrigáltuk. Láthatjuk, hogy az eredmények magasabbak voltak alacsonyabb FA és FB paraméterértékekkel. Ilyen paramétereknél a FC paraméter magasabb ($FC = 1 - (FA + FB)$), így csak kisebb javítások történhettek a tanulási adatok osztálytagságának értékén.

Tézis II/b

A disszertáció 3.11. alfejezetében néhány ismert bizottság-gépi szavazási funkciót ismertettem. Jó teljesítményük motivált újak fejlesztésére. Az alábbiakban az új bizottság-gépi szavazási függvényeket mutatom be, amelyeket összehasonlítok néhány jól ismert szavazási függvénnyel. Néhányuk a lokálisan súlyozott átlag szavazási függvényekhez tartozik.

- **Fuzzy átlag szavazás a konfidenciával súlyozva (V2 a disszertációban):**

  A fuzzy átlag szavazás súlyozható a konfidenciával. Itt egy egyszerű funkciót javaslok, amely segítségével egyfajta konfidenciát nyerhetünk az osztálytagsági értékekből. Ennek a módszernek nyilvánvalóan kevesebb teljesítményre van szüksége más komplexebb módszerekhez képest. Ahol az előrejelzett osztálytagság értékek közelebb vannak a 0 vagy 1 értékekhez, ott nagyobb súlyt kapnak, mint amelyek bizonytalanok (távolabb a 0 vagy 1 értékektől). Legyen $o_i[j]$ a neurális hálózatok kimenete az egyes i tanulók által az egyes j osztályokhoz. A következő képlettel számoljuk ki a kombinált eredményt:

$$o[j] = \frac{1}{N} \sum_{i=1}^{N} ((o_i[j] - 0.5) * (2 * o_i[j] - 1)^2 + 0.5)$$

  Ebből a súlyozott átlagból kapjuk meg a nyertes osztályt:

$$l = argmax(o)$$

- **Fuzzy átlag szavazás, 1-az átlagtól való különbséggel súlyozva (V3 a disszertációban):**

  A tanulók kimenetének ismeretében alkalmazhatunk egy másik súlyozott átlag módszert a fuzzy átlag jobb teljesítményére alapozva az egyéni tanulókhoz képest. A fuzzy átlag kiszámításával kezdve, az egyéni előrejelzéseket megszorozzuk egy súly értékkel, amely az egyéni előrejelzés érték és a közös eredmény közti különbség 1-ből értéke. Számítsuk az $o[j]$ értéket a fuzzy szavazáshoz definiált módon. Ezután kiszámítjuk az új variánst a következők szerint:

$$o'[j] = \frac{1}{N} \sum_{i=1}^{N} ((o_i[j] - 0.5) * (1 - |o_i[j] - o[j]|) + 0.5)$$

  A nyertes osztályt ezután a szokásos módon kapjuk:

$$l = argmax(o')$$

- **Fuzzy átlagos szavazás tanulási mintán való tévesztések reciprokával súlyozva (V4 a disszertációban):**

  Legyen f_i az i tanuló általi tévesztések száma a tanulási mintákon ha ez nem 0, egyébként egy értéket választunk 0 és 1 között, pl. 0.5.

  Az f_i reciprokával súlyozzuk az i tanuló eredményét:

$$o[j] = \left(\frac{1}{N}\sum_{i=1}^{N}\frac{o_i[j]}{f_i}\right)\left(\frac{1}{N}\sum_{i=1}^{N}f_i\right)$$

  Ebből a súlyozott átlag eredményből megkapjuk a nyertes osztályt:

$$l = argmax(o)$$

Meta-szavazás változatai

Fuzzy átlag, szorzat, illetve legtöbb vagy többségi szavazás segítségével összegezzük a szavazási funkciók eredményét.

Elemzés céljából három meta-szavazó változatot határozunk meg:

- V8: Legtöbb szavazat a V1, V2, V3, V4, V5, V6, V7 funkciók eredményéből
- V9: Legtöbb szavazat a V1, V2, V3, V4, V7 funkciók eredményéből
- V10 Fuzzy átlag a V1, V2, V3, V4, V7 funkciók eredményéből

A fenti három meta szavazási függvényhez először kiszámoljuk a szükséges szavazási függvények eredményeit, majd ezeket egyesítjük, ahogy azt az egyes tanulók eredményei alapján számított szavazási függvényekre fentebb leírtuk.

A javasolt szavazási funkciók teljesítményértékelése

Szavazó függvény	MIN	AVG	MAX
min(accuracy)	0.981700	0.986263	0.990600
avg(accuracy)	0.983600	0.988148	0.990900
max(accuracy)	0.983800	0.989761	0.991900
V1–fuzzy average	0.987600	0.992755	0.994800
V2–weighted by confidence	0.987600	0.992769	0.994700
V3–weighted by diff from V1	0.987600	0.992752	0.994800
V4–weighted by 1/training fails	0.987600	0.992761	0.994700
V5–plurality voting	0.981100	0.992449	0.994800
V6–borda voting	0.982000	0.992537	0.994600
V7–geometric mean voting	0.987500	**0.992790**	0.994800
V8–meta–plurality (V1-V7)	0.987600	0.992764	0.994800
V9–meta–plurality (V1-V4,V7)	0.987600	0.992765	0.994800
V10–meta–fuzzy avg (V1-V4,V7)	0.987700	0.992776	0.994800

Minimális, maximális és átlagos pontosság az együttes tanulási szavazási kísérletből az 1. Neurális hálózat algoritmus módosított változatával

A fenti táblázat mutatja az eredményeket, ahol minden körben $5 - 20$ szavazó adta le a voksát, amelyeket aztán a fent definiált szavazó funkciókkal egyesítettünk. A legjobb egyéni eredmény 81 téves besorolás volt a 10000 tesztmintán, a legrosszabb az egyéni 183 sikertelen minta volt, és átlagosan 118.52 sikertelen predikció volt egyéni tanulónként. A jól ismert fuzzy szavazás átlagosan $72,45$ -os tévesztést eredményezett. A szavazási funkciók között nem volt nagy különbség, a legjobb átlageredményt (átlagosan $72,1$ nem sikerült) a geometriai átlag (V7) szavazási függvény érte el.

szavazási függvény	**MIN**	**AVG**	**MAX**
min(accuracy)	0.992700	0.995422	0.996900
avg(accuracy)	0.995188	0.996306	0.997260
max(accuracy)	0.995600	0.997043	0.998000
V1–fuzzy average	0.996000	0.997351	0.998400
V2–weighted by confidence	0.995900	0.997360	0.998300
V3–weighted by diff from V1	0.996000	0.997346	0.998300
V4–weighted by 1/failures	0.995500	0.997321	0.998300
V5–plurality voting	0.995500	0.997280	0.998400
V6–borda voting	0.995600	0.997296	0.998400
V7–geometric mean voting	0.996000	**0.997367**	0.998300
V8–meta–plurality (V1-V7)	0.995900	0.997353	0.998400
V9–meta–plurality (V1-V4,V7)	0.996000	0.997356	0.998400
V10–meta–fuzzy avg (V1-V4,V7)	0.996100	0.997350	0.998300

Minimális, maximális és átlagos pontosság együttes tanulási szavazási kísérlet eredményeiből a 2. Neurális hálózat algoritmus módosított verziójával

A fenti táblázat a tesztelt szavazási funkciók összesített eredményét mutatja az egyes tanulók sikertelen mintáinak minimális, átlagos és maximális számával. A legjobb egyéni eredmény 20 tévesztés volt a 10000 tesztmintán, a legrosszabb a 73 sikertelen minta besorolás volt, és átlagosan csak $36,94$ kudarc volt 10000 mintából egyéni tanulónként. A jól ismert fuzzy átlag szavazás átlagosan $26,49$ hibázást ért el. A szavazási funkciók között nem volt nagy különbség, a legjobb eredményt a szorzat szavazás (illetve a geometriai átlag - V7) adta a teszt adatokon.

szavazási függvény	MIN	AVG	MAX
min(accuracy)	0.996200	0.996878	0.997700
avg(accuracy)	0.996840	0.997418	0.997886
max(accuracy)	0.997100	0.997914	0.998500
V1–fuzzy average	0.997300	0.998126	0.998700
V2–weighted by confidence	0.997300	0.998122	0.998700
V3–weighted by diff from V1	0.997200	0.998128	0.998700
V4–weighted by 1/failures	0.997300	0.998126	0.998700
V5–plurality voting	0.997000	0.998044	0.998700
V6–borda voting	0.997000	0.998044	0.998700
V7–geometric mean voting	0.997100	0.998126	0.998700
V8–meta–plurality (V1-V7)	0.997200	0.998125	0.998700
V9–meta–plurality (V1-V4,V7)	0.997200	0.998128	0.998700
V10–meta–fuzzy avg (V1-V4,V7)	0.997200	**0.998129**	0.998700

Minimális, maximális és átlagos pontosság az együttes tanulás szavazási kísérlet alapján a 3. Neurális hálózati algoritmus módosított változatával.

A táblázat azokat az eredményeket mutatja, ahol minden körben $5-20$ szavazó szavazott a fentebb meghatározott szavazási funkciók segítségével. A legjobb egyéni eredmény 15 sikertelen besorolás, ami 10000 tesztmintán keletkezett, a legrosszabb egyéni eredmény 38 sikertelen minta volt, és átlagosan csak $25,82$ tévedést vétettek egyenként a tanulók. A jól ismert fuzzy átlag szavazás átlagosan $18,74$ hibával teljesített. A szavazási funkciók között nem volt nagy különbség, a legjobb eredményt ($18,71$ sikertelen) az egyik meta fuzzy választói funkciónk (V10) adta.

3 Publications

Journal publications related to the dissertation

[1] B. Nagy, R. Basbous, and T. Tajti. "Lazy evaluations in Łukasiewicz type fuzzy logic". In: *Fuzzy Sets and Systems* 376 (2019), pp. 127–151.

[2] T. Tajti. "Fuzzification of training data class membership binary values for neural network algorithms". In: *Annales Mathematicae et Informaticae* 52 (2020), pages to be approved.

[3] T. Tajti. "New voting functions for neural network algorithms". In: *Annales Mathematicae et Informaticae* 52 (2020), pages to be approved.

Foreign language conference proceedings related to the dissertation

[4] R. Basbous, B. Nagy, and T. Tajti. "Short Circuit Evaluations in Gödel Type Logic". In: *Proc. of FANCCO 2015: 5th International Conference on Fuzzy and Neuro Computing, Advances in Intelligent Systems and Computing*. Vol. 415. 2015, pp. 119–138.

[5] R. Basbous, T. Tajti, and B. Nagy. "Fast Evaluations in Product Logic: Various Pruning Techniques". In: *FUZZ-IEEE 2016 - the 2016 IEEE International Conference on Fuzzy Systems*. Vancouver, Canada: IEEE, 2016, pp. 140–147.

[6] C. Biró, G. Kusper, and T. Tajti. "How to generate weakly nondecisive SAT instances". en. In: *Intelligent Systems and Informatics (SISY), 2013 IEEE 11th International Symposium on : IEEE 11th International Symposium on Intelligent Systems and Informatics : proceedings*. Budapest, Magyarország: IEEE Hungary, 2013, pp. 265–269.

Other publications

[7] T. Tajti. "Fuzzification of neural network pattern outputs for classification problems [abstract]". ICAI 2020 : 11th International Conference on Applied Informatics, Eger, Hungary. 2020.

[8] Z. Gál, T. Tajti, and G. Terdik. "Surprise event detection of the supercomputer execution queues". en. In: *Annales Mathematicae et Informaticae* 44 (2015), pp. 87–97.

[9] Z. Gál et al. "Performance evaluation of massively parallel communication sessions". en. In: *The Sixth International Conference on Parallel, Distributed, GPU and Cloud Computing for Engineering*. Civil-Comp Press, 2019, pp. 1–19.

[10] T. Tajti and B. Nagy. "Motion sensor data correction using multiple sensors and multiple measurements". en. In: *SAMI 2016 : IEEE 14th International Symposium on Applied Machine Intelligence and Informatics, New York (NY), Amerikai Egyesült Államok*. IEEE, 2016, pp. 287–291.

[11] T. Tajti et al. "Indoor localization using NFC and mobile sensor data corrected using neural net". en. In: *ICAI 2014: Proceedings of the 9th International Conference on Applied Informatics*. Vol. 1-2. Eszterházy Károly Tanárképzõ Fõiskola, 2014, pp. 163–169.

[12] T. Gregus ; G. Geda ; P. Magyar ; T. Tajti ; A. Perjési. “Business Oriented Creature Identification”. en. In: *ICAI 2014 The 9th International Conference on Applied Informatics to be held in Eger*. Eger, Magyarország: Eszterházy Károly College, 2014.

[13] C. Biró et al. *Look-ahead alapú SAT solver-ek párhuzamosíthatóságának vizsgálata*. hu. Budapest, Magyarország, 2013.

[14] G. Kusper, C. Biró, and T. Tajti. *WnDGen: Weakly Nondecisive SAT Problem Generator*. en. Tudományos célú szoftver, Version: 1.0, 29.05.2012, Current version: 2.0, 07.01.2013, Megjelenés: Magyarország. 2013.

[15] G. Kusper, C. Biró, and T. Tajti. *SATCounter: Count Clear Clauses SAT Solver*. en. Vol. 1. Magyarország, 2013.

[16] Z. Gál and T. Tajti. “Complex event processing in supercomputer environment: sensor and neural network based analysis”. en. In: *IEEE 4th International Conference on Cognitive Infocommunications: CogInfoCom 2013, New York (NY), Amerikai Egyesült Államok*. IEEE, 2013, pp. 735–740.

[17] J. Kajdi ; K. Nagy ; Ő. Legeza ; J. Kövesi ; T. Tajti ; A. Vigvári. *Az önkormányzati adósságregiszter kialakításának megalapozása*. hu. Magyar Közigazgatási Intézet, 2001, pp. 197–283.

4 References

[1] K. Gödel. "Zum intuitionistische Aussagenkalkül". In: *Mathematish-Naturwissenschaftliche Klasse* 69 (1932). reprinted in Kurt Gödel, Collected Works, Oxford University Press, 1985, pp. 65–66.

[2] P. Hájek, L. Godo, and F. Esteva. "A complete many-valued logic With product-conjunction". en. In: *Archive for Mathematical Logic* 35 (1996), pp. 191–208.

[3] J. Lukasiewicz and A. Tarski. "Investigations in the Sentential Calculus". en. In: (1930). translation in J. Lukasiewicz, Selected Works, Reidel, Dordrecht (1970).

[4] S. Gottwald. *Many-Valued Logic*. en. Edward N. Zalta (ed.) The Stanford Encyclopedia of Philosophy, URL=http://plato.stanford.edu/entries/logic-manyvalued. 2015.

[5] R. Basbous, T. Tajti, and B. Nagy. "Fast Evaluations in Product Logic: Various Pruning Techniques". In: *FUZZ-IEEE 2016 - the 2016 IEEE International Conference on Fuzzy Systems*. Vancouver, Canada: IEEE, 2016, pp. 140–147.

[6] B. Nagy, R. Basbous, and T. Tajti. "Lazy evaluations in Łukasiewicz type fuzzy logic". In: *Fuzzy Sets and Systems* 376 (2019), pp. 127–151.

[7] J. Siltaneva. "A Comparison of Random Binary Tree Generators". In: *The Computer Journal* 45.6 (2002), pp. 653–660.

[8] T. Tajti. "Fuzzification of training data class membership binary values for neural network algorithms". In: *Annales Mathematicae et Informaticae* 52 (2020), pages to be approved.

[9] T. Tajti. "New voting functions for neural network algorithms". In: *Annales Mathematicae et Informaticae* 52 (2020), pages to be approved.

[10] Yann LeCun, Corinna Cortes, and Christopher JC Burges. "The MNIST database of handwritten digits, 1998". In: *URL http://yann.lecun.com/exdb/mnist* 10.34 (1998), p. 14.

[11] Chris Deotte. *25 Million Images! [0.99757] MNIST*. https://www.kaggle.com/cdeotte/25-million-images-0-99757-mnist. 2020 (last accessed October 30, 2020).

[12] Matuzas77. *MNIST classifier with average 0.17% error*. https://github.com/Matuzas77/MNIST-0.17/blob/master/MNIST_final_solution.ipynb. 2020 (last accessed October 30, 2020).

[13] Leijun Li et al. "Exploration of classification confidence in ensemble learning". In: *Pattern recognition* 47.9 (2014), pp. 3120–3131.

[14] R. Basbous, B. Nagy, and T. Tajti. "Short Circuit Evaluations in Gödel Type Logic". In: *Proc. of FANCCO 2015: 5th International Conference on Fuzzy and Neuro Computing, Advances in Intelligent Systems and Computing*. Vol. 415. 2015, pp. 119–138.

UNIVERSITY of DEBRECEN

UNIVERSITY AND NATIONAL LIBRARY
UNIVERSITY OF DEBRECEN
H-4002 Egyetem tér 1, Debrecen
Phone: +3652/410-443, email: publikaciok@lib.unideb.hu

Registry number: DEENK/316/2020.PL
Subject: PhD Publication List

Candidate: Tibor Gábor Tajti
Doctoral School: Doctoral School of Informatics
MTMT ID: 10043970

List of publications related to the dissertation

Foreign language scientific articles in Hungarian journals (2)

1. **Tajti, T. G.**: Fuzzification of training data class membership binary values for neural network algorithms.
   Ann. Math. Inform. [Epub], 1-12, 2020. ISSN: 1787-5021.
   DOI: http://dx.doi.org/10.33039/ami.2020.10.001

2. **Tajti, T. G.**: New voting functions for neural network algorithms.
   Ann. Math. Inform. [Epub], 1-14, 2020. ISSN: 1787-5021.
   DOI: http://dx.doi.org/10.33039/ami.2020.10.003

Foreign language scientific articles in international journals (1)

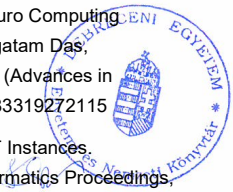
3. Nagy, B., Basbous, R., **Tajti, T. G.**: Lazy evaluations in Łukasiewicz type fuzzy logic.
   Fuzzy Sets Syst. 376, 127-151, 2019. ISSN: 0165-0114.
   DOI: http://dx.doi.org/10.1016/j.fss.2018.11.014
   IF: 3.305

Foreign language conference proceedings (3)

4. Basbous, R., **Tajti, T. G.**, Nagy, B.: Fast evaluations in product logic various pruning techniques.
   In: 2016 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), IEEE, New York, 140-147, 2016. ISBN: 9781509006250

5. Basbous, R., Nagy, B., **Tajti, T. G.**: Short Circuit Evaluations in Gödel Type Logic.
   In: Proceedings of the Fifth International Conference on Fuzzy and Neuro Computing (FANCCO - 2015). Ed.: Vadlamani Ravi, Bijaya Ketan Panigrahi, Swagatam Das, Ponnuthurai Nagaratnam Suganthan, Springer, Cham, 119-138, 2015, (Advances in Intelligent Systems and Computing, ISSN 2194-5357 ; 415.) ISBN: 9783319272115

6. Biró, C., Kusper, G., **Tajti, T. G.**: How to Generate Weakly Nondecisive SAT Instances.
   In: IEEE 11th International Symposium on Intelligent Systems and Informatics Proceedings, IEEE, Piscataway, 265-269, 2013. ISBN: 9781479903054



UNIVERSITY of DEBRECEN

UNIVERSITY AND NATIONAL LIBRARY
UNIVERSITY OF DEBRECEN
H-4002 Egyetem tér 1, Debrecen
Phone: +3652/410-443, email: publikaciok@lib.unideb.hu

List of other publications

Foreign language scientific articles in Hungarian journals (1)

7. Gál, Z., **Tajti, T. G.**, Terdik, G.: Surprise event detection of the supercomputer execution queues. *Ann. Math. et Inf. 44*, 87-97, 2015. ISSN: 1787-5021.

Hungarian conference proceedings (1)

8. Terdik, G., Gál, Z., **Tajti, T. G.**: A Debreceni Universitas lokális hálózatainak adatvédelmi és biztonsági bővítése.
In: Networkshop '95 : Országos konferencia : Gödöllő 1995. április 19-21.. Szerk.: Bajza János, Tóth Beatrix, Nemzeti Információs Infrastruktúra Fejlesztési Program Koordinációs Iroda, Budapest, 124-130, 1995. ISBN: 9630484927

Foreign language conference proceedings (4)

9. Gál, Z., Varga, I., **Tajti, T. G.**, Kocsis, G., Langmajer, Z., Kósa, M., Pánovics, J.: Performance evaluation of massively parallel communication sessions.
In: Proceedings of the Sixth International Conference on Parallel, Distributed, GPU and Cloud Computing for Engineering. Ed.: P. Iványi, B. H. V. Topping, Civil-Comp Press, Pécs, 1-19, 2019. ISBN: 9781905088676

10. **Tajti, T. G.**, Nagy, B.: Motion sensor data correction using multiple sensors and multiple measurements.
In: SAMI 2016 IEEE 14th International Symposium on Applied Machine Intelligence and Informatics : Proceedings : January 21-23, 2016, Herl'any, Slovakia, IEEE Computer Society, Piscataway, 287-291, 2017. ISBN: 9781467387408

11. **Tajti, T. G.**, Geda, G., Balla, T., Vad, G.: Indoor localization using NFC and mobile sensor data corrected using neural net.
In: Proceedings of the 9th International Conference on Applied Informatics. Ed.: by Kovács Emőd, Kusper Gábor, Kunkli Roland, Tómács Tibor, Eszterházy Károly Főiskola, Eger, 163-169, 2014. ISBN: 9786155297182

UNIVERSITY of DEBRECEN

UNIVERSITY AND NATIONAL LIBRARY
UNIVERSITY OF DEBRECEN
H-4002 Egyetem tér 1, Debrecen
Phone: +3652/410-443, email: publikaciok@lib.unideb.hu

12. Gál, Z., **Tajti, T. G.**: Complex event processing in supercomputer environment: sensor and neural network based analysis.
In: 4th IEEE International Conference on Cognitive Infocommunications CogInfoCom 2013 : Proceedings, December 2-5, 2013 Budapest, Hungary. Ed.: Péter Baranyi, IEEE, Danvers, 435-440, 2013. ISBN: 9781479915439

Total IF of journals (all publications): 3,305
Total IF of journals (publications related to the dissertation): 3,305

The Candidate's publication data submitted to the iDEa Tudóstér have been validated by DEENK on the basis of the Journal Citation Report (Impact Factor) database.

22 October, 2020

DEBRECENI EGYETEM
EGYETEMI ÉS NEMZETI KÖNYVTÁR
H-4002 Debrecen, Egyetem tér 1, Pf.: 400
Tel.: 52/410-443, e-mail: publikaciok@lib.unideb.hu

Nyilvántartási szám: DEENK/316/2020.PL
Tárgy: PhD Publikációs Lista

Jelölt: Tajti Tibor Gábor
Doktori Iskola: Informatikai Tudományok Doktori Iskola
MTMT azonosító: 10043970

A PhD értekezés alapjául szolgáló közlemények

Idegen nyelvű tudományos közlemények hazai folyóiratban (2)

1. **Tajti, T. G.**: Fuzzification of training data class membership binary values for neural network algorithms.
   Ann. Math. Inform. [Epub], 1-12, 2020. ISSN: 1787-5021.
   DOI: http://dx.doi.org/10.33039/ami.2020.10.001

2. **Tajti, T. G.**: New voting functions for neural network algorithms.
   Ann. Math. Inform. [Epub], 1-14, 2020. ISSN: 1787-5021.
   DOI: http://dx.doi.org/10.33039/ami.2020.10.003

Idegen nyelvű tudományos közlemények külföldi folyóiratban (1)

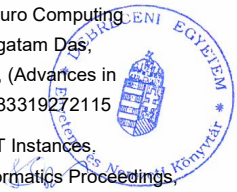
3. Nagy, B., Basbous, R., **Tajti, T. G.**: Lazy evaluations in Łukasiewicz type fuzzy logic.
   Fuzzy Sets Syst. 376, 127-151, 2019. ISSN: 0165-0114.
   DOI: http://dx.doi.org/10.1016/j.fss.2018.11.014
   IF: 3.305

Idegen nyelvű konferencia közlemények (3)

4. Basbous, R., **Tajti, T. G.**, Nagy, B.: Fast evaluations in product logic various pruning techniques.
   In: 2016 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), IEEE, New York, 140-147, 2016. ISBN: 9781509006250

5. Basbous, R., Nagy, B., **Tajti, T. G.**: Short Circuit Evaluations in Gödel Type Logic.
   In: Proceedings of the Fifth International Conference on Fuzzy and Neuro Computing (FANCCO - 2015). Ed.: Vadlamani Ravi, Bijaya Ketan Panigrahi, Swagatam Das, Ponnuthurai Nagaratnam Suganthan, Springer, Cham, 119-138, 2015, (Advances in Intelligent Systems and Computing, ISSN 2194-5357 ; 415.) ISBN: 9783319272115

6. Biró, C., Kusper, G., **Tajti, T. G.**: How to Generate Weakly Nondecisive SAT Instances.
   In: IEEE 11th International Symposium on Intelligent Systems and Informatics Proceedings, IEEE, Piscataway, 265-269, 2013. ISBN: 9781479903054



DEBRECENI EGYETEM
EGYETEMI ÉS NEMZETI KÖNYVTÁR
H-4002 Debrecen, Egyetem tér 1, Pf.: 400
Tel.: 52/410-443, e-mail: publikaciok@lib.unideb.hu

További közlemények

Idegen nyelvű tudományos közlemények hazai folyóiratban (1)

7. Gál, Z., **Tajti, T. G.**, Terdik, G.: Surprise event detection of the supercomputer execution queues. *Ann. Math. et Inf. 44*, 87-97, 2015. ISSN: 1787-5021.

Magyar nyelvű konferencia közlemények (1)

8. Terdik, G., Gál, Z., **Tajti, T. G.**: A Debreceni Universitas lokális hálózatainak adatvédelmi és biztonsági bővítése.
In: Networkshop '95 : Országos konferencia : Gödöllő 1995. április 19-21.. Szerk.: Bajza János, Tóth Beatrix, Nemzeti Információs Infrastruktúra Fejlesztési Program Koordinációs Iroda, Budapest, 124-130, 1995. ISBN: 9630484927

Idegen nyelvű konferencia közlemények (4)

9. Gál, Z., Varga, I., **Tajti, T. G.**, Kocsis, G., Langmajer, Z., Kósa, M., Pánovics, J.: Performance evaluation of massively parallel communication sessions.
In: Proceedings of the Sixth International Conference on Parallel, Distributed, GPU and Cloud Computing for Engineering. Ed.: P. Iványi, B. H. V. Topping, Civil-Comp Press, Pécs, 1-19, 2019. ISBN: 9781905088676

10. **Tajti, T. G.**, Nagy, B.: Motion sensor data correction using multiple sensors and multiple measurements.
In: SAMI 2016 IEEE 14th International Symposium on Applied Machine Intelligence and Informatics : Proceedings : January 21-23, 2016, Herl'any, Slovakia, IEEE Computer Society, Piscataway, 287-291, 2017. ISBN: 9781467387408

11. **Tajti, T. G.**, Geda, G., Balla, T., Vad, G.: Indoor localization using NFC and mobile sensor data corrected using neural net.
In: Proceedings of the 9th International Conference on Applied Informatics. Ed.: by Kovács Emőd, Kusper Gábor, Kunkli Roland, Tómács Tibor, Eszterházy Károly Főiskola, Eger, 163-169, 2014. ISBN: 9786155297182

DEBRECENI EGYETEM
EGYETEMI ÉS NEMZETI KÖNYVTÁR
H-4002 Debrecen, Egyetem tér 1, Pf.: 400
Tel.: 52/410-443, e-mail: publikaciok@lib.unideb.hu

12. Gál, Z., **Tajti, T. G.**: Complex event processing in supercomputer environment: sensor and neural network based analysis.
In: 4th IEEE International Conference on Cognitive Infocommunications CogInfoCom 2013 : Proceedings, December 2-5, 2013 Budapest, Hungary. Ed.: Péter Baranyi, IEEE, Danvers, 435-440, 2013. ISBN: 9781479915439

A közlő folyóiratok összesített impakt faktora: 3,305
A közlő folyóiratok összesített impakt faktora (az értekezés alapjául szolgáló közleményekre): 3,305

A DEENK a Jelölt által az iDEa Tudóstérbe feltöltött adatok bibliográfiai és tudománymetriai ellenőrzését a tudományos adatbázisok és a Journal Citation Reports Impact Factor lista alapján elvégezte.

Debrecen, 2020.10.22.

DEBRECENI EGYETEM * Egyetemi és Nemzeti Könyvtár *