Relevance Effect: Exploiting Bayesian Networks to Improve Supervised Learning

Ardavan S. Nobandegani[†]
ECE Department
McGill University
Montreal, QC H3A 0E9
ardavan.salehinobandegani@mail.mcgill.ca

Jad Kabbara[†]
School of Computer Science
McGill University
Montreal, QC H3A 0E9
jad@cs.mcgill.ca

Ioannis N. Psaromiligkos
ECE Department
McGill University
Montreal, QC H3A 0E9
yannis@ece.mcgill.ca

[†] Co-primary authors

***Abstract*—Deductive logic and its variants enjoy the common property of monotonicity. For tasks such as inductive reasoning and belief revision, this was eventually deemed a serious flaw, prompting attempts to construct non-monotonic versions of logic. With the introduction of the idea of probabilistic reasoning to AI, particularly with the advent of Bayesian networks (BNs), the aforementioned monotonicity was no longer an issue: Probability is inherently non-monotonic. In this work, we introduce the notion of relevance effect which bears on exploiting BNs to generate realizations of relevant variables to be used for potentially improving the performance of a learning model on a supervised classification task. We explore the potential of using the relevance effect in the context of Deep Belief Networks (DBNs) with a focus on relational domains. We show that although the idea is at odds with the non-monotonicity of probabilistic reasoning, we attain an improvement in learning performance in different simulations on both synthetic and real-world scenarios. The observation that adopting this notion has improved the performance of a powerful model like DBNs hints to its potential to be practiced so as to enhance the performance of supervised learning methods in general. We furthermore highlight the connections as well as the implications of our work to the psychology literature.**

I. Introduction

In the past decade, server farm databases have exploded with human photos. In September 2013, Facebook reported storing more than 250 billion photos on its servers with users uploading an average of 14.5 million photos per day [1], providing massive training resources for image recognition applications. At the same time, humans are increasingly relying on intelligent systems such as smart phones to carry out their everyday tasks. As a result, speech recognition, which has arguably seen little use on personal computers, is finding a burgeoning base of users. Such trends have inevitably pushed for scalable and highly accurate machine learning algorithms. The past few years have seen a surge in deep learning research, fueled by the industry's interest in using deep neural networks (DNNs) for applications such as image recognition, speech recognition and natural language processing.

While the research community in deep learning has focused on optimizing deep learning training techniques and introducing improvements for specific applications, it has been noted that a natural next step for deep learning is to capitalize on the underlying relational structure of the domains within which they are employed ([2], [3]). Bayesian networks (BNs) can be viewed as a *universal* graphical language through which the underlying dependency structure governing *any* domain of interest can be expressed (yet, not necessarily a perfect map or, in short, P-map [4]). In this light, BNs can be plausibly invoked to represent such a relational structure, with nodes representing the variables of the domain and edges modeling their immediate (potentially probabilistic) interactions which can be mathematically encoded by some conditional probability distribution (CPD).[1] Furthemore, BNs lie at the heart of areas such as statistical relational learning where, for example, Probabilistic Relational Models and Statistical Relational Models [6] adopt BNs as their building blocks.

Consider a classification task involving input variables $\mathbf{e}_1, \ldots, \mathbf{e}_n$ and output variable $\mathbf{o}$. Assume further that these variables are part of a domain whose underlying dependency structure is modeled by some BN $\mathcal{B}$. In this paper, we explore how this dependency structure, modeled by $\mathcal{B}$, can be exploited to generate novel information that complements the learning process involving the input variables $\mathbf{e}_1, \ldots, \mathbf{e}_n$ such that the classification performance on output variable $\mathbf{o}$ improves. The key contribution of the paper lies in introducing the notion of "relevance effect" which goes against the non-monotonicity of probability as explained in the next section. We explore the potential behind using the relevance effect in the context of Deep Belief Networks (DBNs) with a focus on relational domains, and show that this leads to an improvement in learning performance in both synthetic and real-world scenarios. The observation that adopting this notion has improved the performance of a powerful model such as DBNs hints to its potential to be practiced for enhancing the performance of supervised learning methods, in general.

II. On Rules and The Relevance Effect

A. Rules

We begin by defining a key term that will be used throughout the paper. Given variables $\mathbf{e}_1, \ldots, \mathbf{e}_n$, a *rule* refers to a

[1]BNs furthermore due to their causal interpretation are often preferred to its counterparts (e.g., Markov Nets [4] or Chain Graphs [5]), particularly, when the domain of study enjoys causal structures.

mapping that acts on $\mathbf{e}_1, \ldots, \mathbf{e}_n$. This rule can be of a deterministic or probabilistic nature as will be explained in Section II-B. The choice of the rule is guided by the structure of the BN that models the dependency structure of the domain and includes the said variables. The terms “rule” or “relational rule” are henceforth used interchangeably. The term “relational” is due to the fact that a BN represents how variables interact with each other—i.e., reflects the “relations” between those variables—and, in essence, captures the relational structure of the domain. Rules can be of two types: probabilistic or deterministic. Probabilistic rules are in the form of a CPD allowing one to generate (through sampling), conditioned on some variables, an arbitrary number of realizations for the rule’s output. On the other hand, deterministic rules, for a fixed realization of the input, lead to a single realization for the rule’s output.

B. Relevance Effect

Consider again the setting wherein some BN $\mathcal{B}$ is modeling the underlying dependency structure of a domain of interest. Let $\mathbf{e}_1, \ldots, \mathbf{e}_n$ be the input variables and $\mathbf{o}$ the output variable for a given supervised classification task. Now assume that, due to some side information in the form of a CPD, we are able to generate (simply through drawing samples from the CPD) realizations for some variable $\mathbf{s}$ which, according to the structure of $\mathcal{B}$ and the concept of d-separation [7], is relevant to the output variable $\mathbf{o}$. Then one can plausibly argue that if, to each input vector, we append its corresponding sample generated for $\mathbf{s}$, then the learning performance should improve based on the simple argument that employing more relevant information cannot hurt. There is nothing surprising about this—assuming this is done the “right” way. Here, by “right” way, we refer to the fact that having *observed* the values for $\mathbf{e}_1, \ldots, \mathbf{e}_n$ and $\mathbf{o}$ in the “training set” dictates the following: To correctly generate (probabilistically) samples of $\mathbf{s}$, one must draw samples from the CPD $\mathbb{P}(\mathbf{s}|\mathbf{e}_1, \ldots, \mathbf{e}_n, \mathbf{o})$. In other words, in generating samples of $\mathbf{s}$, one cannot, in principle, ignore the fact that the values for $\mathbf{e}_1, \ldots, \mathbf{e}_n$ and $\mathbf{o}$ are *all* observed.

Now, assume that we intend to adopt the “wrong” way, that is, to purposefully ignore the fact that the values for some of the variables $\mathbf{e}_1, \ldots, \mathbf{e}_n$ and $\mathbf{o}$ are indeed observed and, say, to use $\mathbb{P}(\mathbf{s}|\mathbf{e}_1, \ldots, \mathbf{e}_n)$ or even $\mathbb{P}(\mathbf{s}|\mathbf{e}_1)$ instead of $\mathbb{P}(\mathbf{s}|\mathbf{e}_1, \ldots, \mathbf{e}_n, \mathbf{o})$. Doing so goes against the non-monotonicity of probability and, in that respect, is *systematically* wrong. Simply put, the non-monotonicity of probability implies the following: When the value of, say, $\mathbf{a}$ is observed and we are interested in knowing how $\mathbf{b}$ behaves probabilistically, we must adopt $\mathbb{P}(\mathbf{b}|\mathbf{a})$; that is, we cannot, *in principle*, ignore the fact that the state of $\mathbf{a}$ is observed. Non-monotonicity of probability also echoes in the following statement: Let A, B, C be three events, then, $\mathbb{P}(A|B), \mathbb{P}(A|C)$, and $\mathbb{P}(A|B, C)$ can be arbitrarily specified [8]. Furthermore, non-monotonicity of probability is nicely captured in Pearl’s well-known “explaining away” phenomenon [9].

Simulating learning scenarios where we adopted the “wrong” way of incorporating side information into the learning process led to a key observation which we term the *relevance effect*. In simple terms, this effect can be described as follows: *Finding the value of some relevant variable(s), even in the wrong way as stated above, and incorporating it into the training process could improve the learning performance.* Indeed, in different simulation scenarios where we incorporate side information as described earlier, we show consistent improvement over the conventional methodology in which the learning algorithm is solely trained on the inputs and the output variable (i.e., without any side information). Interestingly, the key idea captured by the relevance effect is the interplay between (i) the relevance of a variable (e.g., $\mathbf{s}$) to the output variable $\mathbf{o}$ of a learning task (i.e., the extent to which two variables are correlated), and (ii) how accurately the value for a relevant variable (e.g., $\mathbf{s}$) is derived. This key interplay is simply overlooked in the literature.

At a high level, the relevance effect can be related to the notion of probably approximately correct (PAC) learning in computational learning theory. However, PAC learning is never concerned about finding the solution in the “wrong” way discussed above. Instead, it is purely concerned about the computational efficiency of learning. What ties it, however, to the notion of relevance effect is the key idea that one just needs to be correct “approximately” for a “good” portion of time in terms of generating $\mathbf{s}$ to see an improvement in learning.

C. The Relevance Effect in the Context of Supervised Learning

Applying the relevance effect in the context of deep learning models highlights its potential to yield improvement for classification tasks in relational domains. There are two rationales behind the choice of DBNs: (i) Deep neural networks, and DBNs as a variant thereof, have repeatedly yielded state-of-the-art performance in different applications and it is worth showing that adopting the introduced notion of relevance effect has the potential to improve the performance achievable by such powerful models, and (ii) to promote the idea of taking advantage of the relevance effect in the deep learning practice. Accordingly, we present the RDNN model (R standing for relevance) in which, at the first layer, rules are applied to the input vector, with the result being appended to the (same) input vectors and then the new vectors being fed to the higher standard DBN layers. This corresponds to increasing the expressive power of the DBN in an appropriate, relevant and controlled fashion by introducing a useful *inductive bias* into the learning process. The use of relational rules in this manner potentially allows for better feature extraction and therefore improved classification performance.

The idea of relevance effect has important implications for supervised learning. It allows one to take advantage of *seemingly* “wrong” side information which, based on the conventional methodology, should not be beneficial and, more importantly, is perceived to be systematically wrong to be used for the supervised learning task at hand. In large domains comprised of numerous variables and modeled by BNs, scenarios may arise where we know CPDs associated to only a small number of edges. In such scenarios, the introduced notion of relevance effect would be particularly useful in enabling us to generate samples of "some" variable(s) in the "wrong" way, which we then append to the corresponding input vector and feed the result into the learning algorithm to potentially gain learning improvement for the classification task of interest.

III. On the Nature of Rules and Relations

In this section, we elaborate on the nature of relational rules and formalize this concept in light of BNs [7]. BNs can be viewed as a universal graphical language through which the dependency structure governing *any* domain of interest can be expressed—not necessarily a perfect map (a.k.a. P-map) however [4]. In this context, nodes in a BN represent the variables/attributes of the domain and an edge signifies a dependency or interaction between two nodes. A relational rule simply then refers to the functional form of the interaction taking place between any two connected nodes. With the BNs modeling the underlying dependency structure of domains, a rule can be viewed as a mapping which enables us to generate, based on a subset of observed variables and thereby ignoring the non-monotonicity of probability, realizations of some variable(s) which are deemed relevant to the output variable according to the notion of d-separation. The relevance effect comes into play as it suggests that finding the value of some relevant variable, even in the "wrong" way as stated above, and incorporating it into the training process could improve the learning performance. We begin with presenting a simple yet informative motivating example.

A. Motivating Example

Consider the simple BN depicted in Figure 1 and the task of deciding on the state of Random Variable (RV) $\mathbf{z}$ (output variable) given the state of RV $\mathbf{x}$ (input variable). In this setting, the edge emanating from $\mathbf{x}$ to $\mathbf{y}$ represents the conditional probability distribution $\mathbb{P}(\mathbf{y}|\mathbf{x})$. We assume that only the parametrization of the edge between $\mathbf{x}$ and $\mathbf{y}$ (i.e., the CPD corresponding to that edge) is known and that of the other (dash-dotted) edges are unknown. According to the graph-theoretic notion of d-separation [9], variable $\mathbf{y}$ is deemed relevant to the output variable $\mathbf{z}$ given the input variable $\mathbf{x}$, i.e., $(\mathbf{z} \not\perp \mathbf{y}|\mathbf{x})$, where $(\mathbf{a} \not\perp \mathbf{b}|\mathbf{c})$ denotes that, given $\mathbf{c}$, RVs $\mathbf{a}, \mathbf{b}$ are dependent. Therefore, the CPD $\mathbb{P}(\mathbf{y}|\mathbf{x})$ corresponding to the edge pointing from $\mathbf{x}$ to $\mathbf{y}$, plays the role of a rule which enables one to generate (i.e., to sample) the relevant variable $\mathbf{y}$ conditioned on $\mathbf{x}$. It is crucial to notice that, for the training set, RV $\mathbf{z}$ is observed along with the input variable $\mathbf{x}$ and, to accurately generate $\mathbf{y}$, one has to employ $\mathbb{P}(\mathbf{y}|\mathbf{x}, \mathbf{z})$ and not $\mathbb{P}(\mathbf{y}|\mathbf{x})$. This is where the key idea of relevance effect comes into play. That is, the probabilistic rule $\mathbb{P}(\mathbf{y}|\mathbf{x})$ could be used to generate a "wrong" but "accurate enough" version of $\mathbf{y}$ which could improve the classification task on $\mathbf{z}$.

B. Rules in the Context of Complex Networks

Oftentimes, the number of variables/attributes in the domain of interest is enormous and a lot of inter-variable interactions take place altogether described by a large and complex

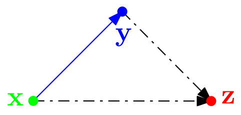


Fig. 1. Sample Case: Input and output variables for the classification task of interest are depicted in green and red, respectively. Variable $\mathbf{x}$ is observed and the task is to decide on the state of $\mathbf{z}$. The CPD corresponding to the blue edge is known and for others is unknown.

network, e.g., protein domain, cell and social networks to mention a few. Let us model the underlying dependency structure governing the attributes/variables of the domain by a BN. We intend here to generalize the idea presented in the motivating example to the setting of complex networks as depicted in Figure 2. The main complication often encountered in these complex domains is that the functional form of many of the interactions (represented pictorially by edges in the graph) is either partially or fully unknown. Yet, despite such an incomplete knowledge of the domain, we need to give adequate answers to the posed inference problem. The idea is then to take advantage of some of the known edges and use them as rules to improve the performance on the task at hand. It is worth noting that the variables that should be appended to the training set are those which, according to the graph-theoretic notion of d-separation, conditioned on the observed variables, are deemed relevant to the output variable of the task at hand.

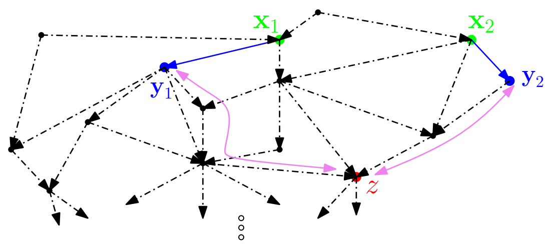


Fig. 2. Sparsely-known large/complex network: Input variables $\mathbf{x}_1$, $\mathbf{x}_2$ (observed) are depicted in green; the output variable for the classification task, $\mathbf{z}$, is depicted in red. Due to the existence of unblocked trails (depicted in magenta) between the to-be-appended variables (i.e., $\mathbf{y}_1$, $\mathbf{y}_2$) and the output variable for the classification (i.e., $\mathbf{z}$), variables $\mathbf{y}_1$, $\mathbf{y}_2$ are deemed relevant to the output variable $\mathbf{z}$ conditioned on the input variables $\mathbf{x}_1$, $\mathbf{x}_2$.

C. Generalization of the Notions of Rules and Relations

We now generalize the notion of rule as follows. In a broader sense, a relational rule simply refers to an algorithmic form of a mapping from a number of variables in the domain (input variables) to some other variable(s) in the domain (intermediate variable(s)). By algorithmic mapping, we mean a mapping that is realized by some algorithm (e.g., one implementing an estimator/regressor) which supersedes the need to specify a mathematical expression for the rule. A rule relevant to the classification task at hand, therefore, enables one to get an *estimate* of some intermediate variable(s) which are deemed relevant to the output variable of the classification task at hand according to the notion of d-separability, in terms of a subset of variables in the domain whose states are known (see Figure 3 for an example). The mapping could be of a deterministic or probabilistic nature. In case of a probabilistic mapping, the act of deriving the intermediate variables boils down to drawing samples from the probabilistic mapping. To our knowledge, this broad interpretation of relational rules as deterministic or probabilistic mappings and their use to generate features relevant to the learning task (due to the notion of d-separation) is novel.

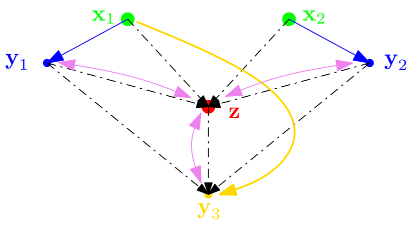


Fig. 3. Generalized interpretation of rules: RVs $\mathbf{x}_1$, $\mathbf{x}_2$ are observed and the classification task is to decide on the state of $\mathbf{z}$. Only the CPDs associated with the blue edges are assumed to be known. The link depicted in gold denotes the existence of a mapping from $\mathbf{x}_1$ to $\mathbf{y}_3$ which enables one to estimate $\mathbf{y}_3$ given $\mathbf{x}_1$. Given that $\mathbf{x}_1$, $\mathbf{x}_2$ are observed, RVs $\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3$ are all deemed relevant to the output variable $\mathbf{z}$ due to the existence of active trails depicted in magenta.

IV. The Relevance Effect in Deep Learning

In this section, we present the framework which we utilize to validate the potential of the relevance effect in improving the learning performance of supervised learning methods. The deep learning model, RDNN, extends DNNs by incorporating relational information into the learning process to promote better feature extraction.

In the first layer of RDNN, the input data is fed into a *relational layer* where relational rules, as introduced in Section III, operate on the input data. This layer may consist of one or more relational rules each providing specific relational information about the input data. A rule can have a discrete or continuous output and it may operate on a subset or all of the components of each input vector. The resulting side information is then appended to the corresponding input vector. The output of the relational layer becomes then the input of the next layer — the first layer of the DNN. Figure 4 shows the RDNN model with the first layer being the relational layer.

With the additional relational information obtained in the first layer, a DNN is now potentially able to extract richer and more meaningful features. The output of the first layer is thus fed to a DBN which learns a hierarchy of multiple layers of representative features using Restricted Boltzmann Machines (RBMs). First, RDNN is pre-trained in a greedy layer-by-layer fashion: Visible stochastic inputs are connected to hidden stochastic feature detectors using symmetrically weighted connections, and learning is done with Contrastive Divergence [10]. The output of each layer becomes the input of the next layer of RBMs, and so on. After the pre-training phase, the weights in RDNN are initialized and the whole

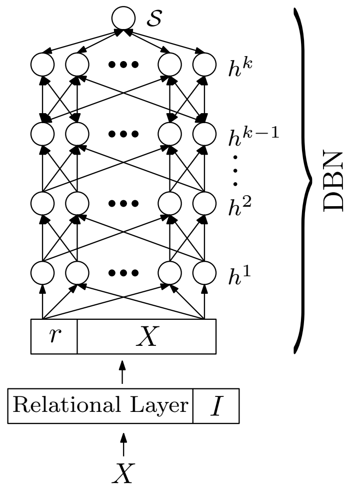
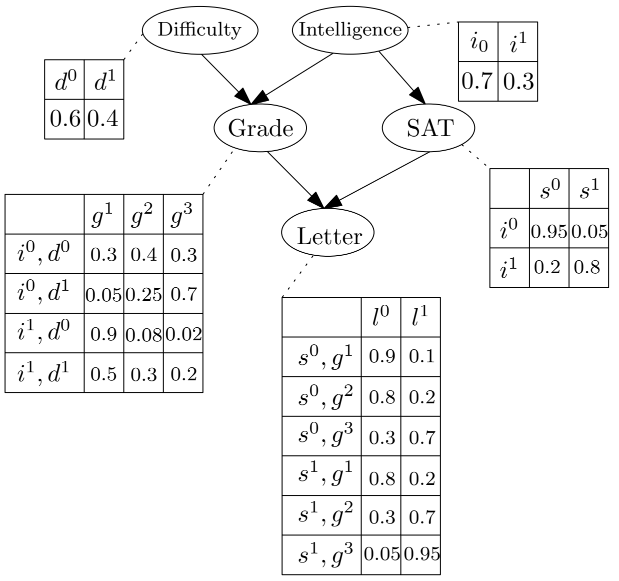


Fig. 4. RDNN: Input vector X is fed to a relational layer where relational rules are applied on the input vectors. The resultant vector r is then appended to X, and this newly-formed vector becomes the input to the standard DBN. Symbol I indicates identity operator which leaves its input intact. Also, h^i denotes the i-th hidden layer of the employed DBN. $\mathcal{S}$ is the softmax unit responsible for classification. Some of the links are omitted to avoid cluttering the picture.

network is fine-tuned using back-propagation. At the network's highest level, the learned features from the previous layer are used to solve classification problems using a classifier. We use a softmax unit as our choice of classifier, however, one could very well use any other classifier, e.g., SVM.

V. Experiments

In this section, we present empirical validation that shows that RDNNs can outperform standard DNNs in classification problems in domains of relational nature. In all simulations, RDNN is compared against a standard DNN based on Hinton and Salakhutdinov's DBN implementation [11] (henceforth simply referred to as DBN). The following apply to all of the experiments: Each dataset comprises a training set, a validation set, and a test set. The rows of the data matrix correspond to samples; the columns correspond to features. All data values are normalized to the $[0, 1]$ interval (based on each column vector of the data matrix). Both algorithms use the same number of hidden layers (specified later in each experiment) and a softmax output unit to perform the classification. The hyperparameters for DBN were optimized to achieve the best performance on the validation set. We set for RDNN the same hyperparameters that were used for DBN unless stated otherwise. Following hyperparameter optimization, both algorithms use an initial momentum of 0.5 and a final momentum of 0.9. For the learning in the RBMs, 1-step constrastive divergence (CD-1) is used. Each hidden layer was pre-trained for 50 epochs, i.e., passes through the entire training set, while for the supervised fine-tuning, the number of epochs was fixed to that number that gives the best performance (in terms to the number of misclassifications) on the validation set. Thus, the performance of each algorithm is measured on the test set at that specified number of epochs.



d^0	d^1
0.6	0.4

i_0	i^1
0.7	0.3

	g^1	g^2	g^3
i^0, d^0	0.3	0.4	0.3
i^0, d^1	0.05	0.25	0.7
i^1, d^0	0.9	0.08	0.02
i^1, d^1	0.5	0.3	0.2

	s^0	s^1
i^0	0.95	0.05
i^1	0.2	0.8

	l^0	l^1
s^0, g^1	0.9	0.1
s^0, g^2	0.8	0.2
s^0, g^3	0.3	0.7
s^1, g^1	0.8	0.2
s^1, g^2	0.3	0.7
s^1, g^3	0.05	0.95

Fig. 5. Student BN.

A. Student Bayesian Network

This experiment is based on a modified version of the Student Bayesian Network relational graph provided in [4] (Fig. 3.4, p. 53). Given two input variables representing the *Intelligence* of a student registered in a course having a certain *Difficulty*, the classification task consists in predicting whether the recommendation letter the student will receive from the course instructor is *strong* or *weak*. The same graph structure and probabilities as in [4] are used, except that we connect the two variables *SAT* and *Letter* and the following CPD is employed: $\mathbb{P}(l^0|g^1, s^0) = 0.9, \mathbb{P}(l^0|g^1, s^1) = 0.8, \mathbb{P}(l^0|g^2, s^0) = 0.8, \mathbb{P}(l^0|g^2, s^1) = 0.3, \mathbb{P}(l^0|g^3, s^0) = 0.3$, and $\mathbb{P}(l^0|g^3, s^1) = 0.05$. The BN is shown in Figure 5. We generate 9000 samples according to the given BN, with 5000 samples being used as a training set, 2000 as a validation set and 2000 as a test set. To provide empirical verification for the idea proposed in Section III, we use the CPD corresponding to the edge connecting RV *SAT* to *Intelligence* as the (non-deterministic) relational rule to be used in the relational layer of RDNN.

Two hidden layers of RBMs were used, each with 5 logistic units. Following hyperparameter optimization, a learning rate of 0.0001 is used for the weights, biases of visible units and biases of hidden units, weight cost of 0.00001 and mini-batch size of 20. Due to the probabilistic nature of the rule used in this experiment, we repeat the RDNN simulation 15 times (each time appending the result of sampling from the probabilistic rule to each training sample) and present in Table 1 the average number of misclassifications along with the error margin (standard deviation). DBN misclassified 797 instances of the test set versus 737 misclassifications by RDNN. This shows that incorporating relational rules in this learning task led to a 7.53% performance improvement. The significance of this result stems from the fact that an improvement was noted despite using the "wrong" CPD, namely $\mathbb{P}(\textit{SAT}|\textit{Intelligence})$, as the rule whereas, according to the dependency structure of the underlying BN and the notion of d-separation, we should have used $\mathbb{P}(\textit{SAT}|\textit{Intelligence, Difficulty, Letter})$, that is, we should have conditioned on all observed variables.

B. 10-Node Bayesian Network

This experiment is based on a BN where each node (representing a variable $\mathbf{x}_i$ with i being the index of the node) depends on its two immediate predecessor nodes. For the first two variables we have: $\mathbf{x}_1 \sim \mathcal{N}(0,1)$, and, with probability p, $\mathbf{x}_2 \sim \mathcal{U}(0,1)$ and $\mathbf{x}_2 \sim \mathcal{U}(-1,0)$ otherwise. The subsequent variables are obtained as follows:

$$(\mathbf{x}_{i+1}|\mathbf{x}_{i-1}, \mathbf{x}_i) \sim \begin{cases} \mathcal{U}(0,1) & \text{if} \quad \psi(\mathbf{x}_{i+1}) > \gamma, \\ \mathcal{U}(-1,0) & \text{if} \quad \psi(\mathbf{x}_{i+1}) < \gamma, \end{cases}$$

where $\psi(\mathbf{x}_{i+1}) :\triangleq \frac{1}{2}(\mathbf{x}_{i-1} + \mathbf{x}_i)$, for some real-valued parameter γ.

In this experiment, we simulate a BN having 10 nodes representing variables $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{10}$. To explore the idea of learning despite merely having partial knowledge of the dependency structure of a domain (as discussed in Section III-B), we assume only RVs $\mathbf{x}_4$ and $\mathbf{x}_5$ are observed (therefore comprising the features of the input vectors). We employ the CPD $\mathbb{P}(\mathbf{x}_6|\mathbf{x}_4, \mathbf{x}_5)$ as the probabilistic rule to be employed in the relational layer of the RDNN. Given this partial knowledge of the network, the classification task consists in predicting whether the 10th variable, $\mathbf{x}_{10}$, is larger than 0. The simulation parameters are set as follows: $\gamma = 0.1$ and $p = 0.4$. We generate 12000 samples with 8000 samples being used as a training set, 2000 as a validation set and 2000 as a test set.

Three hidden layers of RBMs were used, each with 10 logistic units. Following hyperparameter optimization, a learning rate of 0.001 is used for the weights, biases of visible units and biases of hidden units, weight cost of 0.01 and mini-batch size of 50. Similarly to the last simulation, we repeat the RDNN simulation 15 times and present in Table 1 the average number of misclassifications along with the error margin (standard deviation). DBN misclassified 53 instances of the test set versus 41.33 misclassifications by RDNN. This shows that incorporating relational rules in this learning task led to a 22% performance improvement. The significance of this result stems from the fact that an improvement was noted despite using the "wrong" CPD, namely $\mathbb{P}(\mathbf{x}_6|\mathbf{x}_4, \mathbf{x}_5)$, as the rule whereas, according to the dependency structure of the underlying BN and the notion d-separation, we should have used $\mathbb{P}(\mathbf{x}_6|\mathbf{x}_4, \mathbf{x}_5, \mathbf{x}_{10})$, that is, we should have conditioned on all observed variables.

C. Metabolic Pathways Relational Network

This experiment is based on the KEGG Metabolic Relation Network Dataset [12] representing metabolic pathways that model molecular interactions in the human metabolism. The classification task consists in predicting whether enzymes/genes are interacting with more than 3 other neighbors (i.e., whether the neighborhood connectivity is larger than 3).

	RDNN	DBN
Student BN	$\frac{737\pm3.51}{2000}$ **(36.85%)**	$\frac{797}{2000}$ (39.85%)
10-Node BN	$\frac{41.33\pm1.80}{2000}$ **(2.07%)**	$\frac{53}{2000}$ (2.65%)
Metabolic Network	$\frac{231}{8000}$ **(2.89%)**	$\frac{258}{8000}$ (3.23%)

Table 1. Performance of RDNN and DBN in terms of number of misclassifications. The notation is as follows: $\frac{\text{misclassifications}}{\text{size of test size}}$ (Percentage).

Two relational rules are used in the relational layer of RDNN. The first represents the clustering coefficient of enzymes/genes which reflects how much the enzymes/genes tend to form tightly knit groups indicated by a relatively high density of ties, and is given by:

$$c_i = \frac{|\{e_{jk} : v_j, v_k \in N_i, e_{jk} \in E\}|}{k_i(k_i - 1)},$$

where c_i is the clustering coefficient of enzyme/gene v_i, e_{jk} represents a pathway between enzymes/genes v_j and v_k, and k_i is the number of neighbors of enzyme/gene v_i. The second rule represents the betweenness centrality of an enzyme/gene which reflects how central an enzyme/gene is in the network of metabolic pathways, and is given by:

$$d_i = \sum_{i \neq j \neq k} \frac{\sigma_{jk}(i)}{\sigma_{jk}},$$

where d_i is the betweenness centrality of enzyme/gene v_i, σ_{jk} is the total number of shortest paths from node j to node k and $\sigma_{jk}(i)$ is the number of those paths that pass through i. In this experiment, we explore the idea presented in Section III-C. The values of the rules are obtained through least-squares regression. The input to the least-squares regressor consists of only the first 10 (out of 18) features of only 10% (approximately) of the number of samples. We note that these input features or samples were not chosen in any particular way to optimize the least-squares estimate.

Three hidden layers of RBMs were used, each with 50 logistic units. Following hyperparameter optimization, a learning rate of 0.1 is used for the weights, biases of visible units and biases of hidden units, weight cost of 0.1 and mini-batch size of 50. Results for this classification task are shown in Table 1. DBN misclassified 258 instances of the test set versus 231 misclassifications by RDNN. This shows that incorporating relational rules in this learning task led to a 10.47% performance improvement. Interestingly, training the linear LS regressor only on 10% of the training set and about half of the features still enabled the RDNN to outperform DBN by a good margin. The significance of the above observation is three fold: First the output, say $\mathbf{v}$, of the LS regressor (used to derive the values of the rules) is related by a CPD to each realization of the input vector, say $\mathbf{u}$. Adopting LS regression should be perceived, at best, as a single-value approximation for the target variables $\mathbf{v}$ (as opposed to a distribution). Second, neglecting the fact that almost half of the features were observed in deriving the rules using LS regression is systematically wrong based on the notion of non-monotonicity discussed earlier in the paper. Third, the fact that the LS regressor is trained on merely a small fraction of the training set highlights yet another level of approximation.

VI. On the Connection of Rules to Psychology

The idea of relying on prior knowledge and benefiting from it when learning a new task has had a long history in the studies on human cognition [13]. Capacities like one-shot learning [14] and learning to learn [15] signify the crucial role a priori knowledge plays in acquiring new concepts. Translated into our proposed framework, this a priori knowledge available to the reasoner is captured by the notion of relational rule which, by being applied to the inputs and producing an output(s)—a process analogous to the unraveling of a new task in light of the already available knowledge—will improve learning.

Knowledge-Based Cascade-Correlation (KBCC) [16] is a self-organized neural network scheme which in spirit follows the theme of RDNN (Section IV), at Marr's implementational level [17], by recruiting previously-learned neural nets in the self-construction process of devising a new neural network for solving the posed task. Translated into our framework, the to-be-recruited previously-learned neural nets play the role of the abstract rules—symbolizing the possession of some relevant prior knowledge—which we are proposing to be employed in the learning process as discussed in Section IV. However, in recruitment phases, KBCC has to consider *all* the available previously-learned neural nets, as a pool of candidates, to check whether recruiting any of them may lead to a performance improvement [16]; hence KBCC has to struggle with the well-known *exhaustive search* problem [18]—this is analogous to performing exhaustive search in the knowledge base in expert systems. The introduced notion of relevance effect potentially allows for significantly reducing the number of candidates that should be considered by KBCC in recruitment phases (also referred to as *input phases*), thereby alleviating the exhaustive search problem that KBCC has to deal with.

VII. Conclusion and Future Work

In this paper, we introduced the relevance effect which, in simple terms, suggests the following: Finding the value of some relevant variable(s), even in the "wrong" way and incorporating it into the training process could improve the learning performance. Moreover, we formalized the notion of relational rule, generalized it, and discussed how the relevance of a rule to the classification task should be verified through the concept of d-separation and the BN modeling the dependency structure governing the domain at hand. To explore the potential of the relevance effect, we presented a deep learning model, RDNN, wherein relevant (deterministic or probabilistic) relational rules are first applied and then subsequently appended to every input vector; this extended input vector is ultimately fed to a DBN. Appending the relevant relational information to each input vector introduces a useful inductive bias into the learning process of the DBN, thereby leading to an improved classification performance. Through simulations, we provided empirical validation that supports the proposed relevance effect.

While in this work we were interested in investigating the potential of the relevance effect in the context of DBNs, it is worth noting that our method can be generalized to other deep learning methods (e.g., Convolutional Neural Networks [19], Deep Boltzmann Machines [20], etc.) as well as learning techniques beyond those falling under the deep learning paradigm.

Although in Section II-B we provided one possible explanation as to why the relevance effect might lead to an improvement in learning, there are two important factors that may play a role in the emergence as well as the efficacy of the relevance effect: (1) the size of the training set (small vs large), and (2) the choice of the learner (e.g., DBN or SVM). Future work should investigate the significance of these factors for the success of the relevance effect in improving learning.

Also, for DNNs specifically, it would be interesting to investigate alternative designs for injecting relational information into the learning process of DNNs. For example, while in our model the first layer is the relational layer, that is, the relational information is added to the network at the beginning, other designs may add the relational side information at a later stage.

Acknowledgment

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC) under grant RGPIN 262017.

References

[1] Internet.org, "A focus on efficiency: A whitepaper from Facebook, Ericsson and Qualcomm," Sept. 2013.

[2] M. Pickett, "Building on deep learning," DTIC Document, Tech. Rep., 2013.

[3] S. Bengio, L. Deng, H. Larochelle, H. Lee, and R. Salakhutdinov, "Guest editors' introduction: Special section on learning deep architectures," PAMI, vol. 35, no. 8, pp. 1795–1797, 2013.

[4] D. Koller and N. Friedman, *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[5] W. L. Buntine, "Chain graphs for learning," in *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers Inc., 1995, pp. 46–54.

[6] L. Getoor, "Learning statistical models from relational data," PhD dissertation, Stanford University, 2002.

[7] J. Pearl, "Fusion, propagation, and structuring in belief networks," *Artificial intelligence*, vol. 29, no. 3, pp. 241–288, 1986.

[8] ——, "On logic and probability," *Computational Intelligence*, vol. 4, no. 2, pp. 99–103, 1988.

[9] ——, *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1988.

[10] G. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[11] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[12] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[13] G. L. Murphy, "Theories and concept formation," 1993.

[14] B. M. Lake, R. Salakhutdinov, J. Gross, and J. B. Tenenbaum, "One shot learning of simple visual concepts," in *Proceedings of the 33rd Annual Conference of the Cognitive Science Society*, vol. 172, 2011.

[15] H. F. Harlow, "The formation of learning sets." *Psychological Review*, vol. 56, no. 1, p. 51, 1949.

[16] T. R. Shultz, F. Rivest, L. Egri, and J.-P. Thivierge, "Knowledge-based learning with kbcc," *Neural Networks*, vol. 20, p. 21, 2006.

[17] D. Marr, *Vision: A computational investigation into the human representation and processing of visual information*. W.H. Freeman, San Francisco, CA, 1982.

[18] T. R. Shultz and F. Rivest, "Knowledge-based cascade-correlation: Using knowledge to speed learning," *Connection Science*, vol. 13, no. 1, pp. 43–72, 2001.

[19] C. Poultney, S. Chopra, Y. LeCun *et al.*, "Efficient learning of sparse representations with an energy-based model," in *Advances in Neural Information Processing Systems*, 2006, pp. 1137–1144.

[20] R. Salakhutdinov and G. E. Hinton, "Deep boltzmann machines," in *International Conference on Artificial Intelligence and Statistics*, 2009, pp. 448–455.