DE GRUYTER OPEN

JAISCR, 2017, Vol. 7, No. 4, pp. 265 – 286
10.1515/jaiscr-2017-0019

CHARACTERIZATION OF SYMBOLIC RULES EMBEDDED IN DEEP DIMLP NETWORKS: A CHALLENGE TO TRANSPARENCY OF DEEP LEARNING

Guido Bologna[1], Yoichi Hayashi[2]

[1]*Department of Computer Science, University of Applied Science of Western Switzerland, Rue de la Prairie 4, Geneva 1202, Switzerland*

[2]*Department of Computer Science, Meiji University, Tama-ku, Kawasaki, Kanagawa 214-8571, Japan*

submitted: 14th February 2017; accepted: 20th March 2017

Abstract

Rule extraction from neural networks is a fervent research topic. In the last 20 years many authors presented a number of techniques showing how to extract symbolic rules from Multi Layer Perceptrons (MLPs). Nevertheless, very few were related to ensembles of neural networks and even less for networks trained by deep learning. On several datasets we performed rule extraction from ensembles of Discretized Interpretable Multi Layer Perceptrons (DIMLP), and DIMLPs trained by deep learning. The results obtained on the Thyroid dataset and the Wisconsin Breast Cancer dataset show that the predictive accuracy of the extracted rules compare very favorably with respect to state of the art results. Finally, in the last classification problem on digit recognition, generated rules from the MNIST dataset can be viewed as discriminatory features in particular digit areas. Qualitatively, with respect to rule complexity in terms of number of generated rules and number of antecedents per rule, deep DIMLPs and DIMLPs trained by arcing give similar results on a binary classification problem involving digits *5* and *8*. On the whole MNIST problem we showed that it is possible to determine the feature detectors created by neural networks and also that the complexity of the extracted rulesets can be well balanced between accuracy and interpretability.

Keywords: ensembles, Deep Learning, rule extraction, feature detectors

1 Introduction

A main drawback of artificial neural networks is that they have no explicit declarative knowledge representation. They have considerable difficulty in generating the required explanation structures. The absence of an explanation capability in neural networks limits the achievement of the full potential of such systems, whereas the detailed characterization of classification strategies would contribute to their acceptance. Expert systems benefit from an explicit declarative representation of knowledge about the problem domain. Hence, a natural way to elucidate the knowledge embedded within neural networks is to extract symbolic rules, even if this problem is known to be NP-hard [1]. Very recently, Hailesilassie explained in a review [2] that most of rule extraction algorithms are applied to MLPs with a hidden layer. Unexpectedly, very little work was achieved with respect to deep neural networks. Filling this gap may contribute to the real world usability of deep MLPs.

The *Discretized Interpretable Multi Layer Perceptron* (DIMLP) was introduced in [3] [4]. This feed forward neural network model represents a constrained Multi Layer Perceptron (MLP) architecture that can be used to learn any classification problem. The main advantage of DIMLPs is that symbolic rules are extracted in polynomial time and this can also be performed for DIMLP ensembles [5]. In this work we first apply DIMLP ensembles to the Thyroid dataset and the Wisconsin Breast Cancer diagnosis problem, two well-known benchmark datasets. The obtained results show that the predictive accuracy of the extracted rules compare very favorably with respect to state of the art results. Then, deep DIMLP networks trained with stacked autoencoders are applied to MNIST; a well-known datasets in the computer vision domain. Our main contribution is the characterization of the feature detectors extracted by applying rule extraction to deep DIMLPs.

In fact, feature detectors correspond to rule antecedents of generated rules. Moreover, from each feature detector we calculate a centroid of the covered samples, which allows us to visualize average patterns detected by feature detectors. Rules extracted from the MNIST dataset play the role of discriminatory features in particular areas of digits. Another question we tackle is whether the knowledge embedded within DIMLPs trained by deep learning is different from that found in DIMLP ensembles, with respect to feature complexity. In the following Sections we first explain the difficulty of rule extraction and give a summary of representative techniques. In Section 3 we present the DIMLP model as a single neural network, then as an ensemble of DIMLPs and lastly as a single deep network made of stacked autoencoders. Section 4 describes the experiments on three classification problems including a discussion, followed by a conclusion.

2 The difficulty of rule extraction from neural networks

A symbolic rule is defined as: "if tests on antecedents are true then conclusion"; where "tests on antecedents" are in the form $x_i \leq v_i$ or $x_i \geq v_i$; with x_i as an input variable and v_i as a real number. Given a neural network model to classify data in hypothesis h, rule extraction aims to obtain a description $\hat{h}$ which approximates h as closely as possible ($\hat{h} \approx h$). Ideally, $\hat{h}$ would be represented by a small number of rules and antecedents.

The complexity of rulesets is often defined as a measure of two variables: number of rules; and number of antecedents per rule. Low complexity rulesets are preferred than those with high complexity, which involves a large number of rules with a large number of antecedents. However, Freitas pointed out that the comprehensibility of rules is not necessarily related to a small number of rules [6]. Usually, with a large number of extracted rules, comprehensibility is assumed to be low. However, Freitas has identified several drawbacks with respect to model size as the only measure of comprehensibility [6]. He proposed a new measure denoted as *prediction-explanation size*, which strongly depends on the average number of antecedents per rule.

Rule extraction from neural networks can also be viewed as an optimization problem, since a clear trade-off between accuracy and comprehensibility is present. Typically, a greater number of rules may provide better accuracy while diminishing comprehensibility. Moreover, another trade-off is related to the number of rules and the number of uncovered samples. Specifically, higher numbers of rules decreases the number of uncovered samples while it degrades the level of comprehensibility. For instance, in a medical classification problem with classes *sick* and *healthy* a physician could be interested to solely explain why a patient is sick. Hence, all rules belonging to class healthy would be ignored. Without rules of class healthy a ruleset would be smaller (thus, with higher comprehensibility and reduced covering), but it will be impossible to explain why a patient is healthy.

Another important distinction of the extracted rules is whether they are ordered or not. Ordered rules correspond to:

if tests on antecedents are true then ...,

else if tests on antecedents are true then ...,

...,

else ...

Because of the presence of "else", a rule implicitly negates the antecedents of the previous rule.

Thus, with the exception of the first rule all the other rules depend also on a number of hidden antecedents corresponding to the negation of the previous antecedents. In unordered rules "else if" is never present. Thus, contrary to ordered rulesets, a sample can activate more than a rule. Generally, unordered rulesets present more rules and antecedents than ordered ones, since all rule antecedents are explicitly characterized. Hence, long ordered rulesets are difficult to understand since they potentially include many implicit antecedents. Contrary to ordered rules, each rule of an unordered ruleset represents a single piece of knowledge that can be examined in isolation, since all antecedents are explicitly given. With a great number of unordered rules, one would try to accurately understand the meaning of each rule with respect to the data domain. Thus, after a careful inspection of all pieces of knowledge, getting the global picture could be long. However, one could be interested to only some parts of the whole knowledge, for instance those rules with the highest number of covered samples.

2.1 Commonly used approach: feature detectors

Feature detectors represent characteristic patterns contributing to the classification of data samples. As an example, oriented edges play a significant role for the classification of objects. Typical analysis approaches of neural networks trained by deep learning associate feature detectors to hidden neurons. Specifically, patterns involving strong activation of a hidden neuron are defined from the incoming weight signs [7]. Usually, only a feature detector is associated to a hidden neuron.

Let us suppose that an MLP has a hidden layer to learn a classification problem with 2D input pictures, as shown in Figure 1.

Further, we suppose that:

- input values are binary with values 0 and 1,
- weight values are binary with values $-\omega$ and ω (ω being a constant),
- the activation function of hidden neurons and output neurons is a sigmoid function given as

$$\sigma(x) = \frac{1}{1+\exp(-x)}. \qquad (1)$$

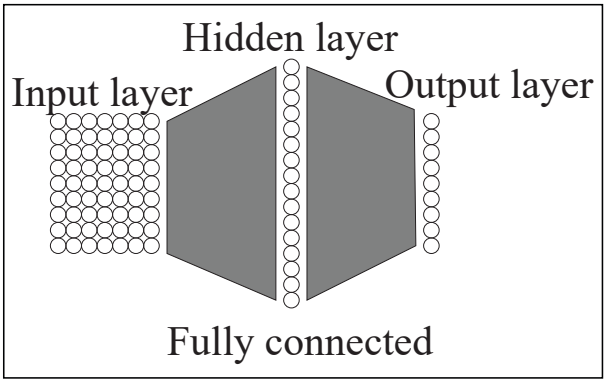

Figure 1. An MLP for a classification task with an input layer representing pixels of a picture and an output layer coding several classes.

In the network shown in Figure 1 we have full connectivity between the input layer and the hidden layer, as well as between the hidden layer and the output layer, with each connection being associated with a weight value. The incoming weight values related to a hidden neuron define a feature detector composed of a sub-vector of positive weight values and a sub-vector of negative weight values. Specifically, weight values equal to ω represent input neurons which activate that hidden neuron the most, while weight values equal to $-\omega$ inhibit that neuron the strongest. If an input picture matches a subset of a feature detector (for instance, when a subset of input neurons with values equal to one correspond to a subset of weight values equal to ω), then the corresponding hidden neuron will be activated but not at maximal activation. An input picture that does not match the feature will not activate the corresponding hidden neuron.

The simple method of defining feature detectors from strong magnitude weight values can not be carried out for realistic neural networks. Usually, input values and weights are continuous; thus for a given hidden neuron activation there will be many possible different input patterns that makes the feature extraction process much more complex. Therefore, we only have an intuitive idea of the overall classification strategy. In addition, network responses depends also on the weight matrix between the hidden layer and the output layer. With many hidden layers as in deep networks feature characterization becomes intractable.

2.2 State of the art

Since the earliest work of Gallant [8], many techniques have been introduced. Andrews et al. presented a taxonomy to characterize rule extraction techniques [9]. Essentially, rule extraction algorithms belong to three categories: *decompositional*; *pedagogical*; and *eclectic*. In decompositional techniques rules are extracted at the level of hidden and output neurons by analyzing weight values. Here a basic requirement is that the computed output from each hidden and output unit must be mapped into a binary outcome which corresponds to the notion of a rule consequent. Note that only shallow networks are analyzed by algorithms belonging to this category.

The basic idea of the pedagogical approach is to view rule extraction as a learning task where the target concept is the function computed by the network and the input attributes are simply the network's input neurons. Finally, the eclectic approach takes into account elements of both decompositional and pedagogical techniques. More recently, Diederich et al. published a book on techniques to extract symbolic rules from Support Vector Machines (SVMs) [10].

Hansen and Salamon demonstrated that combining several neural networks in an ensemble can improve the predictive accuracy with respect to a single model [11]. Nevertheless, only a few authors started to extract rules from neural network ensembles. Bologna proposed the Discretized Interpretable Multi Layer Perceptron (DIMLP) to produce unordered symbolic rules from both single networks and ensembles [4]. Zhou et al. introduced the REFNE algorithm (Rule Extraction from Neural Network Ensemble) [12], which utilizes the trained ensembles to generate instances and then extracts symbolic rules from those instances. Attributes are discretized during rule extraction and it also uses particular fidelity evaluation mechanisms. Moreover, rules have been limited to only three antecedents. More recently Hara and Hayashi proposed the two-MLP ensembles by using the *Recursive-Rule eXtraction* (Re-RX) algorithm [13] for data with mixed attributes [14]. Re-RX utilizes C4.5 decision trees and back-propagation to train the MLPs recursively. Here, the rule antecedents for discrete attributes are disjointed from those for continuous attributes. Subsequently Hayashi at al. presented the *three-MLP Ensemble* by the Re-RX algorithm [15].

With the advent of deep architectures, substantial improvements were accomplished for many classification problems in computer vision. However, very few works have investigated rule extraction from neural networks trained by deep learning, the main difficulty being the complexity of knowledge representation in the hidden layers [2]. An open question is whether their embedded knowledge is more complex or not than that related to shallow architectures. Moreover, rule extraction from deep networks could contribute to better characterize how the predictive accuracy is improved with respect to shallow architectures. The first work on rule inference from Deep Belief Networks was achieved by Garcez and Tran [16]. Note however that the behavior of these stochastic networks is very different with respect to MLPs, which are deterministic.

Zilke introduced a rule extraction algorithm that was applied to deep MLPs [17] by extending an already existing decompositional method [18]. Specifically, this rule extraction technique uses C4.5 decision trees [19]. Essentially, rules are generated from the activations of the last hidden layer with respect to the classes coded in the output layer. By going toward the input layer, rules are generated from the activations of two successive layers. At the end, after aggregation of the rules and by transitivity, rules describes relations between the input layer and the output layer. To reduce computational time, pruning is executed in order to remove the less important components of a deep network. Although on the MNIST problem this technique did not obtain very accurate rules, on other datasets it yielded good accuracy results.

3 The DIMLP model

In [20] we presented the *Interpretable Multi Layer Perceptron* (IMLP), from which symbolic rules can be generated in an easier way with respect to a standard MLP. An example of this network is shown in Figure 2.

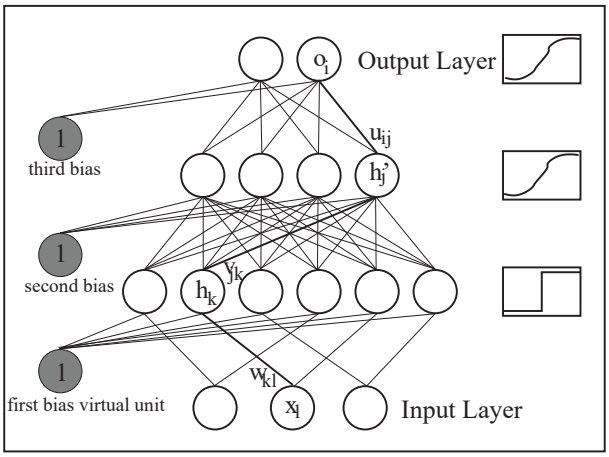

Figure 2. An example of an IMLP network with two hidden layers.

Later we presented the DIMLP model [3], which is a generalization of the IMLP model and finally we demonstrated how to generate symbolic rules from DIMLP ensembles [5].

3.1 IMLP networks

The main difference between an IMLP network and a standard MLP resides in the connectivity configuration between the input layer and the first hidden layer. A hidden neuron is connected to only an input neuron (and the bias virtual neuron), with its activation function being a step function. The output h_k of the k$^{\text{th}}$ neuron of the first hidden layer is

$$h_k = \begin{cases} 1 & \text{if } \sum_l w_{kl} \cdot x_l > 0 \\ 0 & \text{otherwise} \end{cases} . \qquad (2)$$

In the layers above the first hidden layer the activation function is a sigmoid function (cf. eq. 1). As a result, discriminative boundaries are linear and parallel to the axis. To clarify this property an example is shown in Figure 3.

The role of the step activation function is to define possible discriminative hyperplanes that are precisely determined with the weight values of the incoming connections. For instance, in Figure 3 we define a problem with two different classes. Let us assume that the first is characterized when $y_1 > 0.5$ and the second with $y_1 \leq 0.5$. Thus, a possible hyperplane decision boundary could be located in $-w_0/w_1$. Nevertheless, only when $v_1 > |v_0|$ this hyperplane is a discriminative frontier.

Figure 4 illustrates an example with two discriminative hyperplanes, while in Figure 5 is shown an IMLP network unable to learn a classification problem of two classes. Specifically, weights between the hidden layer and the output layer cannot create two discriminative hyperplanes, because the activation values of h_1 and h_2 defines a logical XOR problem, which is not linearly separable. By adding a second hidden layer this classification problem is solved, because the XOR problem defined in the space of binary neurons h_1 and h_2 that is non-linearly separable requires one more hidden layer.

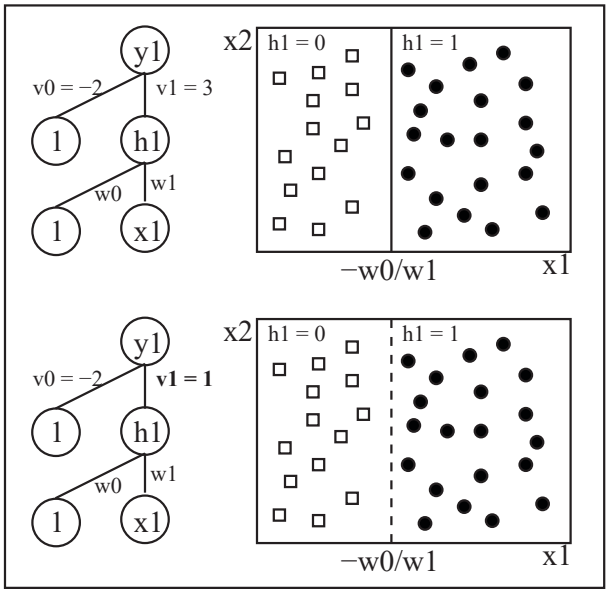

Figure 3. Example of IMLP networks with a unique hidden layer. The network at the top creates a discriminative hyperplane that depends on weight values v_0 and v_1.

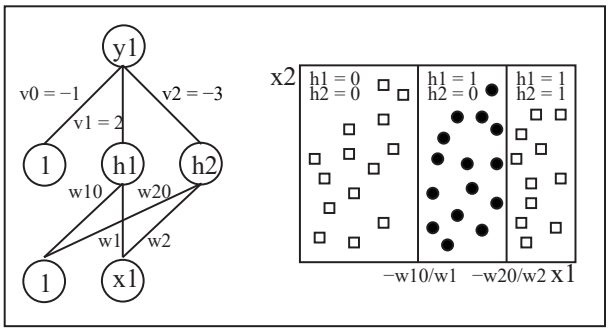

Figure 4. An IMLP network creating two discriminative hyperplanes.

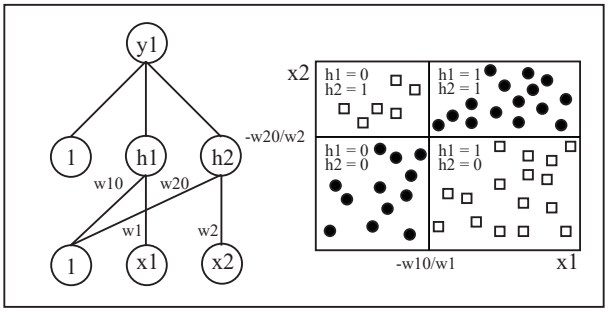

Figure 5. An IMLP network unable to correctly classify a problem of two classes, since with a step function in the hidden layer the activation values of h_1 and h_2 define a XOR problem.

The training of an IMLP network was performed by simulated annealing [20], since the gradient is undefined with step activation functions. A faster learning algorithm for IMLPs with only one hidden layer could be the one used for *Extreme Learning Machines* [21]. Specifically, weights between the input layer and the hidden layer are fixed with random values, then weights between the hidden layer and the output layer are calculated with the pseudo-inverse of the weight matrix. Finally, rules are generated from IMLPs by performing simplification of the boolean expressions related to the activations of the first hidden layer and the class determined by the network.

3.2 DIMLP networks

DIMLP networks are a generalization of IMLP networks. In the first hidden layer the activation function becomes a staircase function that approximates the sigmoid function $\sigma(x)$. Specifically, with R_{min} and R_{max} representing range bounds equal to -5 and 5, the staircase function $S(x)$ is defined as

$$S(x) = \sigma(R_{min}) \quad \text{if } x \leq R_{min}, \tag{3}$$

$$S(x) = \sigma(R_{max}) \quad \text{if } x \geq R_{min}, \tag{4}$$

$$S(x) = \sigma(R_{min} + \left[q \cdot \frac{x - R_{min}}{R_{max} - R_{min}} \right] (\frac{R_{max} - R_{min}}{q})), \tag{5}$$

with $R_{min} <= x <= R_{max}$, q is the number of steps, and "[]" denotes the integer part notation. This function as well as the sigmoid are illustrated in Figure 6 with 50 steps ($q = 50$).

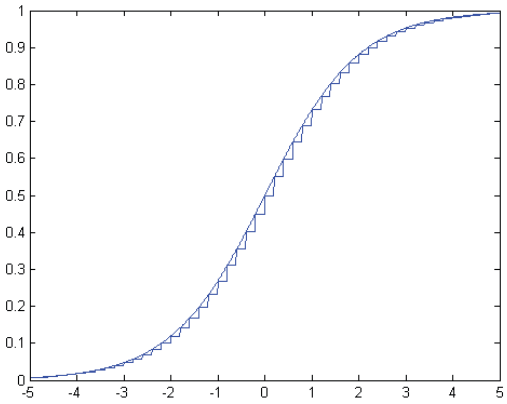

Figure 6. A staircase function with 50 steps that approximates a sigmoid function.

A DIMLP network with staircase activation functions having one step is an IMLP network. As a result, each hidden neuron belonging to the first hidden layer of DIMLPs defines q virtual hyperplanes (cf. Figure 3 with $q = 1$). Thus, rule extraction can be performed by an algorithm whose purpose is to determine whether virtual hyperplanes becomes effective or not.

3.2.1 Rule extraction

A distinctive feature of our rule extraction technique is that fidelity, which is the degree of matching between network classifications and rules' classifications is equal to 100%, with respect to the training set. Here we describe the general idea behind the rule extraction algorithm, since more details are described in [5]. The relevance of a discriminative hyperplane corresponds to the number of points viewing this hyperplane as the transition to a different class. In the first step of the rule extraction algorithm the relevance of discriminative hyperplanes is estimated from all training examples and DIMLP responses.

Once the relevance of discriminative hyperplanes has been established a special decision tree is built according to the strongest relevant hyperplane criterion. In other terms, during tree induction in a given region of the input space the hyperplane having the largest number of points viewing this hyperplane as the transition to a different class is added to the tree.

Each path between the root and a leaf of the obtained decision tree corresponds to a rule. At this stage rules are disjointed and generally their number is large, as well as their number of antecedents. Therefore, a pruning strategy is applied to all rules according to the most enlarging pruned antecedent criterion. The use of this heuristic involves that at each step the pruning algorithm prunes the rule antecedent which most increases the number of covered examples without changing DIMLP classifications. Note that at the end of this stage, rules are no longer disjointed and unnecessary rules are removed.

When it is no longer possible to prune any antecedent or any rule, again, to increase the number of covered examples by each rule all thresholds of remaining antecedents are modified according to the most enlarging criterion. More precisely, for each attribute new threshold values are determined according to the list of discriminative hyperplanes. At each step, the new threshold antecedent which most increases the number of covered examples without altering DIMLP classifications is retained.

The general algorithm is summarized as:

1 Determine relevance of discriminant hyperplanes using available examples

2 Build a decision tree according to the highest relevant hyper-plane criterion.

3 Prune rule antecedents according to the most enlarging pruned antecedent criterion.

4 Prune unnecessary rules.

5 Modify antecedent thresholds according to the most enlarging criterion.

3.2.2 Training algorithm

Training is carried out with the use of a backpropagation algorithm minimizing the *Sum Squared Error function* (SSE)

$$SSE = \frac{1}{2}\sum_{p}\sum_{i}(t_{pi} - o_{pi})^2, \tag{6}$$

with p related to training examples and i related to output neurons. With the use of the staircase activation function the SSE is not differentiable. Thus, we introduced a training algorithm exploiting the fact that with a sufficient number of steps the SSE values calculated with staircase functions is very close to that calculated with sigmoid functions. Specifically, during training the gradient is determined in all the layers with the use of sigmoid functions, whereas the error related to the stopping criterion is calculated with staircase activation functions in the first hidden layer.

3.3 Ensembles of DIMLP networks

We train ensembles of DIMLPs by bagging [22] or arcing [23]. With bagged ensembles each neural network has a slightly different training set. Specifically, bagging selects for each classifier a number of samples drawn with replacement from the original training set. As a result, some samples are absent, while others are repeated.

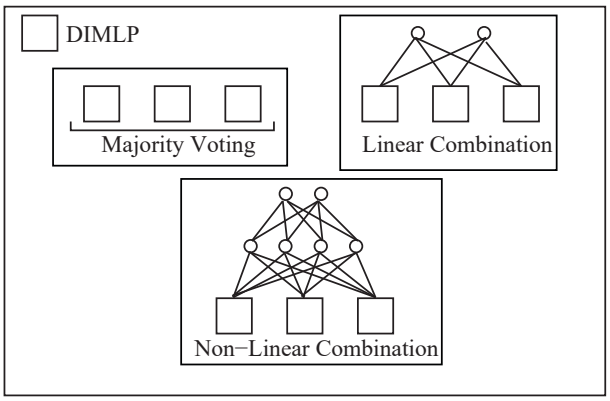

Figure 7. Examples of DIMLP ensembles: majority voting; linear combination; and non-linear combination.

Arcing is a boosting algorithm. In arced ensembles (e.g. ensembles trained by arcing) a probability is associated to each sample. The selection of training samples for each DIMLP network is performed according to these probabilities. For the first network the training samples have the same probability. Then, after the first classifier has been trained the probability of sample selection in a new training set is increased for all unlearned samples and decreased for the others.

The rule extraction technique defined for single DIMLPs can be used for several combined classifiers, since an ensemble of DIMLP networks can be viewed as a single DIMLP network with one more hidden layer for which weight values between dif-

ferent networks are equal to zero. Figure 7 illustrates three different type of DIMLP ensembles.

3.4 DIMLP networks trained by deep learning

We basically use stacked autoencoders as the basic mechanism to obtain deep DIMLPs, denoted as *DIMLP-D*. The first autoencoder has three layers, the output layer corresponding to the input layer. After the training of the first autoencoder, the third layer as well as the incoming weights are discarded and then a second autoencoder can be defined with an input layer corresponding to the hidden layer of the previous autoencoder. The second autoencoder is different from the first, since it has full connectivity between the input layer and the hidden layer. The process can be repeated an arbitrary number of times with autoencoders having full connectivity between successive layers. Figure 8 illustrates the first autoencoder of DIMLP-D.

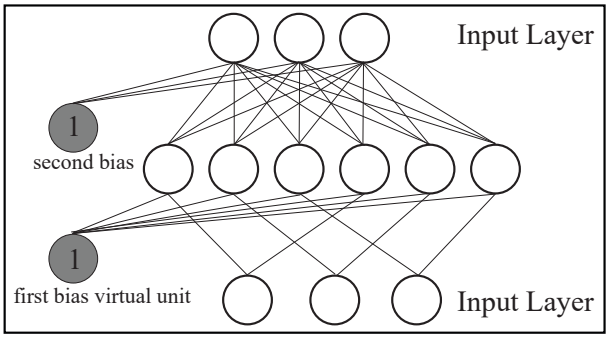

Figure 8. A first autoencoder to train DIMLP-D.

After the last autoencoder has been defined the whole deep network is retrained with respect to the target values of the classification problem. Finally, rule extraction is performed with the same algorithm presented in Section 3.2.1, since it can be applied to any DIMLP architecture having any number of hidden layers.

Denoising autoencoders represent a variant in which the basic idea is to force the hidden layer to create more robust features [24]. Specifically, the autoencoder is trained to reconstruct the input from a corrupted version of it. Basically, the corruption process randomly sets some input components to zero. The proportion of corrupted inputs is often defined between 30%-50%.

[1] D is equal to 28 for MNIST.

4 Experiments

We applied DIMLP networks to two classification problems in the medical domain and a classification task related to computer vision. Deep learning was only applied to the last problem, since it has been observed that appropriate classification tasks for deep learning are much more likely to occur in computer vision, speech recognition and natural language processing. The three datasets we used are available publicly:

- Thyroid dataset: UCI archive: http://archive.ics.uci.edu/ml/
- Wisconsin Breast Cancer dataset: UCI archive: http://archive.ics.uci.edu/ml/
- MNIST dataset: http://yann.lecun.com/exdb/mnist/

Learning parameters were set to default values for both DIMLP ensembles and DIMLPs trained by deep learning. The values are:

- learning parameter: $\eta = 0.1$;
- momentum: $\mu = 0.6$;
- *Flat Spot Elimination*: $FSE = 0.01$;
- number of stairs in the stair function: $q = 50$.

To facilitate visualization with respect to the MNIST problem, for each rule we calculated the average of pixel values for all the covered samples. Specifically, inputs are represented by a 2D matrix; given a rule R_i, and a set A_i containing k_i training samples x^l_{jk} activating this rule, the centroid C^i_{jk} $(j,k = 1,\ldots D)$[1] associated to rule R_i is

$$C^i_{jk} = \frac{1}{k_i} \cdot \sum_{n=1}^{k_i} x^n_{jk}; \quad x^n_{jk} \in A_i. \tag{7}$$

Rule antecedents are represented by colored dots on pictures of rule centroids. Specifically, for the sake of simplicity green dots are used to represent antecedents given as $a_i > t_i$, a_i being a rule antecedent and $t_i \in I\!R$ being a threshold. Similarly, red dots are used to characterize antecedents given as $a_j \leq t_j$. When a rule antecedent is both red and green it is represented in yellow.

4.1 Thyroid dataset

The Thyroid dataset [25] is split into training and testing sets of 3772 and 3428 examples, respectively. Three classes are defined: "Normally Functioning" (Class 1); "Primary Hypothyroidism" (Class 2); "Primary Hyperthyroidism" (Class 3). On the overall 7200 samples of the training and testing sets, 92.6% are classified as normal, 5.1% as hypothyroidism, and 2.3% as hyperthyroidism. Each data sample has 21 attributes with respectively 15 binary and 6 continuous inputs.

Our best result was obtained with a DIMLP architecture of 21 inputs, 42 neurons in a unique hidden layer and three output neurons. We trained ensembles of three networks trained by bagging (DIMLP-B). Early-stopping was made possible by out-of-bag samples. The obtained training accuracy was equal to 99.95% and the predictive accuracy of rules reached 99.50%.

In Table 1 we report the details of two extracted rulesets. The first is the best in terms of rules' predicitive accuracy, while the second presents less rules with lower predicitve accuracy, but still competitive with state of the art results. Fidelity is reported in this table as the degree of matching between rule classifications and network responses on the testing test (fidelity on the training set is 100%). In the fifth row is shown the predictive accuracy of rules when a neural network model agrees with the extracted rules. The number of default rule activations reported in the last row represents how many testing samples were uncovered. Note that in such a situation, the classification is provided by the neural network.

Rule 1: $(\neg x3)$ $(\neg x8)$ $(x17 > 0.0061)$ $(x18 < 0.0336)$ $(x19 < 0.1510)$ $(x21 > 0.0645)$ Class 2

Rule 2: $(\neg x3)$ $(\neg x8)$ $(x17 > 0.0061)$ $(x19 < 0.1510)$ $(x20 > 0.0935)$ $(x21 > 0.0645)$ Class 2

Rule 3: $(\neg x8)$ $(x17 > 0.0061)$ $(x18 < 0.0269)$ $(x21 < 0.0645)$ Class 3

Rule 4: $(x17 > 0.0165)$ $(x19 > 0.0529)$ $(x21 < 0.0645)$ Class 3

Default: Class 1

Figure 9. An example of ruleset extracted from the Thyroid dataset. The accuracy on the training set is 99.95%, while on the testing set it is equal to 99.50%.

The first ruleset is shown in Figure 9, while Figure 10 illustrates the second one. Note that in order to compare our results to the state of the art we replaced the rules of the majority class (Class 1) by the default rule.

Rule 1: $(\neg x3)$ $(\neg x8)$ $(x17 > 0.0061)$ $(x19 < 0.1510)$ $(x21 > 0.0648)$ Class 2

Rule 2: $(x17 > 0.0061)$ $(x21 < 0.0645)$ Class 3

Default: Class 1

Figure 10. An example of ruleset extracted from the Thyroid dataset. The accuracy on the training set is 99.79%, while on the testing set it is equal to 99.36%.

Table 2 compares the results obtained by the rules generated from DIMLP ensembles with state of the art results. Note in the last row that for almost the same level of performance we obtained rules with a lower number of antecedents.

4.2 Wisconsin Breast Cancer dataset

This dataset on breast cancer classification contains 699 samples of two classes: benign and malignant. We removed 16 cases with missing values, hence the total number of samples is equal to 683. Each case is described by nine attributes with integer values in the range $[1, \ldots, 10]$ and a binary class label.

We performed ten repetitions of stratified ten fold cross-validation trials with ensembles of DIMLPs by arcing (DIMLP-A) and bagging (DIMLP-B). The default number of neurons in the first hidden layer was equal to the number of input neurons (9) and the number of neurons in the second hidden layer of the ensembles was empirically defined to be equal to 100.

Ensembles were defined with 25 networks, since it has been observed many times that for bagging and arcing the most substantial improvement in accuracy is achieved with the first 25 networks [5]. Finally, we used early-stopping with out-of-bag samples. Table 3 shows average results and standard deviations. Note that the default rule class here is not replaced by the majority class. The comparison with state of the art results is shown in Table 4.

The average predictive accuracy obtained by the rules extracted from DIMLP-B is similar to that reported in [27]. Note that it is difficult to compare the complexity of rules, since many rule extraction techniques do not report the average number of generated antecedents.

Table 1. Results obtained with DIMLP-B ensembles on the Thyroid dataset. Best predictive accuracy of rules and lowest complexity of rules are in bold.

	DIMLP-B (best rules acc.)	DIMLP-B (less rules)
Training accuracy	99.95	99.79
Predictive accuracy	99.21	99.04
Fidelity	99.64	99.44
Predictive accuracy of rules	**99.50**	99.36
Predictive accuracy of rules 2	99.53	99.47
Number of rules	5	**3**
Avg. number of antecedents per rule	3.8	**2.3**
Avg. number of examples per rule	754.4	1257.3
Number of default rule activations	1	0

Table 2. Comparison of rule extraction algorithms for the Thyroid classification problem. Best accuracy of rules and lowest complexity of rules are in bold.

	Tr. Acc.	Pred. Acc.	#Rules	Avg. #Ant.
Continuous Re-RX [26]	99.05	98.51	5	2.8
C-MLP2LN [27]	99.89	99.36	4	2.5
SSV [27]	99.79	99.33	3	2.7
Fuzzy clustering with ellipsoids [28]	98.10	96.79	25	–
DIMLP-B (1)	**99.95**	**99.50**	5	3.8
DIMLP-B (2)	99.79	99.36	**3**	**2.3**

Table 3. Ten fold cross-validation results obtained with DIMLP-A and DIMLP-B ensembles on the Breast Cancer dataset. Best predictive accuracy of rules and lowest complexity of rules are in bold.

	DIMLP-A	DIMLP-B
Training accuracy	100.0 (0.0)	98.0 (0.1)
Predictive accuracy	96.6 (0.3)	97.1 (0.2)
Fidelity	98.9 (0.3)	98.8 (0.4)
Predictive accuracy of rules	96.2 (0.3)	**96.5 (0.3)**
Predictive accuracy of rules 2	96.9 (0.3)	97.4 (0.2)
Number of rules	25.2 (0.6)	**12.5 (0.5)**
Avg. number of antecedents per rule	3.6 (0.1)	**2.7 (0.0)**
Avg. number of examples per rule	67.1 (3.5)	185.1 (6.4)
Number of default rule activations	0.4 (0.1)	0.5 (0.1)

Table 4. Comparison of rule extraction algorithms for the Wisconsin Breast Cancer classification problem, based on 10-fold cross-validation. Best predictive accuracy of rules and lowest complexity of rules are in bold.

Rule extraction technique	Tr. Acc.	Pred. Acc.	#Rules	Avg. #Ant.
SSV (10 CV) [27]	–	96.3 (0.2)	3	–
FSM (10 CV) [27]	–	96.5	12	–
MINERVA (10 CV) [29]	–	94.5 (1.5)	**4.2**	**3.3**
NeuroLinear + GRG (10 CV) [30]	–	96.0	2	–
Re-RX + J48graft (10 x 10 CV) [31]	96.3 (1.8)	95.8 (1.6)	4.8	1.7
DIMLP-A (10 x 10 CV)	100.0 (0.0)	96.2 (0.3)	25.2	3.6
DIMLP-B (10 x 10 CV)	98.0 (0.1)	**96.5 (0.3)**	12.5	2.7

4.3 Mnist dataset

The MNIST data set is a standard dataset for hand-written digit classification [32]. Digits between zero and nine are represented by 784 inputs with normalized gray levels. The training and testing sets of the MNIST dataset are fixed and contain 60000 and 10000 samples, respectively. Since the rule extraction algorithm applied to DIMLPs is polynomial in terms of algorithmic complexity, it would take too much time to obtain a ruleset from such a high dimensional dataset. However, it is possible to extract rules from smaller classification sub-problems; for instance we can define binary or ternary classification problems from the whole MNIST dataset.

We performed four series of experiments. Firstly, we trained DIMLP networks on digits *5* and *8*, in order to compare our extracted rules with those generated in another work [33]. Secondly, we performed experiments with classification subproblems of two and three classes to determine the predictive accuracy for the whole MNIST dataset. In the third series of experiments the purpose was to distinguish digit *0* from the others and to compare rules generated from deep DIMLPs to those obtained in [17]. Finally, deep DIMLPs were trained on the whole dataset to show that they can reach predictive accuracy similar to that obtained in the state of the art. For illustrative purposes a ruleset was generated from a training set of reduced input dimensionality and a small ensemble of arced DIMLPs.

4.3.1 Recognition of digits *5* and *8*

With the use of Support Vector Machines Cherkassky and Dhar generated rules from a binary classification problem related to digits *5* and *8* [33]. For these two digits we extracted from the whole MNIST dataset 11272 training examples and 1866 testing examples, respectively. The default number of neurons of DIMLP ensembles in the first hidden layer was equal to the number of input neurons and the number of neurons in the second hidden layer was empirically defined to be equal to five.

The architecture for deep learning was determined empirically after a few preliminary experiments. The retained architecture was: 784 input neurons, 784 neurons in the first hidden layer, 200 in the second, 100 in the third, 30 in the fourth and 2 output neurons. Autoencoders were trained for 50 epochs and the whole deep network was retrained for 100 epochs. For all the models, table 5 shows average results and standard deviations of DIMLP ensembles and DIMLPs trained by deep learning.

It turned out that rule complexity in terms of number of extracted rules and number of antecedents per rule was much higher for arced ensembles and deep learning, on average. Moreover, higher complexity involved lower fidelity, but also lower predictive accuracy of the rules with respect to the neural networks they represent. The reason is that the expression power of symbolic rules is much more limited than that of neural networks. However, when rules and DIMLP classifications agreed the predictive accuracy was very high. For instance, for DIMLP-B networks in 98.83% of the testing samples the average predictive accuracy of the rules was 99.0%. Hence, rules did not explain testing samples for 28.1 times ($1.17/100 \cdot 1866 = 28.1$), on average. Due to default rule activations, interpretable rules did not support another 13.1 testing samples. Thus, for DIMLP-B networks the proportion of testing samples not covered by any rule was equal to 2.2%, while it was equal to 7.0% for DIMLP-D. Overall, the more complex a model the more difficult to approximate it with rules.

Cherkassky and Dhar generated rules from SVMs with the ALBA algorithm [34] for digits *5* and *8*. Based on 10 trials, their best result was achieved with the RBF-kernel. The average predictive accuracy was equal to 98.77%, while the average predictive accuracy of the generated rules was equal to 93.52%. As a matter of fact, the number of extracted rules was not reported. Note that the average predictive accuracy of the rules we obtained with DIMLP ensembles and DIMLP-D are: 98.24%; 96.00%; and 96.06%, respectively. It is worth noticing that when the rules agreed with the models (for more than 96% of the testing samples) our accuracies were 99.00%, 99.25% and 99.57%, respectively.

Figure 11 depicts 16 centroids out of 431 calculated from the 16 rules extracted from one of the bagged ensembles. On these centroid pictures, rule antecedents are represented by colored dots. The first 6 centroids on the first row are related to rules covering more than 1000 samples of the training

Table 5. Results obtained with DIMLP networks on the MNIST dataset for digits *5* and *8*.

	DIMLP-B	DIMLP-A	DIMLP-D
Training Accuracy	99.53 (0.04)	**100.00 (0.00)**	99.99 (0.01)
Predictive Accuracy	98.61 (0.09)	98.99 (0.07)	**99.54 (0.10)**
Fidelity	**98.83 (0.21)**	96.47 (0.45)	96.43 (0.09)
Predictive Accuracy of Rules	**98.24 (0.17)**	95.99 (0.55)	96.06 (0.83)
Predictive Accuracy of Rules 2	99.00 (0.05)	99.23 (0.12)	**99.57 (0.09)**
Number of Rules	**262.1 (27.1)**	885.8 (116.2)	879.4 (242.7)
Number of Antecedents per Rule	**7.4 (0.1)**	7.5 (0.3)	7.7 (0.2)
Number of Examples per Rule	**194.6 (19.2)**	56.6 (10.0)	61.6 (31.2)
Number of Default Rule Activations	**13.1 (3.2)**	63.8 (14.0)	63.2 (26.9)

set. Centroids are shown in descending order with respect to the number of covered samples in the training set. Typically, an example activates several rules, since rules are not exclusive. Thus, with respect to the duality of representation between rules and centroids a digit picture is represented by several centroids. Note that many centroids are very similar, since they cover many similar digits. This is also representative of redundant knowledge representations typically found in neural networks.

Red dots tend to appear in dark regions of digit pictures, since the corresponding rule antecedents are true for pixels below given thresholds. On the contrary, green dots tend to be located in the bright areas of the digits, because rule antecedents are true for pixels above given thresholds, thus detecting the presence of digits. The first rule (top left of Figure 11) covers 1705 training samples, two green dots are on the surface of the number and many red dots are around the right upper part. Class of digit *5* is very often characterized by a majority of red dots, thus a majority of rule antecedents detecting absence of intensity. For instance centroids 6, 7, and 8 belong to the class of digit *5* and have a large number of red dots in the holes around the shape of the bright area. Class digit *8* has a tendency to be detected by more green dots than digit *5*.

Figure 12 is like Figure 11, but for a network trained by deep learning. With respect to the observations raised for bagged DIMLPs we do not see major differences.

Qualitatively, no clear difference in the strategy of classification resulted between DIMLP-D and the ensembles. The main difference between DIMLP-B and DIMLP-D resides in the number of rules, which is substantially greater with DIMLP-D. Between DIMLP-D and DIMLP-A the number of extracted rules, as well as the number of antecedents per rule is not statistically different.

Figures 13, 14, and 15 illustrates examples for which rules and networks disagree on the classification decision. Specifically, the left sub-Figure depicts an example that is correctly classified by DIMLP-D, while the right sub-Figure corresponds to the centroid of the activated rule (yielding a wrong classification). As you can see, the five features are not sufficiently specialized to disambiguate the digit belonging to class *5* and the centroid belonging to class *8*. With respect to the latter, if the neural network had created more features (e.g. rule antecedents) in different areas close to the borders, the wrong classification would be likely to vanish. In fact, the improved features would become much more specific to class *8* and then digit *5* would not be covered by it.

Figure 13. The sample at left is correctly classified by DIMLP-D, but it is covered on the right by a rule of the wrong class (a yellow dot represents a dot which is both red and green).

Figure 14. A second case for which the sample at left is correctly classified by DIMLP-D, but it is covered on the right by a rule of the wrong class.

Figure 11. First 16 centroids of a bagged ensemble with rule antecedents represented by colored dots.

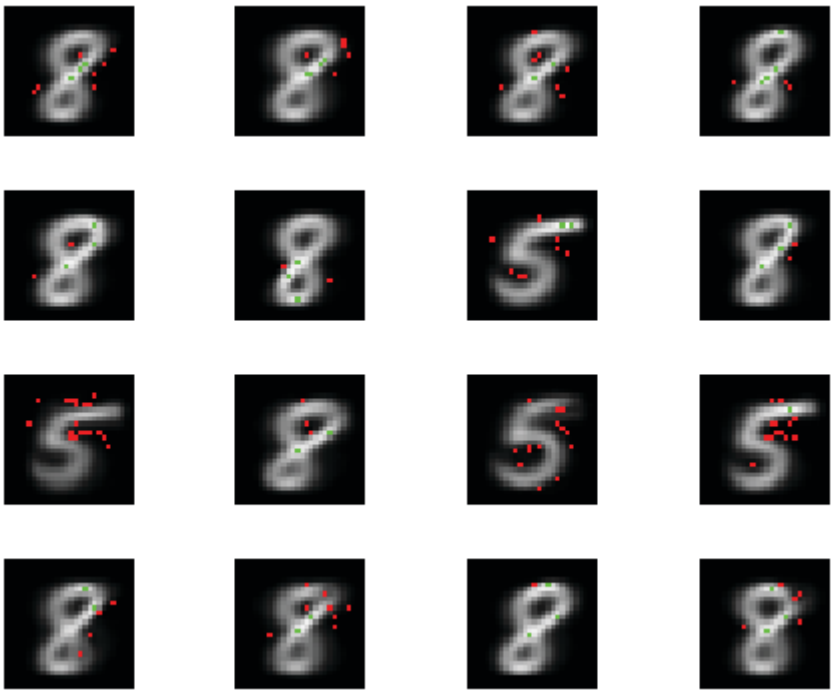

Figure 12. First 16 centroids and rule antecedents of a DIMLP network trained by deep learning.

Figure 15. A third case for which the sample at left is correctly classified by DIMLP-D, but it is covered on the right by a rule of the wrong class.

Figure 16 represents an example of wrong classification, but for which rules and DIMLP-D agree. Finally, Figure 17 illustrates an example in which the activated rule is correct, but with wrong DIMLP-D classification.

Figure 16. The sample at left is misclassified by both DIMLP-D and the activated rule on the right.

Figure 17. The sample at left is correctly classified by the activated rule on the right, but wrongly classified by DIMLP-D.

To decrease the number of extracted rules from deep DIMLPs and bagged DIMLPs, during the learning phase we only used the first 1000 samples of the training set. The DIMLP architectures were identical to those defined above. For deep DIMLPs the only difference was that when the whole network was retrained by supervised learning, we stopped learning after three epochs. For bagged DIMLPs the training of each single network was stopped after it went above 97% accuracy on the training set of 1000 samples. Results are shown in table 6. Compared to previous experiments, the number of rules of deep DIMLPs was reduced by a factor equal to 13.3, while the predictive accuracy of rules decreased to 93.68%, on average. By sacrificing predictive accuracy we clearly obtained lower complexity rulesets, especially for bagged DIMLPs with 47.0 generated rules, on average.

Figure 18 illustrates all the centroids generated from a bagged ensemble. We clearly see that the created features are simpler than those obseved in more accurate rulesets.

4.3.2 Combinations of binary and ternary classification sub-problems

Generally, a classification problem of more than two classes can be transformed into several binary sub-problems. The One-Against-All decomposition defines a binary classification problem in which the purpose is to distinguish a class versus all the others. With ten classes we obtain ten binary classification problems. Another method consists in taking into account all possible combinations of binary classification tasks. This is sometimes referred as *Tournament Learning*; hence, with ten classes we obtain 45 classification sub-problems. For each sample all the classifiers are combined in an ensemble to produce a classification related to the most activated class.

A clear advantage of this framework is that for the MNIST problem each classifier is trained with approximately 20% of the original training set, thus training and rule extraction is faster. Furthermore, the 45 classifiers can be trained in parallel. The same approach can be repeated by considering classification sub-problems of three classes for a total of 120 ternary sub-problems.

For these experiments, we used DIMLP-Ds with the same architectures described in Section 4.3.1. Table 7 illustrates the average predictive accuracy results obtained by aggregating 45 binary classifiers, as well as 120 ternary classifiers over five trials. Note that rule extraction from ensembles of 45 or 120 DIMLP-Ds was not performed, since it would have been too long to execute it.

Figures 19 and 20 illustrate the first 25 centroids and their rule antecedents represented by colored dots.

Table 8 summarizes the results obtained by eight DIMLP-Ds trained to recognize three digit classes, including digit *5* and digit *8*. With respect to binary problems although the number of training samples increased, the average number of extracted rules decreased from 879.4 to 678.0, while the average fidelity increased from 96.43% to 97.46%. Finally, the predictive accuracy of rules increased from 96.06% to 96.95%

Table 6. Results obtained with less accurate deep DIMLPs and bagged DIMLPs on the MNIST dataset for digits *5* and *8*.

	DIMLP-B	DIMLP-D
Predictive Accuracy	94.56 (0.15)	95.25 (0.71)
Fidelity	96.91 (0.50)	95.47 (1.31)
Predictive Accuracy of Rules	93.44 (0.42)	93.68 (0.81)
Predictive Accuracy of Rules 2	95.40 (0.33)	96.57 (0.67)
Number of Rules	47.0 (5.2)	66.2 (6.1)
Number of Antecedents per Rule	4.7 (0.2)	5.3 (0.2)
Number of Default Rule Activations	27.0 (12.1)	45.7 (15.5)

Figure 18. All the centroids generated from a bagged ensemble with rule antecedents represented by colored dots. The predictive accuracy of the ensemble is 94.69%, while for the rules it is equal to 93.57%.

Table 7. DIMLP-D results of average predictive accuracy for ensembles of binary and ternary sub-classifiers.

	Sub-problems of two classes	Sub-problems of three classes
Pred. Acc.	98.11 (0.11)	98.27 (0.05)

Figure 19. First 25 centroids and rule antecedents for a DIMLP-D trained on digits *1*, *5*, and *8*.

Table 8. Average results obtained from eight DIMLP-Ds trained on eight ternary classification sub-problems including class of digits *5* and *8*.

	DIMLP-D
Training Accuracy	99.47 (0.62)
Predictive Accuracy	98.77 (0.57)
Fidelity	97.46 (0.80)
Predictive Accuracy of Rules	96.95 (0.87)
Predictive Accuracy of Rules 2	99.18 (0.35)
Number of Rules	678.0 (148.6)
Number of Antecedents per Rule	8.61 (0.46)
Number of Examples per Rule	97.7 (21.1)
Number of Default Rule Activations	34.5 (6.7)

Figure 20. First 25 centroids and rule antecedents for a DIMLP-D trained on digits *3*, *5*, and *8*.

4.3.3 Recognition of digit *0*

Zilke applied a deep architecture to distinguish digit *0* from all other numbers [17]. Specifically, from the whole MNIST dataset all the samples of class digit *0* were included in this binary classification problem. As reported in [17], the training and testing sets include 12056 and 2195 samples, respectively. Note that the number of samples representing digits different from digit *0* corresponded approximately to the number of digits of class *0*, after random selection.

For deep DIMLPs we defined the architecture used in Sect. 4.4.1. We performed ten trials with ten different training and testing sets, since digits different from class *0* were randomly selected from the whole MNIST dataset. Rule extraction was first performed on the 12056 training samples and then on the first 1500 samples, in order to generate less symbolic rules. Note also that Zilke carried out only a trial by extracting rules with two different sets of parameters. Results are depicted in table 9. The first column of results is related to rule extraction from DIMLP-D trained with all training samples of this binary classification problem, the second column summarizing the results obtained by DIMLP-D trained with only 1500 samples.

We report in the last column of table 9 the results obtained in [17] related to two rulesets. Essentially, extracted rules belong to class *0* and contain about 200 antecedents. Our extracted rulesets, being based on ten different datasets present 253.6 antecedents for the two classes, on average (about half of them for class *0*). However, our rules' predictive accuracy is much higher.

4.3.4 Whole MNIST dataset

We performed two series of experiments using all the samples of the MNIST dataset (60000 training samples and 10000 testing samples). In the first series we wanted to determine whether a deep DIMLP architecture is able to reach a predictive accuracy similar to that obtained in the state of the art. As reported in [32], without using supplementary distorted training samples the best predictive accuracy for a deep MLP was 98.4%.

After a few preliminary experiments we defined a DIMLP architecture with: 784 input neurons, 784 neurons in the first hidden layer, 1000 in the second, 1000 in the third, 1000 in the fourth and 10 output neurons. Stacked denoising autoencoders were trained for only 2 epochs and the whole deep network was retrained for 40 epochs. On ten trials our average predictive accuracy was 98.63% $\pm$0.04. Hence, although the DIMLP model represents a particular MLP architecture it yields very accurate results.

Since the previous DIMLP architecture presents more than three million weights it could require a long time to generate symbolic rules. As a consequence, in a second series of experiments we decided to use arced DIMLP ensembles. To reduce rule extraction execution time we downsampled the MNIST dataset by replacing 2x2 block pixels by their average. Hence, the data dimensionality was reduced by a factor equal to four. As shown in table 10, our best results were obtained with arced ensembles of 100 DIMLP networks having 196 neurons in the first hidden layer, 100 neurons in the second hidden layer and ten neurons in the output layer.

Another input reduction was performed with a resulting number of inputs equal to 121. Specifically, we replaced 3x3 pixel blocks close to the corners by their average. Getting away from the corners, the average was calculated for 2x2 blocks. The number of neurons of arced ensembles in the second hidden layer was equal to 150.

To make it possible to generate rules from the whole MNIST dataset we used a small ensemble of three DIMLP networks having 121 input neurons. Not surprisingly, the predictive accuracy of this small ensemble was lower than the average predictive accuracy obtained by full ensembles (97.97 versus 98.36 ± 0.03). Figure 21 illustrates the first 64 centroids of the generated ruleset.

Tran and Garcez generated symbolic rules from *Deep Beliefs Networks* (DBN) based on stacked Restricted Boltzman Machines [16]. They trained a network on the whole MNIST dataset and obtained a predictive accuracy of 97.63%. They generated an unknown number of rules with predictive accuracy equal to 93.97%. The number of antecedents per rule was equal to the input dimensionality, which is 784. We present our results in Table 11. Note that the results depicted in the first column are those related to the 64 centroids illustrated in Figure 21. We varied the number of extracted rules from 65 to 400

	DIMLP-D (full)	DIMLP-D (partial)	DeepRED [17]
Predictive Accuracy	99.37 (0.09)	97.87 (1.46)	≈ 99.5
Fidelity	97.05 (0.45)	96.83 (0.63)	≈ 89.5 and ≈ 87.0
Predictive Accuracy of Rules	**96.69 (0.45)**	**96.14 (0.83)**	≈ 89.5 and ≈ 87.0
Predictive Accuracy of Rules 2	99.49 (0.08)	98.54 (0.84)	—
Number of Rules	831.9 (30.2)	53.8 (6.2)	—
Number of Antecedents	6065.9 (258.7)	253.6 (35.4)	≈ 200
Number of Default Rule Activations	308.8 (19.1)	159.4 (62.6)	—

Table 9. Results obtained by deep DIMLPs trained to recognize digit *0* with full and partial training sets.

	DIMLP-A (196 inputs)	DIMLP-A (121 inputs)
Pred. Acc	**98.39 (0.05)**	98.36 (0.03)

Table 10. Results of DIMLP ensembles trained by arcing on the whole MNIST dataset after dimensionality reduction.

Figure 21. First 64 centroids and rule antecedents for an ensemble of three networks trained on the whole MNIST dataset.

by ranking the list of rules according to the number of covered samples in descending order. Overall, we obtained more than 70 times less antecedents per rule, with better predictive accuracy.

4.4 Summary and Discussion

On the Thyroid dataset DIMLPs trained by bagging obtained very good results and very compact rulesets. This is consistent with our previous work in which bagging tends to reduce ruleset complexity with respect to arcing [35]. In this work for the Breast Cancer problem as well as for classification subproblems related to MNIST, we observed a similar trend.

On a particular subset of the MNIST dataset involving digit *5* and *8*, deep DIMLPs yielded more accurate results than DIMLP ensembles, whereas the average complexity of rules related to deep DIMLPs was similar to that measured with arced ensembles of DIMLPs. Moreover, the comparison with another work on rule extraction from SVMs trained on the same dataset was favorable to our generated rulesets in terms of predictive accuracy. For another binary classification problem involving the recognition of digit *0*, rules generated from deep DIMLPs resulted more accurate than that extracted in another work (at the same level of rule complexity).

Rules generated from binary or ternary MNIST classification sub-problems emphasized that deep DIMLPs or ensembles developed discriminatory features in particular digit areas. In many cases these features detect values below a threshold, which can be viewed as detecting the absence of digits in areas close to the borders, rather than their presence. Whether the one or the other does not change anything; overall, the concomitant action of a number of features makes it possible to perform the classification. Although many of the generated rule centroids look similar, they still present differences, such as small rotations (cf. digit *1* in Figure 19). Finally, for MNIST binary classification problems the complexity of rules in terms of number of rules and number of antecedents per rule was very similar for deep DIMLPs and ensembles of arced DIMLPs. As a consequence, an interesting question to elucidate in the future is whether deep learning produce in general rules similar to that extracted from boosted ensembles.

With the whole MNIST dataset we illustrated a trade-off between the number of generated rules, rules accuracy and rules' covering. Essentially, we can control the number of extracted rules by sacrificing their covering. Hence, depending also on the application domain, a user could be interested to have at the beginning a general picture of the most important rules (e.g. those with the highest covering). Contrary to ordered rules, unordered rules with their respective centroids can be analyzed individually as modular pieces of knowledge. At a latest stage, one could go further through the details by analyzing rules with lower covering.

Conclusion

We presented a study on rule extraction from ensembles of DIMLP networks and deep DIMLPs. On two benchmark classification problems of medical diagnosis we generated accurate rules from DIMLP ensembles which revealed to be competitive with respect to the state of the art. We trained deep DIMLPs of four hidden layers on a benchmark dataset in the computer vision domain.

For the MNIST dataset we first defined classification sub-problems of two or three classes. The comparison with another work that applied rule extraction to SVMs trained on digit *5* and digit *8* was favorable to our DIMLPs deep networks. Rule extraction emphasized feature detectors that very often discriminate classes in regions close to digit borders. Moreover, we performed experiments of deep DIMLPs on the discrimination of digit *0*. Comparing our results with those of another work was again in our favor.

In the last experiments we generated a ruleset from the whole MNIST dataset. To the best of our knowledge for a deep feed-forward architecture this has been carried out for the first time. This technique has a potential application to reliability analysis of fully automated driving car at the level-4 with the use of deep DIMLPs and image data from driver console. Finally, for Computer Vision classification problems a new general question to elucidate in the future is whether it will be possible to improve predictive accuracy using the extracted feature detectors.

	DIMLP-D	DIMLP-D	DIMLP-D	DBN [16]
Fidelity	99.09	97.70	96.26	—
Predictive Accuracy of Rules	97.16	96.01	94.76	93.97
Number of Rules	65	200	400	—
Avg. Number of Antecedents per Rule	11.1	10.4	10.4	784
Number of Default Rule Activations	7057	5297	4039	—

Table 11. Results on the whole MNIST dataset for a small arced ensemble of three networks with 121 inputs.

References

[1] M. Golea, On the complexity of rule extraction from neural networks and network querying, in: Rule Extraction From Trained Artificial Neural Networks Workshop, Society For the Study of Artificial Intelligence and Simulation of Behavior Workshop Series (AISB), 1996, pp. 51–59

[2] T. Hailesilassie, Rule extraction algorithm for deep neural networks: A review, International Journal of Computer Science and Information Security 14, 7, 2016, 376

[3] G. Bologna, Symbolic rule extraction from the dimlp neural network, in: Hybrid neural systems, Springer, 2000, pp. 240–254

[4] G. Bologna, A study on rule extraction from several combined neural networks, International journal of neural systems 11, 03, 2001, 247–255

[5] G. Bologn, Is it worth generating rules from neural network ensembles?, Journal of Applied Logic 2, 3, 2004, 325–348

[6] A. A. Freitas, Comprehensible classification models: a position paper, ACM SIGKDD explorations newsletter 15, 1, 2014, 1–10

[7] J. Chorowski, J. M. Zurada, Learning understandable neural networks with nonnegative weight constraints, Neural Networks and Learning Systems, IEEE Transactions on 26, 1, 2015, 62–69

[8] S. I. Gallant, Connectionist expert systems, Communications of the ACM 31 (2) (1988) 152–169.

[9] R. Andrews, J. Diederich, A. B. Tickle, Survey and critique of techniques for extracting rules from trained artificial neural networks, Knowledge-based systems 8, 6, 1995, 373–389

[10] J. Diederich, Rule extraction from support vector machines, Vol. 80, Springer Science & Business Media, 2008

[11] L. K. Hansen, P. Salamon, Neural network ensembles, IEEE transactions on pattern analysis and machine intelligence 12, 1990, 993–1001

[12] Z.-H. Zhou, Y. Jiang, S.-F. Chen, Extracting symbolic rules from trained neural network ensembles, Artificial Intelligence Communications 16 , 1, 2003 3–16.

[13] R. Setiono, B. Baesens, C. Mues, Recursive neural network rule extraction for data with mixed attributes, Neural Networks, IEEE Transactions on 19 , 2, 2008, 299–307

[14] A. Hara, Y. Hayashi, Ensemble neural network rule extraction using re-rx algorithm, in: Neural Networks (IJCNN), The 2012 International Joint Conference on, IEEE, 2012, pp. 1–6

[15] Y. Hayashi, R. Sato, S. Mitra, A new approach to three ensemble neural network rule extraction using recursive-rule extraction algorithm, in: Neural Networks (IJCNN), The 2013 International Joint Conference on, IEEE, 2013, pp. 1–7

[16] S. N. Tran, A. dAvila Garcez, Knowledge extraction from deep belief networks for images, in: IJCAI-2013 Workshop on Neural-Symbolic Learning and Reasoning, 2013

[17] J. Zilke, Extracting rules from deep neural networks, Master's thesis, Computer Science Department, Technische Universitt Darmstadt, 2015

[18] R. Setiono, W. K. Leow, Fernn: An algorithm for fast extraction of rules from neural networks, Applied Intelligence 12 , 1-2, 2000, 15–25

[19] J. R. Quinlan, C4.5: Programs for machine learning. morgan kaufmann publishers, inc., 1993, Machine Learning 16, 3, 1994, 235–240

[20] G. Bologna, C. Pellegrini, Constraining the mlp power of expression to facilitate symbolic rule extraction, in: Neural Networks Proceedings, 1998, IEEE World Congress on Computational Intelligence. The 1998 IEEE International Joint Conference on, Vol. 1, IEEE, 1998, pp. 146–151

[21] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: theory and applications, Neurocomputing 70 , 1, 2006, 489–501

[22] L. Breiman, Bagging predictors, Machine learning 24, 2, 1996, 123–140

[23] L. Breman, Bias, variance, and arcing classifiers (technical report 460), Statistics Department, University of California

[24] P. Vincent, H. Larochelle, Y. Bengio, P.-A. Manzagol, Extracting and composing robust features with denoising autoencoders, in: Proceedings of the 25th international conference on Machine learning, ACM, 2008, pp. 1096–1103

[25] M. Lichman, http://archive.ics.uci.edu/ml (UCI machine learning repository 2013)

[26] Y. Hayashi, S. Nakano, S. Fujisawa, Use of the recursive-rule extraction algorithm with continuous attributes to improve diagnostic accuracy in thyroid disease, Informatics in Medicine Unlocked 1, 2015, 1–8

[27] W. Duch, R. Adamczak, K. Grøbczewski, A new methodology of extraction, optimization and application of crisp and fuzzy logical rules, Neural Networks, IEEE Transactions on 12 , 2, 2001, 277–306

[28] S. Abe, R. Thawonmas, M. Kayama, A fuzzy classifier with ellipsoidal regions for diagnosis problems, IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 29 , 1, 1999, 140–148

[29] J. Huysmans, R. Setiono, B. Baesens, J. Vanthienen, Minerva: Sequential covering for rule extraction, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 38 , 2, 2008, 299–309

[30] K. Odajima, Y. Hayashi, G. Tianxia, R. Setiono, Greedy rule generation from discrete data and its use in neural network rule extraction, Neural Networks 21 , 7, 2008, 1020–1028

[31] Y. Hayashi, S. Nakano, Use of a recursive-rule extraction algorithm with j48graft to achieve highly accurate and concise rule extraction from a large breast cancer dataset, Informatics in Medicine Unlocked 1, 2015, 9–16

[32] Y. LeCun, C. Cortes, C. Burges, The mnist database of handwritten digits, 1998, 2012, Available electronically at http://yann.lecun.com/exdb/mnist

[33] V. Cherkassky, S. Dhar, Interpretation of black-box predictive models, in: Measures of Complexity, Springer, 2015, pp. 267–286

[34] W. Verbeke, D. Martens, C. Mues, B. Baesens, Building comprehensible customer churn prediction models with advanced rule induction techniques, Expert Systems with Applications 38 , 3, 2011, 2354–2364

[35] G. Bologna, Y. Hayashi, Qsvm: A support vector machine for rule extraction, in: International Work-Conference on Artificial Neural Networks, Springer, 2015, pp. 276–289

Guido Bologna is Senior Lecturer at the University of Applied Sciences of Western Switzerland since 2011, and Senior Scientist with the Computer Vision and Multi-Media Lab of the University of Geneva, Switzerland. He received a Ph.D. in Computer Science in 1998 at the University of Geneva for the development of a new neural network model from which symbolic rules are easily extracted. Subsequently, he worked as a researcher at the Queensland University of Technology, Brisbane; at the National University of Singapore and at the Swiss Institute of Bioinformatics. His research interests focus on rule extraction from neural networks, computer vision and multimodal interfaces for blind users. He has authored or co-authored over 85 full papers in conference proceedings, book chapters and journal articles.

Prof. **Yoichi Hayashi** received the Dr. Eng. degree in Systems Engineering from the Tokyo University of Science, Tokyo in 1984. In 1986, he joined the Computer Science department of Ibaraki University, Japan, as an Assistant Professor and was a Visiting Professor of Computer Science Department of the University of Alabama at Birmingham and the University of Canterbury, respectively. Since 1996, he is a full Professor of Computer Science department at Meiji University, Tokyo. He is the author of over 230 papers published in computer science. His current research interests include Big Data analytics, high performance classifier, Deep Learning, high accuracy rule extraction and medical informatics. He is the editorial board member of Artificial Intelligence in Medicine and was the Associate Editor of the IEEE Transactions on Fuzzy Systems, the Action Editor of Neural Networks. He is a senior member of the IEEE since 2000.