Relational Neural Markov Random Fields

Yuqiao Chen
University of Texas at Dallas

Sriraam Natarajan
University of Texas at Dallas

Nicholas Ruozzi
University of Texas at Dallas

Abstract

Statistical Relational Learning (SRL) models have attracted significant attention due to their ability to model complex data while handling uncertainty. However, most of these models have been restricted to discrete domains owing to the complexity of inference in continuous domains. In this work, we introduce Relational Neural Markov Random Fields (RN-MRFs) that allow handling of complex relational hybrid domains, i.e., those that include discrete and continuous quantities, and we propose a maximum pseudolikelihood estimation-based learning algorithm with importance sampling for training the neural potential parameters. The key advantage of our approach is that it makes minimal data distributional assumptions and can seamlessly embed human knowledge through potentials or relational rules. Our empirical evaluations across diverse domains, such as image processing and relational object mapping, demonstrate its practical utility.

1 Introduction

Statistical relational learning (SRL) models (Getoor and Taskar, 2007; Raedt et al., 2016) have gained popularity for their ability to learn and reason in the presence of noisy, uncertain, rich, and structured relational data. While expressive, these models face a significant challenge when learning from real data. One such challenge arises when learning from hybrid data: for predicate/relational logic based methods, significant feature engineering is necessary to make learning both effective and efficient, and most learning methods discretize the data (Natarajan et al., 2012; Khot et al., 2011) or make restrictive assumptions on the learned structure (Ravkic et al., 2015).

In classical propositional (feature vector based) domains, approaches that combine the benefit of neural networks (NNs) and graphical models have recently attracted attention. While specific network structure or training procedures differ, these methods generally employ conditional random fields (CRF) that are parameterized by NNs, e.g., in computer vision (Liu et al., 2015; Knöbelreiter et al., 2017). Neuro-symbolic learning in which NNs are combined with a logic-based formalism has also attracted attention (Garcez and Lamb, 2020; Raedt et al., 2020; Qu and Tang, 2019).

Our approach embeds NNs into (relational) graphical models. Specifically, we consider the parametric factor graph formalism (Poole, 2003; Braz et al., 2005) and employ neural potential functions that are flexible enough to handle hybrid (relational) data. In contrast to many existing approaches, our resulting model, dubbed relational neural Markov random fields (RN-MRFs), does not make any restrictive assumptions on the data distribution, e.g., multivariate Gaussian assumptions, nor is it dependent on a specific NN architecture. Instead, the model's expressivity can be appropriately controlled by tuning the network structure/activation functions.

We make the following key contributions. (1) We introduce a general combination of traditional SRL models and NNs by introducing neural potential functions inside parfactor graphs. Our relational neural (RN)-MRF does not make any distributional assumptions and can handle propositional, semi-relational, and relational domains. (2) We explain how to integrate human knowledge in the form of informative potentials and/or relational rules. (3) We present an efficient training procedure for RN-MRFs using maximum pseudolikelihood estimation. (4) We demonstrate the efficacy of RN-MRFs against non-neural potential baselines on several real domains.

2 Related Work

The combination of graphical models and NNs has been extensively explored in image processing and related

Proceedings of the 25th International Conference on Artificial Intelligence and Statistics (AISTATS) 2022, Valencia, Spain. PMLR: Volume 151. Copyright 2022 by the author(s).

areas. The work of Do and Artieres (2010) and Xiong and Ruozzi (2020) on combining NNs and Markov random fields are the most relevant to the approach herein. Do and Artieres (2010) propose neural potential functions in the context of conditional random fields (CRF), where the NN learns a feature representation given the conditional values of a log-linear model. Model weights are then chosen via maximum likelihood estimation. This approach was specifically designed for discrete random variables and additional care is needed in the continuous case to ensure normalizability of the model as well as to ensure that the resulting model fitting procedure can be implemented efficiently. A similar approach was also explored in the relational setting by Marra and Kuzelka (2019) where they model the possible world distribution with relational neural network potential, which share similarity with our method, but cannot handle continuous features.

Xiong and Ruozzi (2020) consider learning MRF models with NN potentials of the form $\exp(nn(x))$, which allows both continuous and discrete domains. Their approach uses the Bethe free energy to perform approximate variational inference for use in the computation of the MLE gradient. Their approach can be applied to learn general MRFs, but as it is not designed to handle relation models, it would be quite slow if applied directly to relational domains. Further as their proposed approximate inference procedure can be unstable, the method also requires an iterative averaging procedure to ensure convergence.

Alternative approaches that combine CRFs and NNs have used NNs to select the CRF parameters. These approaches have been particularly popular in computer vision applications in which large-scale models are common, e.g., (Liu et al., 2015; Zheng et al., 2015; Knöbelreiter et al., 2017). In these settings, the CRFs often have a simple structure and restricted potential functions, e.g., Gaussian, in order to make (approximate) inference practical. Other approaches have demonstrated that there is a close connection between approximate MAP inference and recurrent NNs (Wu et al., 2016).

There are also approaches like Deep ProbLog (Manhaeve et al., 2018), which try to combine NNs with probabilistic reasoning models. In Deep ProbLog, neural predicates are used to handle low level features, and high level relationship between predicates are defined by the logical models, while in our methods, NNs models the dependencies among predicates and functions.

In comparison with most existing related works, which only focus on either discrete models, models that are conditioned on continuous variables, or Gaussian models, the main novelty of the proposed method is that it has the ability to learn and predict complicated distributions of continuous variables, and at the same time provides a mechanism for incorporating human knowledge for better performance. In additionally, our models could be reduced to Neural MLN (Marra and Kuzelka, 2019) or Neural CRF (Do and Artieres, 2010) if only discrete latent variables are present.

3 Preliminaries

A Markov random field (MRF) consists of a hypergraph $G = (\mathcal{V}, \mathcal{C})$, with variable nodes $\mathcal{V}$ and a set of hyperedges $\mathcal{C}$. Each node $i \in \mathcal{V}$ is associated with a variable x_i with domain $\mathcal{X}_i$. We consider hybrid MRFs in which $\mathcal{X}_i$ could be either discrete or continuous. Each hyperedge $c \in \mathcal{C}$ is associated with a non-negative potential function $\phi_c : \mathcal{X}_c \to \mathbb{R}_{\geq 0}$, where $\mathcal{X}_c = \cup_{i \in c}\{\mathcal{X}_i\}$ is the union of variables in the hyperedge c. Often, the nonnegative potential functions are represented in exponential form as $\phi_c(x_c) = \exp f_c(x_c)$, with $f_c : \mathcal{X}_c \to \mathbb{R}$. An MRF defines a joint probability distribution over joint variables $x \in \cup_{i \in \mathcal{V}}\{\mathcal{X}_i\}$

$$p(x) = \frac{1}{Z}\prod_{c \in \mathcal{C}} \phi_c(x_c) = \frac{1}{Z}\exp\left(\sum_{c \in \mathcal{C}} f_c(x_c)\right),$$

where the normalization $Z = \sum_x \prod_{c \in \mathcal{C}} \phi_c(x_c)$ (sums are replaced by integration for continuous variables).

Fitting MRFs to Data: Often, the potential functions are restricted to specific functional forms determined by a fixed set of parameters, θ. Given M training data points, $x^{(1)}, \dots, x^{(M)}$, the MRF and the corresponding distribution $p(x)$ can be fit to data via maximum likelihood estimation (MLE) by applying gradient ascent to maximize the log-likelihood (LL),

$$\log l(x^{(1)}, \dots, x^{(M)}; \theta) = \sum_{m=1}^{M} \log p(x^{(m)}; \theta).$$

The computation of the gradient in each step requires exact or approximate inference over the whole model. Exact inference is intractable in all but the simplest of models, and in large graphical models, e.g., relational models with a large number of instances, even approximate inference procedures can be computationally expensive especially in the case of hybrid models.

As an alternative to MLE the pseudolikelihood (PL) (Besag, 1974), tries to avoid the expensive inference step of MLE. The PL method approximates $p(x; \theta)$ as a product of univariate conditional distributions.

$$p(x; \theta) \approx \prod_{i \in \mathcal{V}} p(x_i \mid x_{\mathcal{V} \setminus i}) = \prod_{i \in \mathcal{V}} p(x_i \mid MB_i),$$

where

$$p(x_i \mid MB_i) = \frac{\exp \sum_{c \ni i} f_c(x_i \mid x_{c \setminus i})}{Z_i(MB_i)},$$

$MB_i = \cup_{c \supset i}\{x_c \setminus x_i\}$ is the Markov blanket of node i, and Z_i is a normalization term, which ensures that $p(x_i \mid MB_i)$ is a valid conditional probability distribution. Learning then attempts to find the model parameters that maximize the log-PL.

$$\log l(x^{(1:M)}; \theta) \approx \frac{1}{M} \sum_m \sum_{i \in \mathcal{V}} \log p(x_i^{(m)} \mid MB_i^{(m)}; \theta).$$

Computing the gradient with respect to the parameters yields

$$\nabla \log l(x^{(1:M)}; \theta) = \frac{1}{M} \sum_m \sum_{i \in \mathcal{V}} \sum_{c \supset i} \Big[\nabla f_c(x_c^{(m)}; \theta_c) - \mathop{\mathbb{E}}_{p(x_i \mid MB_i^{(m)}; \theta)} \Big(\nabla f_c(x_i, x_{c \setminus i}^{(m)}; \theta_c) \Big) \Big]. \tag{1}$$

The first term in the gradient of the log-PL involves the gradient of the log-potentials with respect to θ_c. The second term, which comes from the gradient of $\log Z_i$, involves an expectation with respect to the conditional distributions in the PL approximation. In the discrete case, the expectation is usually computed by enumerating all possible assignments. In the continuous case, the integral may not be computable in closed form and may need to be estimated numerically.

Relational MRFs: Our key goal is to model the relations between attributes of objects by MRFs, which we represent as *relational Markov random fields.* We take an approach similar to lifted first-order models (Poole, 2003; Choi et al., 2010). In our relational MRFs, we use *atoms* to refer the attributes of objects compactly, e.g., an *atom* could be $smoke(P)$ or $friend(P, P')$, indicating that a person smokes or that a person has friendship with another person, where P and P' are both logical variables that could be instantiated as any possible person. Given a set of logical variables, L, a substitution δ is an assignment of instances to L, e.g., in a predicate $smoke(P)$ under the substitution $\delta = \{P \to John\}$, the instantiated atom is $smoke(John)$.

Similar groups of attributes/objects are represented in the form of parametric factors, $\texttt{parfactor}(\phi, A, L, C)$, where ϕ is the potential function, A a set of *atoms*, L a set of logical variables, and C is the constraint deciding allowable substitutions. A relational MRF is defined by $\mathcal{F}$ a set of `parfactor`s, and is equivalent to an MRF after grounding (instantiating atoms with all possible substitutions). Similar to MRFs, a relational MRF defines a joint distribution over $RV(\mathcal{F})$ the set of all variables in the grounded graph.

$$P(RV(\mathcal{F})) = \frac{1}{Z} \prod_{h \in \mathcal{F}} \prod_{\delta \in \Delta_h} \phi_h(A_h \delta),$$

where Δ_h is the set of possible substitutions to logical variables L of `parfactor` h, and $A_h\delta$ is the set of grounded variables obtained by grounding A_h with δ.

4 Relational Neural Markov Random Fields

We now introduce relational neural Markov random fields (RN-MRFs), a generic MRF model that combines the expressiveness of NNs and the representational power of relational MRFs. For RN-MRFs, we propose to use NNs to model the log-potential functions. This type of potential can be arbitrarily expressive, allowing us to capture complex relationships among the model variables. The expressiveness can be tuned by altering the NN structure, the activation functions, and the training technique, e.g., using dropout.

We introduce two types of neural potential functions. First, for continuous domains with explicit boundaries, say $[-1, 1]$ and $[0, 1.5]$, the potential function is defined as $\phi_{nn}(x_c) = \exp nn(x_c)$, where $nn(x_c)$ is an artificial NN, e.g., MLP, with input dimension equal to dimension of the hyperedge c that takes each variable $x_i, i \in c$ as input to the NN. For typical choices of activation functions, e.g., ReLU, the resulting potential will be integrable over the domain of interest.

For unbounded domains, such as $(-\infty, +\infty)$, the potential function might not be normalizable, i.e., the corresponding integral may not exist. As a simple example, if $nn(x)$ is a linear function of x, the potential function is not integrable over $(-\infty, \infty)$ as it can attain arbitrarily large values depending on the choice of x. Such a potential does not correspond to a probability distribution, which creates representational issues as well as algorithmic issues. To ensure normalizability, we propose to have potentials of the form $\phi(x_c) = \phi_0(x_c) \cdot \phi_{nn}(x_c)$, where $\phi_0(x_c)$ is a helper distribution chosen to ensure normalizability.

As an example, if the neural net consists only of ReLU activations, then the resulting NN will be piecewise linear. Thus, the product of a Gaussian helper potential and the neural net potential will be piecewise-Gaussian (the sum of a negative quadratic function with a linear term is still a negative quadratic function). This is the approach to normalizability that we will adopt. If, in addition, the helper distribution is easy to sample from, we will see that it can be used to estimate the integrals that are required to compute the gradient of the log-pseudolikelihood.

Similar to relational MRFs, RN-MRFs are defined by a tuple $\texttt{parfactor}(\phi_0, \phi_f, A, L, C)$, where ϕ_f is not limited to NN potentials and can include other types of potentials such as Markov Logic Network (MLN) rules

(Richardson and Domingos, 2006). Bounded domains do not require helper functions, and they can be set to uniform distributions over the appropriate domains. The joint distribution corresponding to an RN-MRF can be factorized as a product of a joint neural potential and a helper distribution.

$$\begin{aligned} P(RV(\mathcal{F})) &= \frac{1}{Z} \prod_{h \in \mathcal{F}} \prod_{\delta \in \Delta_h} \phi_{nn}(A_h\delta) \cdot \phi_0(A_h\delta) \\ &= P_{nn}(RV(\mathcal{F})) \cdot P_0(RV(\mathcal{F})) \end{aligned}$$

4.1 Encoding Human Knowledge

One of the key attractive features of RN-MRFs is the ability to encode human knowledge in three different ways: (1) altering the structure of the model, i.e., assuming dependencies among a set of object attributes by creating a `parfactor` among a set of *atoms*, (2) designing the potential functions, i.e., by specifying the helper function, mapping features, or tuning the NN structure, and (3) adding (weighted) logic rules that define new relationships between features.

Consider a simple image denoising task in which the goal is to predict the original image given the noisy image. Below, we describe a possible RN-MRF to model relationships among pixels in this scenario. We use the notation *helper*:*potential* to describe the model.

$$\begin{aligned} &LG(1,0,1) : \phi_{nn1}(|val(P_1) - val(P_2)|) \text{ st. } nb(P_1, P_2) \\ &LG(1,0,1) : \phi_{nn2}(|obs(P) - val(P)|), \end{aligned} \tag{2}$$

where the first (parfactor) rule states that there exists a dependency between the values of adjacent pixels ($\{P_1, P_2\}$), the second rule states that there is a relationship between observed and true values of every pixel P, and the helper functions are specified as a two dimensional linear Gaussians (LG) with $slope = 1$, $intercept = 0$, and $variance = 1$.

For the helper distributions, we typically use either categorical distributions for discrete domains, multivariate Gaussians (or mixtures of Gaussians) for continuous domains, and categorical Gaussians for hybrid domains. The parameters of the helper functions can be learned if desired.

The above rules provide an example of the first mode of human knowledge integration. These two rules also include the second mode of knowledge integration where the relationships are precisely defined. For instance, the first rule specifies that the helper distribution is a linear Gaussian. The second rule also defines a linear Gaussian helper distribution on the absolute difference between the observed and true values of every pixel. Typically, when dealing with two pixels P_1 and P_2, they will both be employed as two different inputs for a neural net. However, the first `parfactor` allows for a preprocessing step that computes the absolute difference between the pixels and uses that as the input. This is akin to feature mapping for standard supervised learning.

The third method to encode the human knowledge is to specify `parfactor`s with (soft) logical rules that define potential functions with a tunable weight parameter. The higher the weight, the higher the probability of the rule being true for a given data set (Richardson and Domingos, 2006). For instance, consider a robot mapping domain where the task is to predict the type of an observed segment S given the length and depth of all segments. Length and depth are continuous features, while type is discrete ⟨'W', 'D', 'O'⟩, representing Wall, Door, and Other. We can express this in the RN-MRF formalism by introducing the following potentials/helpers.

$$\begin{aligned} &U : \phi_{nn}(length(S), depth(S), type(S)) \\ &U : \phi_{mln1}(length(S) > 0.5 \Rightarrow type(S) != \text{'W'}) \\ &U : \phi_{mln2}(type(S_1) = \text{'D'} \Rightarrow type(S_2) != \text{'D'}) \\ &\quad \text{st. } nb(S_1, S_2), \end{aligned}$$

where U is a uniform distribution, which is equivalent to not having a helper function.

The first `parfactor` uses the NN potential to model the conditional type distribution given the segment's length and depth. Since this dataset is small, using only the data observations to train the neural nets could result in overfitting. The second `parfactor` specifies that if the length of segment is larger than 0.5, it should be of type $Wall$. The logical formula can be computed given the length and the type, resulting in a binary value True/False (or 0/1), indicating whether or not the logical formula is satisfied.

The potentials are of the form $\phi_{mln}(x_c) = \exp[w \cdot logic(x_c)]$, where w is a learnable parameter representing the strength of the rule. If the rule is satisfied, the potential value will be $\exp w$, otherwise it is 1. The potential could also be used in representing the interrelationship between objects. For example in the third `parfactor`, the rule means that for a pair of adjacent segments, if one is of type *Door*, the other one should not be of the same type. This captures the knowledge that it is less likely for two doors to be next to each other. In addition, hybrid rules that use continuous values of the objects could also be employed.

5 Learning RN-MRFs

The main challenge when learning RN-MRFs is the presence of continuous variables. Most current learning algorithms do not support hybrid domains with arbitrary potentials. One recent exception is the MLE algorithm proposed by Xiong and Ruozzi (2020). While MLE could potentially be used to learn the model parameters, computing the gradient of the log likelihood requires inference, which is intractable for large MRFs or models with complicated potential functions. In relational settings, the size of the grounded RN-MRF is typically too large to admit efficient inference, and we are interested in models in which the *potential functions are arbitrary.* Hence, we propose to learn our model with maximum pseudolikelihood estimation instead. The MPLE approach learns the model parameters by maximizing the log-pseudolikelihood given the data, usually with gradient ascent. In the computation of the gradient of the log-pseudolikelihood, as in (1), two terms are considered: the gradient of the log-potential functions and the expectation of the log-potentials with respect to the local conditional distributions. For training the parameters of the NNs, we can first calculate the gradient of the pseudolikelihood with respect to the output of the neural net and then apply back propagation to obtain the gradient of the network parameters.

While computing the first piece of the gradient is straightforward, calculating the expectation term is nontrivial. This is because there is not a closed form equation for computing the expectation of an arbitrary function with respect to an arbitrary continuous distribution. We propose to use (self-normalized) importance sampling for the calculation.

$$
\begin{aligned}
&\mathbb{E}_{p(x_i|MB_i^{(m)})} \nabla nn_c(x_i, x_{c\setminus i}^{(m)}) \\
&= \int_{-\infty}^{\infty} p(x_i|MB_i^{(m)}) \cdot \nabla nn_c(x_i, x_{c\setminus i}^{(m)}) dx_i \\
&= \int_{-\infty}^{\infty} \frac{b_i(x_i)}{\int_{-\infty}^{\infty} b_i(\hat{x}_i) d\hat{x}_i} \cdot \nabla nn_c(x_i, x_{c\setminus i}^{(m)}) dx_i \\
&\approx \frac{1}{\sum_{n\sim Q} \frac{b_i(x_i^{(n)})}{Q(x_i^{(n)})}} \sum_{n\sim Q} \frac{b_i(x_i^{(n)})}{Q(x_i^{(n)})} \cdot \nabla nn_c(x_i^{(n)}, x_{c\setminus i}^{(m)}),
\end{aligned}
\tag{3}
$$

where $b_i(x_i) = \prod_{h\ni i} \phi_h(x_i, x_{h\setminus i}^{(m)})$, and Q is a proposal distribution from which we can draw samples for approximating the integral. In the case that the variable i is bounded, the proposal distribution could be chosen to be a uniform distribution over the objects in the domain.

In case of unbounded domains and/or non-trivial helper functions, the proposal could be the product of helper functions that make up the conditional distribution, i.e., $Q(x_i) = \prod_{h\ni i} \phi_{0_h}(x_i, x_{h\setminus i}^{(m)})$. As the potentials in RN-MRFs are the product of helper functions and the neural potentials, the computation of the ratio $\frac{b_i(x_i)}{Q(x_i)}$ can be simplified.

$$
\begin{aligned}
\frac{b_i(x_i)}{Q(x_i)} &= \frac{\prod_{h\ni i} \phi_{0_h}(x_i, x_{h\setminus i}^{(m)}) \cdot \phi_{nn_h}(x_i, x_{h\setminus i}^{(m)})}{\prod_{h\ni i} \phi_{0_h}(x_i, x_{h\setminus i}^{(m)})} \\
&= \prod_{h\ni i} \phi_{nn_h}(x_i, x_{h\setminus i}^{(m)})
\end{aligned}
$$

The expectation can then be approximated as

$$
\frac{\sum_{n\sim Q} \prod_{h\ni i} \phi_{nn_h}(x_i, x_{h\setminus i}^{(m)}) \cdot \nabla nn_c(x_i^{(n)}, x_{c\setminus i}^{(m)})}{\sum_{n\sim Q} \prod_{h\ni i} \phi_{nn_h}(x_i, x_{h\setminus i}^{(m)})}. \tag{4}
$$

Given these gradients, we turn our attention to training. A naïve method to train the graphical model would be to compute the gradient of the parameters for each local variable distribution, sum them to build the full gradient, and perform one step of gradient ascent. However, this approach would require many passes of neural net forward/backward operations. In an RN-MRF, many of the factors share the same potential function, making parallelization possible. To increase efficiency, we consider creating an aggregated input matrix $\{x_c^{(m)}\}_{data} \cup \{x_i^{(n)} \times x_{c\setminus i}^{(m)}\}_{samples}$ for each NN potential, which includes both the data points and sampling points used in the approximation of the gradient. Then, we could run one pass of feed forward operation to obtain the value of $\phi_{nn_h}(x_i, x_{h\setminus i}^{(m)})$ and use it to compute the corresponding gradient for each data point. For $x \in \{x_c^{(m)}\}_{data}$, the gradient is 1, and for sampling data $x \in \{x_i^{(n)} \times x_{c\setminus i}^{(m)}\}_{samples}$, the gradient is the negative of the expectation term, which can be computed with equation (4). Finally, we back propagate the gradient through the neural net and update the network parameters with gradient ascent.

Additionally, given the expressive power of NNs, modeling distributions using NNs can be prone to overfitting, especially on continuous domains where the learned distribution could have high peaks at the data points and be zero outside of them. To reduce the chance of severe overfitting, popular methods such as weight decay and drop out could be applied. However, we propose to modify the NNs by clamping the output layer, i.e., $clamp(nn(x), a, b)$ bounds the output from below by a and above by b, to prevent the learned potentials from taking values that are too extreme. In back propagation, positive gradients are not propagated if $nn(x) > b$, and same for negative gradients when $nn(x) < a$.

A complete description of the RN-MRF learning algorithm can be found in Algorithm 1. During each iteration, a set of grounded variables will be sampled. This is performed at the first order level by sampling substitutions of atoms, and Markov blanket of the resulting

Algorithm 1 Learning RN-MRF

```
1: Input: A RN-MRF $\mathcal{F}$, set of trainable potentials $\Phi$, and dataset $\mathcal{M}$
2: Return: Set of learned potential functions
3: repeat
4:     Uniformly draw a set of variables $\mathcal{V}_s$ from all grounded atoms, and find their Markov Blanket.
5:     Initialize the aggregated input matrix $\mathcal{D}_c$ for $\phi_c \in \Phi$
6:     for each data frame $m \in \mathcal{M}_s$ do
7:         for each variable $i \in \mathcal{V}_s$ do
8:             Let proposal $Q(x_i) = \prod_{h \supset i} \phi_{0_h}(x_i, x^{(m)}_{h \setminus i}))$
9:             Sample N number of $x^{(n)}_i \sim Q$
10:            for each $\phi_c$ where $c \supset i$ do
11:                Add sample $x^{(m)}_c$ to $\mathcal{D}_c$
12:                Add samples $x^{(n)}_i \times x^{(m)}_{c \setminus i}$ to $\mathcal{D}_c$
13:            end for
14:        end for
15:    end for
16:    Run forward pass for each $\phi_c \in \Phi$ with data $\mathcal{D}_c$
17:    Compute the gradient w.r.t the neural network output
18:    Run back propagation for each $\phi_c \in \Phi$
19:    Update each $\theta_c$ with gradient ascent
20: until Convergence
```

grounded variables can be also grounded through unification. After that, the aggregated data is computed, followed by a forward pass of the neural nets. The gradient of the log pseudo likelihood w.r.t. the output of the neural net is computed and back propagated through the whole network. Finally, parameters are updated using standard gradient ascent.

6 Experiments

We now present empirical evidence of the flexibility of RN-MRFs by considering different tasks/data domains. We try to answer the following questions with empirical validation.

Q1: Do RN-MRFs provide an effective data representation in continuous or hybrid domains?

Q2: How well do the neural potentials model complicated (high dimensional and multi-modal) dependencies?

Q3: Can human knowledge be easily incorporated to improve performance?

We selected several different problem domains: image denoising with multi-modal noise, digit deduction on MNIST images with synthetic rules, and relational object mapping with human knowledge. For each experiment, we use different baselines according to the domain, including Gaussian and/or Categorical Gaussian models, Neural CRF (Do and Artieres, 2010), and expert/hand-created models, such as hybrid MLNs (Wang and Domingos, 2008). Notice that we do not compare against carefully engineered method that work on specific tasks as the goal is to show the generalized effectiveness of RN-MRF across variety of domains.

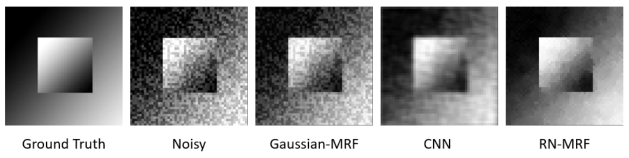

Figure 1: Sample outputs of different models on a image denoising task, where noise are following multi-modal distribution.

We report the ℓ_1 and ℓ_2 error of MAP predictions for continuous domains and the MAP prediction accuracy rate and F1 score for discrete domains. For continuous and hybrid domain inference, we use expectation particle belief propagation (EPBP) (Lienart et al., 2015), which use sample points for approximating the continuous BP messages with a dynamic Gaussian proposal. All algorithms were run on a single core of a machine with a 2.2 GHz Intel Core i7-8750H CPU and 16 GB of memory, and were implemented in Python3.6 and source code is available via an anonymous repository `https://github.com/leodd/Relational-Neural-MRF`.

6.1 Image Denoising

In this experiment, we showcase the application of our RN-MRF model on image domains with a simple image denoising task, e.g., an image has been corrupted with Gaussian mixture noise, which requires the model to learn the relationship between neighboring pixels and observations. The reason that we choose this domain is threefold: (1) while at the outset, this domain appears propositional, the inherent grid structure allows a relational model to effectively model this task; (2) variables are naturally continuous in this domain; and (3) the optimal potential functions cannot be easily represented with a simple distribution, e.g., a Gaussian distribution. For this task, the dataset was created by adding multi-modal noise, generated by a mixture of Gaussian distribution $0.5 \cdot \mathcal{N}(0.1, 0.0025) + 0.5 \cdot \mathcal{N}(-0.1, 0.0025)$, to 90, 300 x 400 images and 6, 50 x 50 synthetic images.

We consider a grid structured MRF/CRF, with one layer of random variables for the input noisy image and one layer of random variables modeling the original

Table 1: Comparison of different models on the image denoising task.

Model	ℓ_1 Error	ℓ_2 Error
Gaussian-MRF	186.14 ± 7.26	18.97 ± 1.47
CNN	120.33 ± 13.70	12.86 ± 3.33
RN-MRF(no Helper & FM)	58.55 ± 5.89	3.07 ± 0.89
RN-MRF(with only FM)	94.28 ± 17.12	7.70 ± 1.98
RN-MRF(with Helper & FM)	99.97 ± 20.09	7.81 ± 2.47

image. We model two types of dependencies – (1) a dependency that connects the noisy observation with the corresponding random variable in the original image, and (2) a pairwise dependency between two random variables that correspond to neighboring pixels in the original image. To represent these dependencies, we define an RN-MRF as in (2). Both the potentials are modeled with linear Gaussian helper function and with the feature mapping (FM) $|x_1 - x_2|$. Both neural net potentials ϕ_{nn1} and ϕ_{nn2} have two hidden layers, with 64 neurons for the first layer and 32 neurons for the second layer.

To investigate the modeling power of RN-MRF, we compare our model with Gaussian-MRFs, which have potential functions of the form $\exp(.5(x-\mu)\Sigma(x-\mu)^T)$ where the mean vector μ and covariance matrix Σ are learnable parameters, and with convolutional neural networks with 3 convolution layers (conv1: 32 channels, conv2: 16 channels, conv3: 1 channels, each has kernel size 3). Additionally, we also evaluate the RN-MRF model both with and without a linear Gaussian helper function and without both the feature mapping and the helper function.

All models were trained using PMLE with the Adam optimizer for 1500 training iterations (until convergence). For each iteration, the algorithm randomly samples 100 variables for computing the aggregated input matrix, and approximates the expectation term with 20 sampling points. For inference, we used EPBP with a Gaussian proposal distribution and approximate the messages with 50 sampling points. We report the average image ℓ_1 and ℓ_2 error.

Figure 1 visualizes the performance of the different approaches on sample test data points. CNN produces blurry result because it does not model the relationship between neighboring pixels explicitly. Gaussian MRF produces noisy results as the Gaussian function fails to fully capture multi-modal dependency between the ground truth pixel and the observation. With neural potential functions, our RN-MRF model was able to model the complicated relationship, leading to successful recovery. Similar conclusions can be obtained by inspecting the ℓ_1 and ℓ_2 errors on the test set, see Table 1. This suggests that the RN-MRF framework is effective at learning dependency between continuous variables (**Q1**).

It can be observed that RN-MRF with Feature Mapping produces a less accurate result as compared to RN-MRF without any assumption embedded. As shown in the supplementary material, resulting images of RN-MRF(with FM) are visually similar to the one without assumption, but with a slightly different tune. This is likely due to the assumption enforced by feature mapping, which suggests pixels with different gray scale share the same distribution. However, due to how data were generated, the pixel value is clipped to $[0, 1]$ when noise is added, so the darkest and brightest pixels will have higher probability. This shows that *even without careful tuning, our learning algorithm and neural potentials are able to perform well* (**Q2**). Also, notice that RN-MRF (with FM & Helper) has similar result as RN-MRF (with FM), because the helper function only helps sampling, which does not have much impact in this task where continuous domain is bounded. It is important to note that although having helper functions might not result in better performance in bounded domains, it is necessary for both the inference and learning algorithm to function properly in unbounded domains.

6.2 Digit Deduction

To showcase the usability and the ability of our model in combining neural networks with relational logic rules, we perform a toy digit deduction task on a synthetic time conversion dataset based on MNIST dataset. In this task, we consider a digit prediction task given four MNIST images associated with a set of relational rules. The first image and second image constitute UTC time $T_1 \in \{0, ..., 23\}$, and third and fourth image form the local time $T2$. Given the local time zone $Z \in \{-11, ..., +0, ..., +12\}$, we enforced a relationship among these images. To model this task, we create the following RN-MRF.

$U : \phi_{cnn}(Img, D)$
$U : \phi_{mln1}(T_1 = 10 * D_1 + D_2)$ st. D_1 and D_2 form T_1
$U : \phi_{mln2}((T_1 + Z)\%24 = T_2)$.

The neural potential ϕ_{cnn} is a convolutional neural network with 2 convolution layers and 2 linear layers (conv1: 8 channels, conv2: 3 channels, both with kernel size 3, fc1: 64 neurons, fc2: 10 neurons), which models the distribution of the actual digit given the image. Potentials ϕ_{mln1} and ϕ_{mln2} model the rules converting digit to time and converting UTC time to local time given the time zone. For the training data and testing, we generate 1000 triples of T_1, T_2 and Z randomly. Images were randomly picked from the MNIST dataset to form T_1 and T_2. For prediction, both images and

Table 2: Comparison of different models on the digit deduction task.

Model	Digit Accuracy	Time Accuracy
CNN	0.925 ± 0.007	N/A
RN-MRF(with MLN)	0.970 ± 0.006	0.945 ± 0.009
RN-MRF(ϕ_{nn} only)	0.925 ± 0.009	0.045 ± 0.006

the time zone are given. Digits and time will be predicted by MAP inference with EPBP. We compare the prediction accuracy of RN-MRF model with a neural net counterpart without rules embedded.

As Table 2 shows, RN-MRF with MLN rules produces a more accurate result compared with the CNN, which does not include the additional constraint – showing the importance of adding human knowledge. We also tried modeling the aforementioned constraints with neural potentials. However, the result is similar to the plain CNN. This is likely a result of the limited training data; we only have 1000 data points for modeling the time conversion relation, which has 24^3 possible states. The addition of logical rules in RN-MRF helps to regularize and compensate for the lack of data (**Q3**).

6.3 Robot Mapping

To investigate the power of RN-MRF in modeling hybrid relational data and the usefulness of using a weighted logic model to capture human knowledge, we consider a real-world relational robotic map building domain (Limketkai et al., 2005), where the goal is to build a labeled object map of indoor spaces from a set of laser range-scanned line segments defined by their endpoint coordinates. In this experiment, the laser scanned data are corridor maps from the Radish robotics data set repository. Each line segment can be mapped to one of three different types of objects {'Wall', 'Door', 'Other'}. The ground truth labeling, constructed by hand, is given in the dataset. Similar to Wang and Domingos (2008), we pre-process the endpoint data by computing the segments' length, depth, and angle, which are considered as evidence. The pre-processing step will make it easier for us to come up with meaning logical rules. We also include the *neighbor* predicate used by Wang and Domingos (2008): $neighbor(s_1, s_2)$ is true if the minimum endpoint distance between segment s_1 and s_2 is under a specified threshold.

We chose this domain for several reasons: (1) it is relational and has both continuous and discrete features, (2) some relationships between features can be encoded in the form of logical language while some relationships are more complicated and can be represented with neural potentials and (3) the training data size is small – adding human knowledge/domain expertise should help to prevent overfitting. To predict the segment type, we define the following RN-MRF.

$$
\begin{aligned}
&U : \phi_{nn1}(length(S), depth(S), angle(S), type(S))\\
&U : \phi_{nn2}(depth(S_1) - depth(S_2), type(S_1), type(S_2))\\
&U : \phi_{mln1}(angle(S) > 30^\circ \Rightarrow type(S) != \text{'W'})\\
&U : \phi_{mln2}(angle(S) > 89^\circ \Rightarrow type(S) = \text{'O'})\\
&U : \phi_{mln3}(type(S_1) = \text{'D'} \Rightarrow type(S_2) != \text{'D'})\\
&\quad \text{st. } nb(S_1, S_2)
\end{aligned}
$$

Neural potential $nn1$ models the local relationship among the length, depth, angle, and type, which in principle would be sufficient on its own. The type of a segment can also be predicted given the type of its neighbor and their depth difference; this relationship is modeled by $nn2$. For example, a door object is usually "deeper" than wall objects. If a segment is a wall and has a higher depth value than its neighbor, then its neighbor has higher probability to be a door object. We also include three MLN potentials. The first suggests that if the angle of a segment is larger than 30°, then it is likely not a wall object. Similarly, the second formula implies that a segment is of type 'Other' if its angle is larger than 89°. Finally, $mln3$ encodes the knowledge that neighboring segments are both not likely to be door objects.

We compare RN-MRFs with Categorical Gaussian MRFs, hybrid MLNs (Wang and Domingos, 2008), and Neural CRFs (Do and Artieres, 2010). For hybrid MLNs, we use the HMLN constructed by Wang and Domingos (2008). For CG MRFs and Neural CRFs, we use the grounded graph defined by the above RN-MRF, which could be considered as a CRF model if the length, depth, and angle of all segments are given. To make the comparison between RN-MRFs and these discriminative models fair, we also train our model conditionally and report the result as RN-CRF. Additionally, in order to show the effect of MLN potentials, we include the model both with and without MLN rules. For all potentials that involve a neural network, the neural model is set to a fully connected network with 2 hidden layers (64 and 32 neurons, ReLU activation).

The dataset consists of 5 distinct laser-scanned maps. For each map, both the endpoint coordinates and relational predicates are provided. In our experiment, we evaluate all models and algorithms using leave-one-out cross validation. All models are trained with PMLE (3000 iterations, 30 sampling points) and tested with EPBP (20 sampling points). The results are shown in Table 3.

RN-MRF and RN-CRF perform similarly to Neural CRF, which is likely the result of their somewhat similar reliance on neural networks in the construction

Table 3: Comparison on the robot mapping task.

Model	Accuracy	F1(Wall)	F1(Door)	F1(Other)
RN-MRF	0.876	0.944	0.821	0.809
RN-MRF(no MLN)	0.852	0.912	0.807	0.781
RN-CRF	0.901	0.945	0.88	0.836
RN-CRF(no MLN)	0.88	0.935	0.842	0.812
Neural-CRF	0.884	0.961	0.842	0.782
CG-MRF	0.762	0.838	0.703	0.667
Expert HMLN	0.761	0.815	0.807	0.487

of the potential functions. All of the neural network based methods significantly outperform Categorical Gaussian MRFs and the expert-specified hybrid MLN. One reason for the significant discrepancy is that the local features (length, depth and angle of a segment) are not well-modeled with a unimodal distribution, e.g., in the data set the length distribution of wall objects is closer to bimodal. However, both CG-MRF and HMLN use Gaussians for modeling local continuous features. This suggests that the neural potentials accurately encode this multimodality (**Q2**). Also note that both RN-MRF and RN-CRF are slightly better than their counterparts without MLN rules. With the neural potentials alone, the RN-MRF appears to capture many of the most important dependencies accurately, but there are also some edge cases, which rarely appear in the dataset, that are difficult to learn. With the addition of a few simple MLN rules, the model better able to account for these dependencies (**Q3**).

7 Discussion

We presented a general relational MRF model that seamlessly handles complex dependencies in relational and hybrid domains and allows for human knowledge to be specified as an inductive bias either in the form of priors or weighted logic rules. When modeling these complex dependencies, RN-MRFs only need to make minimal assumptions on the underlying distribution by utilizing the expressiveness of neural potential functions. In addition, we presented a maximum pseudo-likelihood learning algorithm for general relational MRF/CRF models, that performs well in the context of relational domains and can be trained efficiently. Our empirical evaluations show that the RN-MRF model can be applied to various domains and performs better than non-neural potential based modeling approaches under the same learning conditions. There are two main limitations for our proposed method: 1) The MPLE algorithm cannot learn with missing data. If a grounded atom does not have data associated with it, we can drop it and its Markov blanket during learning. In certain extreme cases, it is possible that none of the grounded atoms can be used for learning. 2) Different neural potentials might need different learning rates, which is problematic if we need to learn a large number of potentials at the same time. For future work, we plan to explore scaling up this approach to graphs with missing data or latent variables, allowing for richer human knowledge such as preferences and/or qualitative constraints and exploring the trade-offs/benefits versus neuro-symbolic learning methods.

Acknowledgements

This work was supported in part by the DARPA Explainable Artificial Intelligence (XAI) Program under contract number N66001-17-2-4032, by the DARPA Perceptually-enabled Task Guidance (PTG) Program under contract number HR00112220005. Sriraam Natarajan gratefully acknowledges the support of ARO award W911NF2010224.

References

J. Besag. Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society: Series B (Methodological)*, 36(2):192–225, 1974.

R. D. S. Braz, E. Amir, and D. Roth. Lifted first-order probabilistic inference. In *IJCAI*, 2005.

J. Choi, E. Amir, and D. J. Hill. Lifted inference for relational continuous models. In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence (UAI)*, 2010.

T. Do and T. Artieres. Neural conditional random fields. In Y. W. Teh and M. Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 177–184. JMLR Workshop and Conference Proceedings, 2010.

A. Garcez and L. C. Lamb. Neurosymbolic AI: the 3rd wave. *CoRR*, abs/2012.05876, 2020.

L. Getoor and B. Taskar. *Introduction to Statistical Relational Learning.* MIT Press, 2007.

T. Khot, S. Natarajan, K. Kersting, and J. Shavlik. Learning Markov logic networks via functional gradient boosting. In *ICDM*, pages 320–329, 2011.

P. Knöbelreiter, C. Reinbacher, A. Shekhovtsov, and T. Pock. End-to-end training of hybrid cnn-crf models for stereo. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1456–1465, 2017.

T. Lienart, Y. W. Teh, and A. Doucet. Expectation particle belief propagation. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3609–3617, 2015.

B. Limketkai, L. Liao, and D. Fox. Relational object maps for mobile robots. In *IJCAI*, 2005.

F. Liu, C. Shen, G. Lin, and I. Reid. Learning depth from single monocular images using deep convolutional neural fields. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(10):2024–2039, 2015.

R. Manhaeve, S. Dumancic, A. Kimmig, T. Demeester, and L. D. Raedt. Deepproblog: Neural probabilistic logic programming. In *NeurIPS*, 2018.

G. Marra and O. Kuzelka. Neural markov logic networks. *CoRR*, abs/1905.13462, 2019.

S. Natarajan, T. Khot, K. Kersting, B. Gutmann, and J. Shavlik. Gradient-based boosting for statistical relational learning: The Relational Dependency Network case. *MLJ*, 2012.

D. Poole. First-order probabilistic inference. In *IJCAI*, 2003.

M. Qu and J. Tang. Probabilistic logic neural networks for reasoning. In *NIPS*, 2019.

L. D. Raedt, K. Kersting, S. Natarajan, and D. Poole. *Statistical Relational Artificial Intelligence: Logic, Probability, and Computation.* Morgan & Claypool, 2016.

L. D. Raedt, S. Dumančić, R. Manhaeve, and G. Marra. From statistical relational to neuro-symbolic artificial intelligence. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, 2020.

I. Ravkic, J. Ramon, and J. Davis. Learning relational dependency networks in hybrid domains. *Mach. Learn.*, 100(2-3):217–254, 2015.

M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62, 2006.

J. Wang and P. Domingos. Hybrid markov logic networks. In *Twenty-Second AAAI Conference on Artificial Intelligence (AAAI)*, 2008.

Z. Wu, D. Lin, and X. Tang. Deep markov random field for image modeling. In B. Leibe, J. Matas, N. Sebe, and M. Welling, editors, *Computer Vision – ECCV 2016*, pages 295–312. Springer International Publishing, 2016.

H. Xiong and N. Ruozzi. General purpose mrf learning with neural network potentials. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*. International Joint Conferences on Artificial Intelligence Organization, 2020.

S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr. Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1529–1537, 2015.