

Multi-instance Learning using Recurrent Neural Networks

A. S d'Avila Garcez - Department of Computing, City University London
Northampton Square, London EC1V 0HB UK

Email: aag@soi.city.ac.uk

G. Zaverucha - Programa de Engenharia de Sistemas e Computação
COPPE/UFRJ, Caixa Postal 68511, 21945-970, Rio de Janeiro, Brazil

Email: gerson@cos.ufrj.br

Abstract—Multiple instance learning is an increasingly important area in machine learning. In multi-instance learning, the training set is structured into subsets (or bags) of instances. The bags are labelled, but the label of each instance is unknown or irrelevant. In this paper, we revisit the connectionist approach to multi-instance learning. We propose a recurrent neural network model for multi-instance learning. We have applied the new model to a benchmark multi-instance dataset. The results provide evidence that connectionist multi-instance learning is more promising than previously anticipated. We argue that a principled connectionist approach should provide robust and efficient multi-instance learning, yet comparative results should be taken with caution as a result of varying methodologies.

Index Terms—Multiple Instance Learning, Recurrent Networks, Structured Learning, Neural-Symbolic Integration

I. INTRODUCTION

Multiple instance learning is an increasingly important area in machine learning, relational learning and Inductive Logic Programming (ILP) [1], [2]. In multi-instance learning, the training set is structured into subsets (or bags) of instances. The bags are labelled, but the label of each instance is unknown or irrelevant. If a bag is labelled as a positive example then it is known that at least one of the instances is positive. If, however, a bag is labelled as a negative example then it must be the case that all the instances are negative.

Multi-instance learning (MIL) has been applied to drug development. The fitness of a molecule in the process of drug development depends upon some of the molecule's shapes. Biochemists know that certain molecules will fit, but they do not know, in general, which are the shapes that do. A dataset with these characteristics, known as the musk dataset, has been used for benchmark MIL experiments. The dataset describes 102 molecules with 166 features each. A molecule may have a number of different shapes because the bonds can rotate. Out of the 102 molecules described in the dataset, 39 have been classified by experts as musk, while 63 have been classified as non-musk. The MIL task is to classify the molecules given the set (bag) of features.

More generally, suppose that there are n bags $\{b_1, \dots, b_n\}$ and the i -th bag contains m_i instances. Each instance is a feature vector with elements in $\{-1, 0, 1\}$, denoting respectively *false*, *unknown* and *true*. Let each bag i contain feature vectors of size k_i (different bags may have vectors of different

sizes). Let $q = \max\{k_1, \dots, k_n\}$ denote the size of the largest feature vector in $\{b_1, \dots, b_n\}$.

Since the seminal work in [1], a number of techniques have been proposed to solve the multi-instance problem. In Section IV, we shall compare our experimental results with most of these. For now, we would like to focus attention on [3], which reports 96.8% accuracy on the Musk dataset, [4] which reports good levels of accuracy using simple variations of the nearest-neighbour method, and [5], [6], which use variations of gradient-descent. All of the papers we have checked report, in their comparative results, an accuracy of 75% for the standard backpropagation algorithm [7]. Most of these papers do not mention, however, as reported in the original paper [1], that this result was obtained by a network with 125 hidden neurons. It is quite possible that a network with this number of hidden neurons is overfitting the data. In fact, in our own experiments, using networks with 10 to 40 hidden neurons, a maximum accuracy of 90.79% was obtained by standard backpropagation. Using 10-fold cross-validation, an average accuracy of 86.96% was achieved. In this process, as should be expected, the test set has been kept completely separate from the training. In [3], however, it seems that for each fold, instead of taking the average test set error, the smallest error for each bag is calculated on the test set (see *main* algorithm in [3] which contains the cross-validation algorithm, but calculates $\min(\text{error})$ on the test set). There are other methodological variations in some of the other papers, which require careful consideration when comparing results, as discussed below.

Our objectives in this paper are to (i) contribute to addressing the problem of how to carry out structured learning in connectionist systems, (ii) propose and evaluate a method for MIL based on neural networks taking into account the studies carried out previously by the researchers in this area, and (iii) note that benchmark comparative results in this area should be taken with caution due to methodological variations. In Section II, we briefly discuss the relevant background work. In Section III, we present the new neural model for MIL. In Section IV, we present the experimental results¹. In Section V, we discuss results in comparison with related work. In Section

¹We are grateful to Rafael V. Borges for many useful discussions and for running the experiments reported in this paper. The second author would like to thank Brazilian research agencies CNPq (Federal), FAPERJ (Rio de Janeiro), and FACEPE (Pernambuco) for their financial support.

VI, we conclude and discuss directions for future work.

II. CONNECTIONIST MULTI-INSTANCE LEARNING

Robust learning is a key aspect of artificial intelligence and, more generally, computing [8]. We argue that the small incremental changes that characterise most of network learning, e.g. gradient descent, seem to be more appropriate than the more direct learning mechanisms of ILP and other symbolic methods. This makes ILP generally more sensitive to noise than network models. On the other hand, ILP is easily interpretable, while networks require the difficult task of knowledge extraction if they are to be explained. The neural-symbolic approach seeks to address both the problem of extraction and the problem of making use of structure in connectionism, by providing algorithms that can establish one-to-one correspondences between symbolic knowledge and connectionism, before and after learning [9], [10].

In what follows, we take inspiration from the neural-symbolic approach to compare the traditional, symbolic approach to multi-instance learning and the connectionist approach based on gradient-descent. Let us start by discussing some of the main work related to multiple instance learning within a neural-network framework. In [11], a method for learning determinate logic programs using neural networks was introduced. It used the Linus system [12] to create a propositionalisation of the logic program which was then fed to the network for learning with standard backpropagation. In [13], two ILP datasets were transformed into multi-instance datasets and the algorithm of [6] was used. Background knowledge was used to create the instances, but not to define the initial structure of the network. The same ideas from [11] were used in [13] to define the set of inputs for the network. In [14], the initial structure of the network was defined by a translation algorithm according to background knowledge given in the form of a function-free (grounded) logic program. In [13], the empirical evaluation included the addition of noise into the datasets. The results showed, in a limited way, that neural networks can be better at dealing with noise than standard ILP. The first-order knowledge-based networks [15] also showed better performance than standard ILP on certain problems.

In [16], knowledge from relational databases was added into the structure of a network creating a deep hierarchy in a feedforward network. Some synchronisation and external memory allocation was then needed to create the training examples for parts of the network to run. An alternative would be to consider recurrent networks, which have been shown useful for sequence learning [17]. In fact, a recurrent network could be used to memorise the instances in a bag and then process the bag classification. The main reason against this idea is the fact that the bag represents a set rather than a sequence, and a recurrent network would give priority to more recent memories, i.e. later in the sequence. To solve this problem, we systematically randomise the presentation of the instances to the network so that the network would learn the appropriate relations in the bag without being biased by the sequence presentation. Whilst this is clearly a possibility, and

in fact a main aspect of the approach presented in this paper, most of the work on connectionist MIL has been focused on feedforward networks only. In this paper we will evaluate the use of recurrent networks in some detail.

We now recall the main algorithms in some detail. In [5], a modified backpropagation for multiple instances was introduced. Function $dmax$ below tries to simulate the MIL problem definition. The modified backpropagation then uses $dmax$ in the output neuron instead of the usual sigmoid activation function. In [5], it is shown that $dmax$ and the real max function are within an error bound given by $\ln(n)/\alpha$, where n is the number of instances in a bag and α is a learning parameter. Differently from max , $dmax$ can be differentiated as shown below. This is used in the error backpropagation. We claim that $dmax$ does not capture the intended meaning of MIL because it tries to maximise the instance value while the value of MIL instances is really unknown or irrelevant.

$$dmax(x_1, \dots, x_n) = \frac{1}{\alpha} \ln\left(\sum_{i=1}^n e^{\alpha x_i}\right)$$

$$|dmax(x_1, \dots, x_n) - \max\{x_1, \dots, x_n\}| < \frac{\ln(n)}{\alpha}$$

$$\frac{\partial}{\partial x_j} dmax(x_1, \dots, x_n) = \frac{e^{\alpha x_j}}{\sum_{i=1}^n e^{\alpha x_i}}$$

In [6], standard backpropagation is used with a modified error function to achieve a similar result as [5]. Below, $b_i = +$ (resp. $b_i = -$) denote that the bag is positive (resp. negative), and o_{ij} is the network's output given instance j of bag i as the network's input. Neurons' activations are in the range $[0, 1]$ and 0.5 is used as threshold. When the network agrees with the bag for a given instance then the intermediary error E_{ij} is assumed to be zero. Otherwise, the error is computed w.r.t. a target of 0.5. We regard this as a conservative approach if compared e.g. with [3]; [6] seems to maximise uncertainty² in the range $[0, 1]$.

$$E_{ij} = \begin{cases} 0 & \text{if } b_i = + \text{ and } o_{ij} \geq 0.5 \\ 0 & \text{if } b_i = - \text{ and } o_{ij} < 0.5 \\ \frac{1}{2}(o_{ij} - 0.5)^2, & \text{otherwise.} \end{cases}$$

The error that is passed on to neuron o_{ij} so that gradient descent can be applied is not E_{ij} , but the error for the bag E_i as described below³. E_i is computed outside the network after all the instances of bag i have been presented to the network and all E_{ij} 's have been calculated and safely stored. In this way, a single error value ends-up being used for the entire set of instances in the bag.

²One could assume that the instance target should be equal to the bag's classification (i.e. zero or 1), but this approach has been shown, at least in the drug development example, to produce too many false positives [1].

³If the bag is positive and some instance satisfies it (i.e. gives error zero) then function min will guarantee that no weight update takes place in the network (i.e. it is as if the network has nothing to learn). If the bag is negative and some instance gives a non-zero error then function max makes sure that some learning (i.e. weight change) will take place. Dually, if the bag is positive and all errors are non-zero then some learning will take place, and if the bag is negative and all errors are zero then no learning needs take place.

$$E_i = \begin{cases} \min\{E_{ij}\} & \text{if } b_i = + \\ \max\{E_{ij}\} & \text{if } b_i = - \end{cases}$$

We should note that [6] uses each instance of a bag as an input to the network, while [5] presents all the instances of each bag all at once in parallel as input to the network. When we ran the same experiments, the result using standard backpropagation, i.e. learning as if the Musk dataset were not a MIL problem, (86.96%) was comparable to [5] (88%) and better than [6] (83.8%). The results using recurrent networks (considering a limited number of network configurations) were better than any feedforward network approach using backpropagation and were comparable to Bayesian and Citation kNN [4], and slightly worse than [1]. This brief analysis indicates to us that (i) there are a number of options in terms of structure and learning algorithm and (ii) the use of recurrent networks should be investigated.

III. STRUCTURED NETWORK LEARNING

We propose *structured network learning* (SNL), an approach where, first, a prototype vector is learned from the set of instances in a bag. Then, this prototype vector is used through supervised learning for the classification of the bag, given the bag’s label. In this way, the instance learning can be seen as a kind of (supervised or unsupervised) pre-processing for a subsequent supervised learning at the level of the bags. The model can be implemented by a recurrent network such that the instances are provided as input, the prototype is learned through the network’s context units, and the bag classification is learned by the network’s output in the usual way.

Each instance of a bag is a feature vector i_i . The set of examples is a set of bags. Each bag b_j is labelled $+$ or $-$, but not each instance. A recurrent neural network \mathcal{N} is used to map each input vector i_i of size q into a prototype vector $p_j^{(t)}$ (in the network’s context units) and a single output neuron (representing the bag’s classification). The size of the prototype vector can vary depending on the application and is a parameter of the system⁴. Let the prototype vector have size m . Thus, our network contains $q + m$ neurons in its input layer and $m + 1$ neurons in its output layer. Given a bag $b_j = \{i_1, i_2, \dots\}$, the target output for the bag is known, call it $bag \in \{-1, 1\}$ and let $bag = 1$ if $b_j = +$ and $bag = -1$ otherwise. Let $o^{(t)}$ denote the actual output of \mathcal{N} given input $\langle i_i^{(t)}, p_j^{(t)} \rangle$ at time t . The training error for \mathcal{N} at time t is calculated in the usual way:

$$e^{(t)} = \frac{1}{2}(o^{(t)} - bag)^2$$

Each instance is randomly selected from the bag. The prototype p_j is set to zero initially. After every instance has been presented, the error is propagated back through the network and the weights, including the weights in the recurrent connections, are adjusted in the usual way. The process is repeated until $e^{(t)}$ is smaller than a predefined small number.

In the SNL algorithm below, the idea is to use standard off-the-shelf methods both in the (higher-level) learning of the

bag classification and in the (lower-level) learning of instances. This is reassuring from a neural-symbolic perspective whereby the goal is to achieve higher-level functionalities within the same efficient model of computation [9].

We argue that [5] and [6] are inadequate in that they either use the same global error for every instance or a very large network that takes the instances accross all the bags as input assuming that they all should have the same arity. SNL uses a small network that takes as input each instance at a time. SNL uses gradient-descent for learning from instances and prototypes. As a result, the network scales better than [5] with the size of the instances. Yet, the network should be able to learn the proportional contribution of each instance to the classification of the entire bag. Differently from [6], a sequence of prototypes with multiple error values will be considered during learning as the network changes its weights. This is done in SNL with the use of a memory and synchronisation, as implemented by the network’s recurrent connections. Finally, SNL uses a bipolar activation function and inputs in $\{-1, 0, 1\}$, where 0 denotes *unknown*, allowing the representation and learning of bags containing feature vectors of different sizes. The network will contain $q + m$ input neurons as defined above, with some neurons having input *zero* for certain bags.

IV. EXPERIMENTAL RESULTS

We have applied SNL to the Musk dataset [1]. In this problem, each molecule is defined by a set of observations, each observation done by measuring the distance between a point and the boundary of the molecule, considering 162 different angles. Also, each molecule will have information about the existence of oxygen atoms. Each molecule is then classified by the Musk strength (as positive or negative). The musk strength of a molecule is considered as positive if, at least, one of the observations indicates so.

Below, we compare results with a range of MIL algorithms. Recurrent networks with 15 and 20 hidden neurons and 3, 6 and 9 context units were evaluated. The table shows the accuracy results and confidence intervals (when available) on the Musk dataset based on 10 runs using 10-fold cross-validation. We have also run the same experiment on feedforward networks with 10, 15, 20, 30 and 40 hidden neurons using standard backpropagation.

The feedforward networks have a much larger number of input neurons and no context units since the entire bag is presented in one go to the network as a single input vector. The feedforward networks with 40 hidden neurons performed considerably better than the results reported so far in the literature for backpropagation (for a network with 125 hidden neurons). The use of this larger number of hidden neurons must have had a negative effect on the network’s generalization. The recurrent networks performed even better than the feedforward networks, achieving a very high performance in some runs.

We believe that the recurrent networks have produced better results because they allow the learning of a bag’s prototype vector in its context units prior to classification. This seems to

⁴In this paper, networks with 3, 6 and 9 context units were evaluated.

	Musk1	Confidence
EM-DD [3]	96.8	(??-??)
Iterated Discrim [1]	92.4	(87-97.8)
Citation kNN [4]	92.4	(??-??)
SNL (20 hidden and 9 context neurons)	90.22	(86.71-93.73)
Bayesian kNN [4]	90.2	(??-??)
SNL (15 hidden and 9 context neurons)	89.13	(83.47-94.79)
Diverse Density [18]	88.9	(??-??)
SNL (20 hidden and 6 context neurons)	89.13	(85.03-93.23)
Multi-instance Neural Networks [5]	88.0	(??-??)
SNL (15 hidden and 3 context neurons)	86.96	(83.30-90.62)
SNL (20 hidden and 3 context neurons)	85.87	(81.97-89.77)
SNL (15 hidden and 6 context neurons)	85.87	(80.32-91.42)
Backpropagation (40 neurons)	86.96	(83.12-90.79)
Backpropagation (10 neurons)	85.87	(81.81-89.93)
Backpropagation (15 neurons)	84.78	(80.76-88.81)
BP-MIP [6]	83.8	(74.6-93)
Backpropagation (20 neurons)	83.7	(79.65-87.74)
Backpropagation (30 neurons)	83.7	(77.49-89.90)
Relational NN, with reshuffling [16]	82	(79-85)
Backpropagation (125 neurons) [5]	75	(??-??)
C4.5	68.5	(40.9-61.3)

produce a more natural modelling of the original MIL problem. The temporal dimension of recurrent networks is used to create an internal representation through the randomized presentation of sequences of bag instances. We hypothesize that this internal representation is responsible for an implicit assignment of labels to the instances of the bags, and is thus a useful MIL representation of context. By randomizing the presentation of the input sequences for learning, the context representations do not become dependent on the most recent sequences and seem to provide a good prototype representation of the set (or bag).

This study has concluded that (i) standard backpropagation can perform MIL to a level of accuracy much higher than previously reported in the literature, and (ii) the use of simple recurrent networks allows a further improvement on accuracy, possibly to very high levels, as a result of a more structured feature learning. The only system showing a performance distinctly better than the recurrent networks of SNL is the EM-DD [3], which does not seem to use cross-validation in the usual way, as already discussed above. The non-standard use of cross-validation by EM-DD precludes a direct, fair comparison with SNL. If we compare only the systems for which confidence intervals have been provided then the SNL’s results are worse only than one of the original MIL algorithms proposed in [1]. We shall discuss this in some detail in the next section. In a nutshell, we believe that further experiments on model selection, considering different numbers of context units, would allow the recurrent networks to produce results comparable to [1], since those have been obtained already in some folds as part of the current validation. We also argue that SNL is more general in what concerns its applicability to different application domains, as discussed below.

V. DISCUSSION

Dietterich et al, in their seminal paper [1] defined the multiple-instance learning (MIL) problem as the task of identifying an attribute or classifying an example given by a set (bag) of instances or alternative feature vectors. In the same paper, the authors propose to perform such task through the use

of geometrical entities called Axis-Parallel hyper Rectangles (APR). Their idea consists of defining an APR in the domain space such that all the positive feature vectors are inside the rectangles. Three different strategies were proposed to learn this APR from examples: The first one, called *all-positive APR*, tries to create the minimum rectangle that contains all the positive instances, the second one, called *elim-kde*, uses the negative instances to “shrink” the APR and therefore reduce the chances of false positives, and the third one, called *iterated-discrim* starts with small APRs, growing them iteratively in order to form better descriptions of the domain.

An alternative approach to solve the problem consists of the Diverse Density (DD) method [18]. The Diverse Density of a point in space is a measurement identifying the union of the positive bags near to this point minus the intersection of the negative bags. Estimating this values for each instance in a bag may allow for the classification of the bag as a whole. The idea was extended through an Estimation-Maximization algorithm (EM-DD) [3], where the estimation step selects a hypothesis consisting of instances from each bag that are most likely to be responsible for the label, and the maximization step uses a gradient-ascent search to find a new hypothesis that maximizes the diverse density. In [4], a lazy-learning approach was proposed to tackle the MIL problem. In a lazy-learning technique, the examples are not used directly to build a model, but are stored for future use in the classification. The approach uses the k-nearest neighbours (kNN) classification algorithm in two different ways. The first (Bayesian-kNN) uses a probabilistic approach to estimate the probability that a bag is positive given their neighbours. The second (citation-kNN) uses not only the neighbouring bags, but also the neighbours of these neighbours.

In our experiments, *iterated-discrim*, *citation-kNN* and *EM-DD* are the only three methods showing better results than SNL’s recurrent networks. Putting EM-DD aside due to the methodological discrepancies already mentioned, we argue that SNL is more general than APR and kNN. With the use of a recurrent network, SNL can learn a prototype, context vector that should, if learning is successful, create a good feature space representation of the multiple instances. With the use of backpropagation and universal-approximator networks, SNL can learn to fit any curve, and not only rectangles, into this feature space. It should, as a result, be more generally applicable than APR to other multi-instance learning domains. Experiments in other domains are ongoing.

As mentioned earlier, neural networks have also been applied to the MIL problem. For the networks that used backpropagation-like algorithms, e.g. [16], the performance reported was poorer than the other symbolic techniques above. However, little detail can be found in the literature about how these networks were trained exactly (e.g. training parameters). Other approaches using neural networks include the BP-MIP [6], where as discussed earlier, the error of each instance is adjusted to try and reflect the behaviour of the entire bag during learning.

Like SNL, in [16], Uwents and Blockeel propose a more general approach than APR or citation-kNN that also uses neural networks, called *relational neural networks* (RNN).

RNNs use different neural networks to represent classes of objects in a kind of unfolding of a recurrent network in time, potentially with the use of different sets of weights. A relational database is used as inspiration to connect the networks so that the output of one network can become the input of another. The resulting network is capable of learning such relations. Notice that the multi-instance learning problem is a special case of the more general relational learning problem. The results using RNNs have been generally worse than SNL, in these limited experiments. The main problem seems to be that dictating a predefined relational structure by building the RNNs as opposed to learning it in the SNL's context layer, may lead to a loss of performance when this structure is not quite the most appropriate from the beginning. We also believe that training performance of single-hidden layer SNL should be generally better than multiple-hidden layer RNNs, as indicated by experiments carried out by others, e.g. [19].

As mentioned above, standard backpropagation was also applied to the Musk testbed using a variety of network set-ups. In [1], an unusually large number of hidden neurons (128) is used, while in [16] only one example from each bag seems to ever be used for training, probably leading to a poor context representation. Our best result using backpropagation is slightly better than two of the SNL set-ups with 3 and 6 context units, indicating that the feedforward approach should, in fact, be a contender for further model evaluation. Perhaps the MIL problem can be seen as a simple learning task with noise, whereby for the negative bags, the labels are all correctly assigned, but for the positive bags, at least one of the labels is known to have been correctly assigned, while the others should be treated as false-positives. Given the natural capacity of neural networks to deal with noisy domains, we argue that they should obtain generally good performance in MIL problems.

We have investigated a number of feedforward and recurrent network architectures to tackle the Musk example, some of which are not reported in this paper. In all of them, including the ones reported here, we have used learning rate of 0.5 and no momentum. As expected, we have found a strong relationship between the number of hidden and context neurons and network performance. Recurrent networks have been used as an alternative to temporal logic learning [20]. In this setting, the MIL problem can be re-defined as the task of learning to classify sequences of instances over time. The representation of bags as sequences allows the more efficient training of each instance in the network, while the use of recurrent links and context units allows the system to learn an internal representation for the bags. We hypothesize that this learned internal representation should be sufficient for the identification of the labels to be assigned to the bags.

Although we propagate instances through time in the network, the goal of MIL is to learn concepts about each bag. Therefore, the values of one bag do not need be considered by the network when learning the next bag. In order to represent the beginning of a new bag, we reset the values in each recurrent link, applying zero to the associated input neurons. The remaining of the process is the same as in a normal

temporal learning task. When analysing the network's results, for the sake of consistency with the results obtained using backpropagation, we have considered all the outputs obtained by the network for each instance of each bag, and have applied the MIL principle, i.e. the bag was labeled as positive by the network if at least one of the instance results is positive. An alternative approach would have been to assume that after the last presentation of the instances of a bag to the network, the network has got internally all the information about the bag and, therefore, use only the recurrent network's output from this last presentation as the network result to validate the bag. This is part of ongoing further experiments using SNL. Finally, it is probably useful reiterating that, in principle, bags do not take sequences into consideration whilst recurrent networks do. The use of different presentation sequences might create a strong bias in the network's results, particularly if the later of the two evaluation methods above were to be used. To deal with this issue, in all of our experiments, we have presented different sequence orders to the networks. Therefore, before putting each sequence through a training step, we have randomized the order of the values in the sequence. This process was repeated each time each sequence was presented. This use of random sequences should reduce the chances of overfitting. In the test phase, the original order was considered as the valid representation of the domain.

VI. CONCLUSION AND FUTURE WORK

When dealing with the multi-instance learning problem, we have proposed to combine instance-level preprocessing and bag-level classification in an integrated system based on recurrent networks. Results indicate that both feedforward and recurrent networks can be considered as worthy contenders in MIL problems. Further work in model selection and comparisons is warranted.

As future work, an extension would be to include the current network's output as part of the context. This should improve accuracy further as it provides more information to the prototype, context units. One may also consider adding specific features to the prototype vector as extra information for learning. For example in [4], instance frequencies and different distance measures were considered. These could easily have been included in the SNL prototype before network learning. Finally, the system can be extended to perform semi-supervised learning [21] in a more modular way, with the prototype-learning using a simple unsupervised learning and the bag classification using standard backpropagation given the prototypes.

At first inspection, structured learning may seem incompatible with connectionism. However, as the problem of MIL indicates, certain relevant domains are structured by nature. It would be very counter-productive not to take advantage of such knowledge when it is available. At the same time, some properties of unstructured connectionism are highly desirable: robustness, fault tolerance, parallelism. The work on the interplay between neural networks and ILP seeks to reconcile symbolic structures and connectionism when tackling the problems of structured learning [10], [19].

Algorithm 1 Structured Network Learning

```

for each bag  $b_j$  in  $\{b_1, \dots, b_n\}$  do
  set prototype  $p_j^{(0)}$  to zero;
  repeat
    select  $i_i^{(t)}$  randomly from  $b_j$ ;
    train network  $N$  with input  $\langle i_i^{(t)}, p_j^{(t)} \rangle$  and target
    output bag;
    calculate error  $e^{(t)}$ ,
    change weights  $\Delta W^{(t)} = -\eta \nabla e^{(t)}$ .
  until  $e^{(t)} < \epsilon$ 
end for

```

REFERENCES

- [1] T. G. Dietterich, R. H. Lathrop, and T. Lozano-Perez, "Solving the multiple instance problem with axis-parallel rectangles," *Artificial Intelligence*, vol. 89, no. 1-2, pp. 31–71, 1997.
- [2] J. Foulds and E. Frank, "A review of multi-instance learning assumptions," *The Knowledge Engineering Review*, vol. 25, pp. 1–25, 2010.
- [3] Q. Zhang and S. A. Goldman, "Em-dd: An improved multiple-instance learning technique," in *In Advances in Neural Information Processing Systems*. MIT Press, 2001, pp. 1073–1080.
- [4] J. Wang, "Solving the multiple-instance problem: A lazy learning approach," in *In Proc. 17th International Conf. on Machine Learning*. Morgan Kaufmann, 2000, pp. 1119–1125.
- [5] J. Ramon and L. D. Raedt, "Multi instance neural networks," in *Proceedings of the ICML-2000 workshop on attribute-value and relational learning*, 2000, pp. 53–60, uRL: http://www.cs.kuleuven.ac.be/cgi-bin-dtai/publ_info.pl?id=31670.
- [6] Z. hua Zhou and M. ling Zhang, "Neural networks for multi-instance learning," In: *Proceedings of the International Conference on Intelligent Information Technology*, Tech. Rep., 2002.
- [7] D. Rumelhart, G. Hinton, and R. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, D. Rumelhart and J. McClelland, Eds. MIT Press, 1986, vol. 1, pp. 318–362.
- [8] L. Valiant, "Knowledge infusion: In pursuit of robustness in artificial intelligence," in *Proceedings of the 28th Conference on Foundations of Software Technology and Theoretical Computer Science*, Bangalore, India, 2008, pp. 415–422.
- [9] A. d'Avila Garcez, K. Broda, and D. Gabbay, *Neural-Symbolic Learning Systems: Foundations and Applications*, ser. Perspectives in Neural Computing. Springer, 2002.
- [10] A. d'Avila Garcez, L. Lamb, and D. Gabbay, *Neural-Symbolic Cognitive Reasoning*, ser. Cognitive Technologies. Springer, 2009.
- [11] R. Basilio, G. Zaverucha, and A. d'Avila Garcez, "Inducing relational concepts with neural networks via the LINUS system," in *Proceedings of the Fifth International Conference on Neural Information Processing ICONIP'98*, 1998, pp. 1507–1510.
- [12] N. Lavrac, S. Dzeroski, and M. Grobelnik, "Experiments in learning nonrecursive definitions of relations with LINUS," Josef Stefan Institute, Yugoslavia, Tech. Rep., 1990.
- [13] B. Kijssirikul and T. Lerdlamnaoachai, "First-order logical neural networks," *Int. J. Hybrid Intell. Syst.*, vol. 2, no. 4, pp. 253–267, 2005.
- [14] A. d'Avila Garcez and G. Zaverucha, "The connectionist inductive learning and logic programming system," *Applied Intelligence Journal, Special Issue on Neural Networks and Structured Knowledge*, vol. 11, no. 1, pp. 59–77, 1999.
- [15] R. Basilio, G. Zaverucha, and V. Barbosa, "Learning logic programs with neural networks," in *Inductive Logic Programming*. Springer LNAI 2157, 2001, pp. 15–26.
- [16] W. Uwents and H. Blockeel, "Classifying relational data with neural networks," in *Proceedings of 15th International Conference on Inductive Logic Programming*, ser. Lecture Notes in Computer Science, S. Kramer and B. Pfahringer, Eds., vol. 3625. Bonn, Germany: Springer, 2005, pp. 384–396.
- [17] M. Jordan, "Attractor dynamics and parallelisms in a connectionist sequential machine," in *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, 1986, pp. 531–546.
- [18] O. Maron and T. Lozano-Perez, "A framework for multiple-instance learning," in *Neural Information Processing Systems 10*. Cambridge, MA: MIT Press, 1997.
- [19] W. Uwents, G. Monfardini, H. Blockeel, M. Gori, and F. Scarselli, "Neural networks for relational learning: an experimental comparison," *Machine Learning*, vol. 82, no. 3, pp. 315–349, 2011.
- [20] L. Lamb, R. Borges, and A. d'Avila Garcez, "A connectionist cognitive model for temporal synchronisation and learning," in *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence AAAI 2007*. AAAI Press, 2007, pp. 827–832.
- [21] Z. Zhou and J. Xu, "On the relation between multi-instance learning and semi-supervised learning," in *ICML '07: Proceedings of the 24th international conference on Machine learning*. New York, NY, USA: ACM, 2007, pp. 1167–1174.