Machine Learning
https://doi.org/10.1007/s10994-021-06069-5

A taxonomy of weight learning methods for statistical relational learning

Sriram Srinivasan[1] · Charles Dickens[1] · Eriq Augustine[1] · Golnoosh Farnadi[2] · Lise Getoor[1]

Received: 27 June 2020 / Revised: 18 August 2021 / Accepted: 7 September 2021

Abstract

Statistical relational learning (SRL) frameworks are effective at defining probabilistic models over complex relational data. They often use weighted first-order logical rules where the weights of the rules govern probabilistic interactions and are usually learned from data. Existing weight learning approaches typically attempt to learn a set of weights that maximizes some function of data likelihood; however, this does not always translate to optimal performance on a desired domain metric, such as accuracy or F1 score. In this paper, we introduce a taxonomy of search-based weight learning approaches for SRL frameworks that directly optimize weights on a chosen domain performance metric. To effectively apply these search-based approaches, we introduce a novel projection, referred to as scaled space (SS), that is an accurate representation of the true weight space. We show that SS removes redundancies in the weight space and captures the semantic distance between the possible weight configurations. In order to improve the efficiency of search, we also introduce an approximation of SS which simplifies the process of sampling weight configurations. We demonstrate these approaches on two state-of-the-art SRL frameworks: Markov logic networks and probabilistic soft logic. We perform empirical evaluation on five real-world datasets and evaluate them each on two different metrics. We also compare them against four other weight learning approaches. Our experimental results show that our proposed search-based approaches outperform likelihood-based approaches and yield up to a 10% improvement across a variety of performance metrics. Further, we perform an extensive evaluation to measure the robustness of our approach to different initializations and hyperparameters. The results indicate that our approach is both accurate and robust.

Keywords Statistical relational learning · Weight learning · Probabilistic graphical models · Markov logic networks · Probabilistic soft logic · Black-box optimization

Editor: Luc De Raedt

✉ Sriram Srinivasan
ssriniv9@ucsc.edu

Extended author information available on the last page of the article

Published online: 13 December 2021

1 Introduction

Statistical relational learning (SRL) frameworks (Richardson and Domingos 2006; De Raedt and Kersting 2011; Getoor and Taskar 2007) combine the power of graphical models with probabilistic programming to produce accurate models on complex relational data. Often these frameworks use first-order logical rules to represent complex relations and a relational dataset/database to instantiate a probabilistic graphical model. Several SRL frameworks have been developed in the past years; some of them generate directed graphical models (Poole 1993; Jaeger 1997; Friedman et al. 1999; Neville and Jensen 2007), and others generate undirected graphical models (Taskar et al. 2002; Richardson and Domingos 2006; Bach et al. 2017). More frameworks exist that extend existing statistical methods like logistic regression to relational realm (Mehran Kazemi et al. 2014) and others that extend logic programming to support probabilistic inference (Sato 1995; Fierens et al. 2015; De Raedt et al. 2007). Of these frameworks, Markov logic networks (MLNs) (Richardson and Domingos 2006) and probabilistic soft logic (PSL) (Bach et al. 2017) are two popularly used SRL frameworks in the context of undirected graphical models. Both make use of weighted first-order logical rules to generate a version of logical Markov random fields (MRFs). While MLNs use discrete logic to construct a MRF over discrete random variables, PSL uses soft logic to construct a special kind of MRF over continuous random variables in range [0, 1] called a hinge-loss Markov random field (HLMRF). PSL and MLNs have achieved state-of-the-art results in various domains such as recommender systems (Kouki et al. 2017; Lalithsena et al. 2017; Choi et al. 2015), bioinformatics (Sridhar et al. 2016), natural language processing (Ebrahimi et al. 2016; Johnson et al. 2017; Khot et al. 2015; Beltagy et al. 2013), product search (Alshukaili et al. 2016; Platanios et al. 2017; Srinivasan et al. 2019), fake news detection (Chowdhury et al. 2020) and social network analysis (Farnadi et al. 2017; Chen et al. 2017).

Learning the weights of the rules is one of the key challenges for SRL frameworks that use logical rules to define their models such as PSL and MLNs since the weighted rules interact in complex ways and cannot be optimized independently. Because these rules act as templates to generate the full graphical model, the weights of these rules, i.e., the parameters of the model, are used in multiple places in the instantiated graphical model, and the context varies depending on the other rules that have been instantiated. In addition, the corresponding probability distribution is not easy to compute; specifically, computing the normalization constant is often intractable.

Typically, the weights of the rules are learned through maximizing some form of likelihood function (Bach et al. 2013; Lowd and Domingos 2007; Singla and Domingos 2005; Kok and Domingos 2005; Chou et al. 2016; Sarkhel et al. 2016; Das et al. 2016, 2019; Farabi et al. 2018). This is a well-motivated approach if the downstream objective makes use of the probability density function directly. However, the objective is to often improve an external domain metric such as accuracy, F1 for classification, or AUROC for ranking. Several approaches address this issue by augmenting the metric into a loss function and solving a max-margin problem (Huynh and Mooney 2009, 2010; Bach et al. 2013). This does not directly optimize the desired metric but instead optimizes a surrogate loss. Further, such approaches do not easily extend to new metrics as they require deriving new losses, which may be non-convex and hard to optimize.

In our previous work (Srinivasan et al. 2020b), we introduced the first weight learning approach based on Bayesian optimization (a search-based approach) for PSL. This was, to the best of our knowledge, the first application of black-box algorithms for the task of weight learning in a SRL framework. We also introduced a projection for the weights in PSL which was necessary for an effective search. In this paper, we extend and generalize our previous work and introduce a new taxonomy of weight learning approaches based on search strategies. We generalize our projection space and introduce a new sampling strategy that is effective at approximating the projected weight space and is essential for the success of all search-based weight learning approaches. Further, we extend the scope of these search-based approaches for weight learning to MLNs. Finally, we perform a complete evaluation of different approaches over multiple datasets and metrics to show the effectiveness of the search-based approaches in both PSL and MLNs.

We introduce four search-based approaches that are all novel for the task of finding the best set of weights for a model (weight configuration) in weighted logic-based SRL frameworks. The key advantage of these approaches is that they directly optimize the chosen domain performance metric and, unlike other approaches, do not require re-derivation of the loss function for each metric. Our proposed approaches are based on black-box optimization methods used in learning hyperparameters in other machine learning approaches (Claesen and De Moor 2015; Bergstra et al. 2011). Our first two approaches *random grid search for weight learning* (RGS) and *continuous random search for weight learning* (CRS) are based on simple yet powerful search approaches popularly used in tuning hyperparameters of deep learning models (Bergstra and Bengio 2012). While RGS is simple and its effectiveness is determined by the specified grid, the effectiveness of CRS is determined by the new sampling space we introduce in this work. Our next approach, *Hyperband for weight learning* (HBWL), is based on the Hyperband algorithm (Li et al. 2018) that effectively distributes resources to perform efficient random search. Hyperband has been shown to efficiently allocate resources and maximize the search for the best solution. Finally, our fourth approach, *Bayesian optimization for weight learning* (BOWL), is based on Gaussian process regression (GPR) (Rasmussen and Williams 2005) in a Bayesian optimization (BO) (Mockus 1977) framework. BO is an effective approach for optimization of black-box functions (Lizotte et al. 2007; Martinez-Cantin et al. 2009; Srinivas et al. 2012; Brochu et al. 2010) and GPR is a non-parametric Bayesian approach that is often used to approximate arbitrary functions. GPRs have been used extensively for hyperparameter tuning in machine learning (Snoek et al. 2012).

In order to perform efficient and effective search of weights using these approaches, we introduce a new parameter search space, which we refer to as scaled space (SS), which is an accurate representation of the true weight space. We show that SS is both accurate and complete in representing the weights. Further, we also show that SS takes into account the impact of model instantiation (also referred to as *grounding*). As sampling from SS is challenging, we introduce an approximation of SS which enables us to efficiently sample weight configurations to perform search-based weight learning.

We develop all our search-based approaches for two powerful SRL frameworks, PSL and MLNs. We perform our empirical study on both PSL and MLNs to show the effectiveness of search-based approaches. While the applicability and effectiveness of our approaches are specifically shown for PSL and MLNs in this paper, we note that our approaches can be easily applied to other SRL frameworks that define a model through a set of logical rules.

Our contributions in this paper (extending from our previous work (Srinivasan et al. 2020b)) are as follows: (1) we generalize and introduce a new taxonomy of weight learning approaches in SRL frameworks by reformulating the problem of weight learning as a black-box optimization problem and introducing four search-based approaches, referred to as RGS, CRS , HBWL , and BOWL , to perform this optimization; (2) we introduce a new search space called the scaled space (SS) which we show is an accurate representation of the true weight space; (3) we introduce an approximation of SS which generalizes the search-based approaches and simplifies the process of sampling weight configurations in these methods; (4) we show that the search-based approaches are effective at learning weights in both PSL and MLNs and that these approaches outperform likelihood-based approaches on multiple datasets by up to 10%; and (5) finally, we show the scalability of search-based approaches and perform an elaborate set of experiments to show that, of the four approaches, BOWL is robust to initializations, acquisition function (process of choosing next best point, explained in Sect. 3.6.2), and the hyperparameter used in the sampling of weight configurations.

This paper is organized as follows. In Sect. 2, we provide a brief background on MLNs, PSL, black-box optimization, Bayesian optimization, and Gaussian process regression which are essential in understanding our approaches. Next, along with a motivating example, we introduce our four search-based weight learning approaches for MLNs and PSL in Sect. 3. Then, in Sect. 4, we introduce our novel projection and its approximation which are crucial to the success of our approaches. We then extend our approaches to accommodate for negative weights in Sect. 5, which is essential to fully support MLNs. In Sect. 6, we evaluate all of our approaches on several real-world datasets and metrics. Finally, in Sect. 7 we conclude and discuss potential future work.

2 Background

In this section, we first briefly review two well-known SRL frameworks, Markov logic networks (MLNs) (Richardson and Domingos 2006) and probabilistic soft logic (PSL) (Bach et al. 2017), along with two weight learning approaches for MLNs and three weight learning approaches for PSL. Finally, we present the required background on black-box optimization and Gaussian process regression (GPR) which serves as the foundation for our proposed approaches.

2.1 Statistical relational learning

SRL methods such as PSL and MLNs combine the power of probabilistic graphical models with weighted first-order logical rules to capture relational structure in any domain. A set of weighted first-order logical rules are instantiated with data D to generate a logical Markov random field (LMRF). This process is referred to as grounding and every instantiated rule consisting of only ground predicates is called a ground rule. Every ground predicate represents a random variable in the LMRF which may be observed (x) or unobserved (y) and a ground rule represents a clique potential $\boldsymbol{\phi}$ in the LMRF. Every potential in the LMRF is associated with a weight equal to the weight assigned to the logical rule from which it was instantiated. The weight of a logical rule represents the importance of the rule in the model. The weighted sum of potentials in the LMRF is referred to as the energy function $\mathbf{E}$. A LMRF can be formally defined as:

Definition 1 (*Logical Markov random fields*) Let $\mathbf{y} = \{y_1, y_2, ..., y_n\}$ be n unobserved random variables, $\mathbf{x} = \{x_1, x_2, ..., x_m\}$ be m observed variables or evidence, and $\boldsymbol{\phi} = \{\phi_1, \phi_2, ..., \phi_\iota\}$ be ι potentials describing different logical relations between variables. The output of a potential function $\phi_i(\mathbf{x}, \mathbf{y})$ is a real-valued scalar representing compliance of $\mathbf{x}$ and $\mathbf{y}$ with ϕ_i. Further, let $\mathbf{w} \in \{w_1, w_2, \ldots, w_l\}$ be a set of weights associated with each potential. The energy function of a LMRF is defined as:

$$\mathbf{E}(\mathbf{y}|\mathbf{x}) = \sum_{i=1}^{l} w_i \phi_i(\mathbf{x}, \mathbf{y}) s.t., \ \mathbf{y} \in \{0,1\}^n; \ \mathbf{x} \in \{0,1\}^m \tag{1}$$

and the conditional likelihood of a LMRF is defined as:

$$P(\mathbf{y}|\mathbf{x}, \mathbf{w}) = \frac{1}{Z(\mathbf{x})} exp(\hat{s} \cdot \mathbf{E}(\mathbf{y}|\mathbf{x})) \tag{2}$$

where $Z(\mathbf{x}) = \int_{\mathbf{y}} exp(\hat{s} \cdot E(\mathbf{y}|\mathbf{x}))$ is a normalization constant and $\hat{s} \in \{1, -1\}$ determines the sign in the exponent based on if the potential measures satisfaction or dissatisfaction.

To better understand LMRFs and the process of grounding, consider a simple collective labeling problem:

Example 1 Assume we have a set of users $\mathbf{U}$ and a label that can be either *true* or *false* associated with each user, such as if a user *Smokes* or not. The label for *Smokes* is observed for some users ($\mathbf{U}_o$) and unobserved for the rest ($\mathbf{U}_u$). The task is to infer the labels for $\mathbf{U}_u$. Let the input data be defined over users represented by the variables U, V that can take values in $\mathbf{U}$. The data includes a social network described by friendship links, *Friend*(U, V), and a local predictor that predicts if a user smokes or not based on the user's features, *LocalPredictor*(U). Below is a simple SRL model for collectively inferring labels:

$$\begin{aligned} w_1 &: LocalPredictor(U) \rightarrow Smokes(U) \\ w_2 &: Smokes(U) \wedge Friend(U, V) \rightarrow Smokes(V) \end{aligned}$$

where w_1 and w_2 are weights for the rules. The above model is then grounded with users in $\mathbf{U}$ to generate an LMRF. Each ground rule (e.g., w_1: $LocalPredictor(\}\}Bob'') \rightarrow Smokes(\}\}Bob''))$ generates a potential function ϕ_i. Each ground predicate created by instantiating the *Smokes* predicate with users in $\mathbf{U}_u$ generates a set of unobserved random variables $\mathbf{y}$ and the rest of the ground predicates generate a set of observed random variables $\mathbf{x}$. Figure 1 shows the resulting graphical model when instantiated over a small social network of 100 individuals. We can observe here that the graphical model generated, even with only 100 users and a simple model, tends to be large. This makes all aspects of SRL (in specific model instantiation, weight learning, and inference) particularly challenging (Getoor and Taskar 2007; De Raedt and Kersting 2011).

As seen in the example, all the potentials generated by a rule share the same weight. Equation 1 can therefore be alternatively written as $\mathbf{E}(\mathbf{y}|\mathbf{x}) = \sum_{i=1}^{r}[w_i \Phi_i(\mathbf{x}, \mathbf{y})]$, where r is the number of template rules, w_i represents the weight of the i^{th} rule, $\Phi_i = \sum_{j \in g_i} \phi_j(\mathbf{x}, \mathbf{y})$, and g_i is a set of ground rules generated by the i^{th} rule.

Inference in a LMRF is performed by finding a maximum aposteriori (MAP) estimate of the random variables $\mathbf{y}$ given evidence $\mathbf{x}$. This is performed by maximizing the density function or equivalently maximizing the energy function in Eq. 1. MAP inference is expressed as:

$$\underset{\mathbf{y}}{\operatorname{argmax}} \, P(\mathbf{y}|\mathbf{x}) = \underset{\mathbf{y}}{\operatorname{argmax}} \, \hat{s} \cdot \mathbf{E}(\mathbf{y}|\mathbf{x}) \tag{3}$$

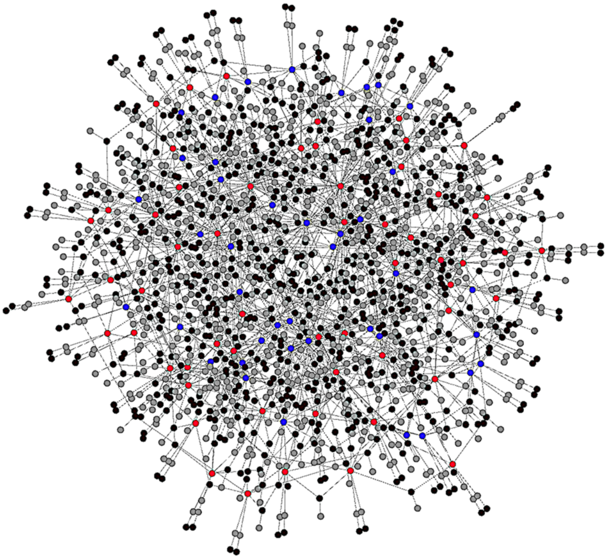


Fig. 1 Factor graph produced by grounding the example SRL model with synthetic data for 100 users. The blue nodes are users who smoke, red nodes are users who do not smoke, grey nodes are the rest of the grounded atoms and the black nodes are potentials. Here we see that the resulting factor graph, even for this simple case, is large, complex, and highly connected

MLN and PSL make specific choices in LMRF to generate a Markov network and a hinge-loss Markov random field (HL-MRF), respectively. Next, we first describe the choices made by MLNs and then PSL, along with weight learning approaches commonly used by both.

2.2 Markov logic networks

MLNs (Richardson and Domingos 2006) use boolean logic and use discrete random variables in the LMRF. Potential functions $\boldsymbol{\phi}$ are indicator functions that have a value of one if the ground rule is satisfied and zero otherwise. A potential in MLN is of the form:

$$\phi_i(\mathbf{x}, \mathbf{y}) = n_i(\mathbf{x}, \mathbf{y}) \tag{4}$$

where $n_i(\cdot, \cdot)$ is the satisfaction of the i^{th} ground rule. Since the potentials measure the satisfaction of the ground rules, the $\hat{s}$ is set to one and the MAP inference objective in MLNs can be expressed as a weighted count of the number of satisfied ground rules:

$$\underset{\mathbf{y}}{\operatorname{argmax}}\, P(\mathbf{y}|\mathbf{x}) = \underset{\mathbf{y}}{\operatorname{argmax}}\, \mathbf{E}(\mathbf{y}|\mathbf{x}) \tag{5}$$

$$= \underset{\mathbf{y}}{\operatorname{argmax}} \sum_{i=1}^{\iota} w_i n_i(\mathbf{x}, \mathbf{y}) \tag{6}$$

Many implementations of MLNs exist (Noessner et al. 2013; Shavlik and Natarajan 2009; Niu et al. 2011; Venugopal et al. 2016; Islam et al. 2018). In this paper we are interested in understanding how different weight learning techniques effect the quality of predictions, rather than the efficiency of inference. However, it is important to note, that due to the non-convexity of MAP inference in MLNs a global solution is not guaranteed and the quality of MAP estimates can have consequences on the effectiveness of weight learning techniques. In Sect. 6, we use Tuffy (Niu et al. 2011) for empirical evaluations as this framework has been widely applied and validated as state-of-the-art. MAP inference in Tuffy is performed using the WalkSAT algorithm (Kautz et al. 1996; Niu et al. 2011). WalkSAT proceeds by iteratively flipping atom values that would result in a more satisfied world.

Here we focus on the discriminative framework of weight learning for MLNs where a training dataset is partitioned into an unobserved random variable set $\mathbf{y}$ and a non-empty evidence set $\mathbf{x}$. Richardson and Domingos (2006) originally proposed a generative learning framework where the evidence set is empty, however, the discriminative approach consistently outperformed this approach (Singla and Domingos 2005).

2.2.1 Maximum-likelihood estimation

Maximum-likelihood estimation (MLE) based approaches for weight learning minimize the negative conditional log-likelihood (CLL) of a training dataset. Lowd and Domingos (2007) discuss multiple methods for solving the MLE problem including gradient descent, contrastive divergence, diagonal newton, and conjugate gradient. Recent efforts for scaling discriminative MLE weight learning reduce the search space (Farabi et al. 2018) by leveraging symmetries to speed up (Ahmadi et al. 2012; Van Haaren et al. 2015) or approximate the inference subproblem (Sarkhel et al. 2016; Das et al. 2016, 2019). For this discussion and for later experiments, we use the diagonal Newton method (DN) as the representative MLE weight learner as it was found to be the among the most effective for MLE weight learning in MLNs (Lowd and Domingos 2007) and is the default optimizer for Tuffy (Niu et al. 2011).

A Newton-based approach reaches the global minimum by multiplying the gradient with the inverse of the Hessian at every iteration:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \mathbf{H}^{-1}\mathbf{g}$$

where $\mathbf{H}$ is the Hessian, $\mathbf{g}$ is the gradient w.r.t. $\mathbf{w}$, and t is the iteration number. Since computing the Hessian can be expensive and infeasible, a diagonal Newton method is used as an approximation for the Hessian. The Hessian of the negative CLL is the covariance matrix of the CLL, this is approximated through samples generated using MC-SAT (Poon and Domingos 2006). The final update for the weight of a logical rule at each iteration is given by:

$$w_i = w_i - \alpha \frac{\mathbb{E}_{\mathbf{w}}(N_i) - N_i}{\mathbb{E}_{\mathbf{w}}(N_i^2) - (\mathbb{E}_{\mathbf{w}}(N_i))^2} \tag{7}$$

where $N_i = \sum_{j \in g_i} n_j(\mathbf{x}, \mathbf{y})$, g_i is a set of ground rules generated by the i^{th} rule, $\mathbb{E}_{\mathbf{w}}$ is expectation w.r.t. the weights and α is the step size.

2.2.2 Large-margin estimation

Large-margin estimation (LME), also referred to as max-margin estimation, focuses on maximizing the MAP state rather than producing accurate probabilistic models (Huynh and Mooney 2009, 2010). The intuition motivating large-margin weight learning is that the ground-truth state $\mathbf{y}$ should have energy lower than any alternate state $\tilde{\mathbf{y}}$ by a wide margin that is defined by a loss function L. The objective function to find the optimal set of weights is given by:

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} \frac{1}{2}||\mathbf{w}||^2 + C\xi$$
$$s.t. \forall \tilde{\mathbf{y}} : \mathbf{w}^T(\Phi(\tilde{\mathbf{y}}, \mathbf{x}) - \Phi(\mathbf{y}, \mathbf{x})) \leq -L(\mathbf{y}, \tilde{\mathbf{y}}) + \xi \tag{8}$$

where L is a loss function such as the L1 distance between $\mathbf{y}$ and $\tilde{\mathbf{y}}$, ξ is a slack variable, and $C > 0$ is a user-specified parameter. Equation 8 is then solved by performing a cutting-plane approach for structural support vector machines (Joachims et al. 2009) or Frank-Wolfe optimization (Lacoste-Julien et al. 2013). Both of the mentioned methods of optimization require repeatedly solving two argmax subproblems:

1. Prediction: $\operatorname{argmax}_{\mathbf{y}} \mathbf{w}^T\Phi(\mathbf{y}, \mathbf{x})$
2. Separation: $\operatorname{argmax}_{\tilde{\mathbf{y}}} L(\mathbf{y}, \tilde{\mathbf{y}}) + \mathbf{w}^T\Phi(\tilde{\mathbf{y}}, \mathbf{x})$

An approximate solution to the prediction subproblem can be found using any MAP inference technique, e.g., WalkSAT. Then separation can be viewed as augmented MAP inference and is typically performed using the same solver. For instance, when L is the L1 distance between $\mathbf{y}$ and $\tilde{\mathbf{y}}$, then L can be represented by n new potentials $\{\phi_{t+1}, \phi_{t+2}, ..., \phi_{t+n}\}$ where ϕ_{t+i} is the 0, 1 indicator function for whether y_i equals $\tilde{y}_i$.

2.3 Probabilistic soft logic

PSL (Bach et al. 2017), unlike MLNs, uses soft logic and relaxes random variables to be in the range [0, 1]. Specifically, PSL uses Łukasiewicz logic to generate potentials which take the form of hinges. For example, $a \rightarrow b$ corresponds to the hinge potential $\max(a - b, 0)$, and $a \wedge b$ corresponds to $\max(a + b - 1, 0)$ (see Bach et al. (2017) for full details). Notice that this interpretation is equivalent to Boolean logic when the random variables are 0 or 1, and for other values, it is a measure of truth or satisfaction. From this perspective, a potential function $\boldsymbol{\phi}$ is the distance to satisfaction of a ground rule. A hinge potential in PSL is of the form:

$$\phi_i(\mathbf{x}, \mathbf{y}) = \max(\ell_i(\mathbf{x}, \mathbf{y}), 0)^{d_i} s.t., \ \mathbf{0} \leq \mathbf{y} \leq \mathbf{1} \ ; \mathbf{0} \leq \mathbf{x} \leq \mathbf{1} \tag{9}$$

where ℓ_i is a linear function and $d_i \in \{1, 2\}$ provides a choice of two different loss functions, $d_i = 1$ (i.e., linear) and $d_i = 2$ (i.e, quadratic). Since the potential functions in PSL measure distance to satisfaction, $\hat{s}$ in (2) is set to -1. Further, weights in PSL are positive and real, i.e., $w_i \in \mathbb{R}^+$. The MAP inference in PSL is expressed as:

$$\underset{\mathbf{y}}{\operatorname{argmax}} P(\mathbf{y}|\mathbf{x}) = \underset{\mathbf{y}}{\operatorname{argmin}} \mathbf{E}(\mathbf{y}|\mathbf{x}) \tag{10}$$

A key advantage of using PSL is that the inference objective is convex. This enables the use of efficient convex optimization procedures, such as alternating direction method of multipliers (ADMM) (Boyd et al. 2011). Hence, given known weights, inference in PSL can be performed at scale enabling predictions on very large real-world datasets (Srinivasan et al. 2020a). Unfortunately, the task of learning the rule weights from training data is not as efficient; although, as we will see in Sect. 3, having tractable MAP inference is still helpful.

There are three primary approaches used to perform weight learning in PSL: Maximum Likelihood Estimation (MLE), Maximum Pseudolikelihood Estimation (MPLE), and Large-Margin Estimation (LME) (Bach et al. 2013).

2.3.1 Maximum likelihood estimation (MLE)

This approach maximizes the log-likelihood function with respect to the weights of the rules based on the training data. The partial derivative of the log of the likelihood function given in Eq. 2 for PSL with respect to w_q for $q \in \{1, \dots, r\}$ is:

$$\frac{\partial logP(\mathbf{y}|\mathbf{x})}{\partial w_q} = \mathbb{E}_{\mathbf{w}}\left[\Phi_q(\mathbf{x}, \mathbf{y})\right] - \Phi_q(\mathbf{x}, \mathbf{y}) \tag{11}$$

where $\mathbf{w} = \{w_1, ..., w_r\}$ and $\mathbb{E}_{\mathbf{w}}$ is expectation w.r.t. the weights. It is infeasible to compute the expectation, hence to make the learning tractable a MAP approximation is used that replaces the expectation in the gradient with the corresponding values in the MAP state. This approach is a structured variant of the voted perceptron algorithm (Collins 2002).

2.3.2 Maximum pseudolikelihood estimation (MPLE)

An alternative approach that maximizes the pseudolikelihood function is given by:

$$P^*(\mathbf{y}|\mathbf{x}) = \prod_{i=1}^{n} P^*(y_i|MB(y_i), \mathbf{x}) \tag{12}$$

where n is the number of random variables and $MB(y_i)$ is the Markov blanket of y_i (Besag 1975). Equation 12 is maximized using a gradient ascent based approach and the derivative of the log-pseudolikelihood function with respect to w_q is given by:

$$\frac{\partial logP^*(\mathbf{y}|\mathbf{x})}{\partial w_q} = \sum_{i=1}^{n} \mathbb{E}_{y_i|MB}\left[\sum_{j \in g_q : i \in \phi_j} \phi_j(\mathbf{x}, \mathbf{y})\right] - \Phi_q(\mathbf{x}, \mathbf{y}) \tag{13}$$

where $i \in \phi_j$ implies that variable y_i participates in the potential ϕ_j. Using a Monte Carlo approach this derivative can be computed in linear time in the size of $\mathbf{y}$.

2.3.3 Large-margin estimation (LME)

The LME formulation and high-level approach for optimization in PSL is the same as for MLNs. The difference lies in the fact that random variables are continuous in PSL and this results in an infinite number of constraints in the LME formulation. Further the prediction and separation subproblems are:

1. Prediction: $\text{argmin}_{\mathbf{y}}\ \mathbf{w}^T\Phi(\mathbf{y}, \mathbf{x})$
2. Separation: $\text{argmin}_{\tilde{\mathbf{y}}}\ \mathbf{w}^T\Phi(\tilde{\mathbf{y}}, \mathbf{x}) - L(\mathbf{y}, \tilde{\mathbf{y}})$

The solver used for the prediction and separation subproblems is one of the optimization algorithms implemented for PSL inference, e.g., ADMM. Notice the separation subproblem objective is a difference of convex functions and is hence non-convex. Thus a local optimal solution will be found.

In our empirical evaluation, we compare our approach with the above discussed weight learning approaches for PSL and MLNs. Next we briefly discuss black-box optimization, Bayesian optimization, and Gaussian process on which we base our approaches.

2.4 Black-box optimization

Black-box optimization is a well studied technique, especially in the context of hyperparameter tuning (Bergstra and Bengio 2012; Shahriari et al. 2016).

Definition 2 (*Black-box optimization*) Given a black-box function $\gamma(\tilde{\mathbf{x}}) : \mathbb{R}^d \rightarrow \mathbb{R}$, where d is the input dimension, the task of finding an $\tilde{\mathbf{x}}$ that yields the optimal value for $\gamma(\tilde{\mathbf{x}})$ is called *black-box optimization*.

The goal of black-box optimization is to find the best possible value for $\tilde{\mathbf{x}}$ that optimizes the function $\gamma(\tilde{\mathbf{x}})$. This is often paired with a predefined constraint on usable resources (generally number of epochs or time). While this process can be embarrassingly parallel for some approaches, the general strategy can be defined sequentially. The basic approach for black-box optimization is to define a search space, choose a point $\tilde{\mathbf{x}}$, evaluate $\gamma(\tilde{\mathbf{x}})$ to update a model, and repeat this process until the resources have been exhausted. In the end, the $\tilde{\mathbf{x}}$ with the best $\gamma(\tilde{\mathbf{x}})$ is returned. The procedure is summarized in Algorithm 1. While the process of selection of $\tilde{\mathbf{x}}$ and evaluation of $\gamma(\tilde{\mathbf{x}})$ on different points can be simple and made to run in parallel for algorithms like RGS, CRS , and HBWL introduced in this paper (Sect. 3), other algorithms like BOWL (also introduced in this paper) use the Bayesian optimization (BO) framework and employ a Gaussian process to approximate γ and assume a serial setup to ensure optimal selection of the next point on which to evaluate.

In the context of BO, various strategies have been proposed to choose the next point to evaluate given the previous evaluations (Srinivas et al. 2010; Kushner 1964; Mockus 1977; Thompson 1933). Each strategy is encoded through an acquisition function α. The objective of these strategies is to minimize the number of epochs required to find the best solution. A simple black-box Bayesian approach iteratively obtains a point to explore from the acquisition function α using the prior distribution; then the function γ is evaluated to obtain a new outcome at that point which is then used to update the posterior. Gaussian process regression (GPR) is a non-parametric Bayesian approach which is effective in performing black-box optimization in a BO framework.

Algorithm 1: Black-box optimization via search

```
Result: $\tilde{\mathbf{x}}^*$ : point with the best value for $\gamma(\cdot)$
$\tilde{\mathbf{X}}$ = search space;
while stopping criteria not met do
    choose a $\tilde{\mathbf{x}} \in \tilde{\mathbf{X}}$;
    update the model of choice with $(\tilde{\mathbf{x}}, \gamma(\tilde{\mathbf{x}}))$;
    update $\tilde{\mathbf{x}}^*$ if current $\gamma(\tilde{\mathbf{x}})$ is better than previous value or based on the model
end
```

2.4.1 Gaussian process regression

A Gaussian process (GP) (Rasmussen and Williams 2005) describes a distribution over a function space and is fully characterized by a mean function μ_0 and a covariance matrix $\mathbf{K} = \left[K_{i,j}\right]$. More formally, consider a finite set of s inputs, $\tilde{\mathbf{X}} = \tilde{\mathbf{x}}_{1:s}$. Then, for any input $\tilde{\mathbf{x}}_i \in \tilde{\mathbf{X}}$, let $g_i = \gamma(\tilde{\mathbf{x}}_i)$ represent the function γ evaluated at $\tilde{\mathbf{x}}_i$. In GPR, we assume a prior distribution over $\mathbf{g} = g_{1:s}$ that is jointly Gaussian, specifically, the prior distribution over $\mathbf{g}$ can be expressed as $\mathbf{g} \sim \mathcal{N}(\mathbf{m}, \mathbf{K})$, where $m_i = \mu_0(\tilde{\mathbf{x}}_i)$. To allow for a more general setting with noisy function evaluations, let $\tilde{y}_i$ be the noisy output of the function $\gamma(\tilde{\mathbf{x}}_i)$, i.e., $\tilde{y}_i = \gamma(\tilde{\mathbf{x}}_i) + \epsilon$, where ϵ is Gaussian $\mathcal{N}(0, \sigma)$ noise. As the prior and noise are both Gaussian, if the vector $\tilde{\mathbf{y}} = [\tilde{y}_i]$ of function evaluations at a set of points $\tilde{\mathbf{X}}$ is observed, then the likelihood of the noisy function evaluation at new point $\tilde{\mathbf{x}}_{s+1}$ is also jointly Gaussian. Hence, the posterior distribution over $\tilde{y}_{s+1}$ given $\tilde{\mathbf{y}}$ can be expressed as $\tilde{y}_{s+1}|\tilde{\mathbf{y}} \sim \mathcal{N}(\mu_s(\tilde{\mathbf{x}}_{s+1}), \sigma_s(\tilde{\mathbf{x}}_{s+1}))$ where

$$\mu_s(\tilde{\mathbf{x}}_{s+1}) = \mu_0(\tilde{\mathbf{x}}_{s+1}) + K_{s+1}^T(\mathbf{K} + \sigma^2\mathbf{I})^{-1}(y - m)$$
$$\sigma_s(\tilde{\mathbf{x}}_{s+1}) = K_{s+1,s+1} - K_{s+1}^T(\mathbf{K} + \sigma^2\mathbf{I})^{-1}K_{s+1}$$

and K_{s+1} is the vector of covariances between the new input $\tilde{\mathbf{x}}_{s+1}$ and every observed input $\tilde{\mathbf{x}}_i \in \tilde{\mathbf{X}}$. Assuming one has a method for computing covariances between any two points in the input space, then the posterior mean and variance of the noisy output of the function γ at any point can be computed using the above expressions. In GPR, a kernel function, $k(\cdot,\cdot)$, is used, and it plays the crucial role of defining the covariance between points in the input space, i.e., $K_{i,j} = \mathbb{E}\left[(\gamma(\tilde{\mathbf{x}}_i) - \mu_0(\tilde{\mathbf{x}}_i))(\gamma(\tilde{\mathbf{x}}_j) - \mu_0(\tilde{\mathbf{x}}_j))\right] = k(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j)$. Consequentially, k encodes assumptions about the function the GP is defining a distribution over. The choice of kernel function is often the key to finding the best approximation of the true function.

2.4.2 Kernel functions

In its most generic form, a kernel function, $k(\cdot,\cdot)$, is any mapping of two inputs from a space $\mathcal{X}$ to $\mathbb{R}$. For a valid GP, the choice of kernel function must correspond to an inner product in some inner product space. Formally, for any two inputs $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$, $k(\cdot,\cdot)$ is equal to the inner product of the inputs after being mapped to an inner product space $\mathcal{H}$ via a transformation $\Psi : \mathcal{X} \rightarrow \mathcal{H}$, i.e.,

$$k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}, \qquad (\mathbf{x}, \mathbf{x}') \mapsto k(\mathbf{x}, \mathbf{x}')$$

where

$$k(\mathbf{x}, \mathbf{x}') = \langle \Psi(\mathbf{x}), \Psi(\mathbf{x}') \rangle_{\mathcal{H}} \tag{14}$$

Requiring that a kernel, $k(\cdot,\cdot)$, corresponds to an inner product ensures that the matrix defined by the inputs $\tilde{\mathbf{x}}_1, \cdots, \tilde{\mathbf{x}}_s \in \mathcal{X}$ and the kernel, namely the Gram matrix $\mathbf{K} = [K_{i,j}]$ where $K_{i,j} = k(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j)$, is positive semi-definite, and hence can be used as a covariance matrix. Kernels with this property are referred to as positive semi-definite kernels or covariance functions.

There is a suite of positive semi-definite kernels that have been proposed in the literature, each equipped with different properties that may be more or less suited for a problem domain (Genton 2001; Rasmussen and Williams 2005; Schölkopf and Smola 2002). For instance, the stationary class of kernels, which includes the widely used squared exponential (employed in later Sect. 3.6) and Matérn class of kernels (Matérn 1960) assumes the covariance between inputs is translation invariant. More formally, stationary kernels assume that for some vector $v \in \mathcal{X}$, $k(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j) = k(\tilde{\mathbf{x}}_i + v, \tilde{\mathbf{x}}_j + v)$. Another common assumption is that of rotational invariance. This property is held by a class of kernels called dot product kernels, which only depend on the inputs $\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j$ through $\tilde{\mathbf{x}}_i \cdot \tilde{\mathbf{x}}_j$. One example of a dot product kernel is the inhomogeneous polynomial kernel $k(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j) = (\sigma_0^2 + \tilde{\mathbf{x}}_i \cdot \tilde{\mathbf{x}}_j)$.

Commonly, a domain will not satisfy the assumptions made by a standard kernel. Rather, it is necessary to design a specialized covariance function for the setting. At a high level, two approaches for designing covariance functions that meet a set of desired properties for a problem are: (1) proposing a novel positive semi-definite kernel function or (2) creating a new kernel from existing covariance functions through positive semi-definite preserving operations (Genton 2001). In this paper, we employ the second approach. Specifically, we leverage the fact that a covariance function defined over a real-valued projection of the input space is also a covariance function. More concretely, suppose $\mathcal{E}$ is an $\mathbb{R}^p$ -valued function on $\mathcal{X}$ and $\tilde{k}$ is a covariance function on $\mathbb{R}^p \times \mathbb{R}^p$, then

$$k(\mathbf{x}, \mathbf{x}') = \tilde{k}(\mathcal{E}(\mathbf{x}), \mathcal{E}(\mathbf{x}')) \tag{15}$$

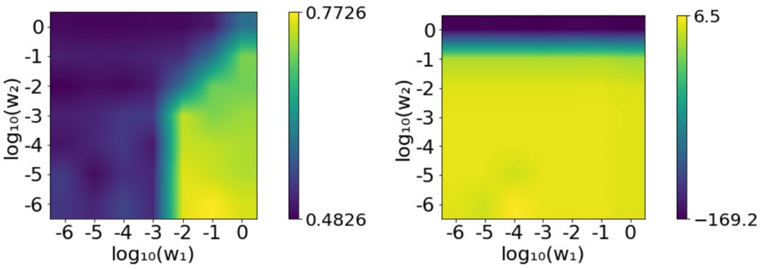
is a covariance function. The projection for the input space, $\mathcal{E}$, is designed to meet the assumptions made by the kernel, $\tilde{k}$, that is applied to the output. For instance, as previously mentioned, stationary kernels assume the covariance between two inputs is strictly a function of their Euclidean distance. While it is common for this assumption to be incorrect, stationarity can be achieved via a projection that warps the input space to more closely meet this property. If $\tilde{k}(\mathbf{x}, \mathbf{x}')$ is a dot product kernel, a different projection would be necessary to satisfy the rotational invariance assumption. In Sect. 4.2 we propose a projection over the weight space that allows the use of stationary kernels for our weight learning application.

3 Search-based approaches for weight learning

As mentioned earlier, commonly used approaches for rule weight learning in SRL are generally based on maximizing a likelihood function. In this section, we first give a motivating example that highlights the issues with likelihood-based approaches and then we propose four search-based approaches to learn weights in SRL frameworks. Note that, for conciseness, we assume weights are constrained to be non-negative and introduce a simple method for adapting search-based learning algorithms to support negative weights in Sect. 5.

3.1 Motivating example

Consider our simple collective classification model, Example 1, applied to a toy dataset with 100 users. Figure 2 shows the performance of the model in PSL as we vary the rule weights logarithmically from 10^{-6} to 1.0. Figure 2a shows AUROC and Fig. 2b shows the log-likelihood of the model. Lighter shades (yellow) represent a high value and darker shades (dark blue) represent a low value. We observe that the AUROC is maximized when the first rule's weight is 0.1 and the second rule's weight is 10^{-6}. However, the likelihood is not maximized at these weights. For this simple model and dataset, we



(a) AUROC **(b)** log-likelihood

Fig. 2 Heat map for AUROC and log-likelihood for the model in Example 1. The lighter color indicates higher values and higher values are desired for both metrics

observe that the likelihood is not well correlated with the AUROC. While this behavior is shown using PSL, a similar outcome can be observed when using MLN models as well.

3.2 Problem definition

Consider a LMRF-based SRL model with r rules where each rule $i \in \{1 \dots r\}$ is associated with a non-negative weight $w_i \in \mathbb{R}_+$. Grounding all the rules with data D yields a set of m observed random variables $\mathbf{x} = \{x_1, \dots, x_m\}$, n unobserved random variables $\mathbf{y} = \{y_1, \dots, y_n\}$, and ι potentials $\boldsymbol{\phi} = \{\phi_1, \phi_2, \dots, \phi_\iota\}$. All unknown random variables are associated with corresponding ground truth $\mathbf{y}^* = \{y_1^*, \dots, y_n^*\}$ used to compute evaluation metrics. Let $\mathbf{w} = \{w_1, \dots, w_r\}$ be the vector representing the set of rule weights, i.e., the weight configuration. Next, let $\omega(\mathbf{y}, \mathbf{y}^*) : (\mathbf{y}, \mathbf{y}^*) \rightarrow \mathbb{R}$ be a problem-specific evaluation metric (e.g., accuracy, AUROC, or F-measure) and let $\gamma(\mathbf{w}) : \mathbf{w} \mapsto \omega(\operatorname{argmax}_y P_\mathbf{w}(\mathbf{y}|\mathbf{x}), \mathbf{y}^*)$ where $P_\mathbf{w}(\mathbf{y}|\mathbf{x})$ is the distribution function of the SRL model (2) parameterized by $\mathbf{w}$. Then the objective of search-based weight learning can be expressed as finding the set of weights that maximize the function γ which represents the true metric function ω, i.e., $\operatorname{argmax}_\mathbf{w} \gamma(\mathbf{w})$.

In order for the search-based weight learning objective function, $\gamma(\mathbf{w})$, to be well defined, it is necessary that a unique solution to the inference problem exists and is found exactly. This condition is satisfied for L_2 regularized PSL MAP inference as the problem becomes strongly convex and obtaining a high accuracy solution, i.e. a solution that is nearly the global optimal, is practical to find. For MLNs it is usually not true that a unique optimal solution exists, and even if it does, MAP inference for MLNs is NP-hard and only an approximate solution is typically found. Despite this, in Sect. 6 we show that search-based weight learning methods for MLNs still tend to perform well in practice.

In order to solve the search based optimization problem $\operatorname{argmax}_\mathbf{w}(\gamma(\mathbf{w}))$, we introduce four approaches based on hyperparameter search methods from other areas of machine learning. The first two approaches are based on random grid search and continuous search used in deep learning (Bergstra and Bengio 2012), the next approach is based on the Hyperband algorithm used in statistical machine learning (Li et al. 2018), and the last is based on BO with GPR used for hyperparameter tuning in deep learning (Snoek et al. 2012).

3.3 Random grid search for weight learning

A straightforward search-based approach to weight learning is an exhaustive exploration over a set of weight configurations $\mathbf{W}$ generated through a user-specified grid of weights. The user-specified grid to generate weight configurations $\mathbf{W}$ is typically constructed by specifying a finite collection of v values $V = \{V_0, \cdots, V_v\}$ that can be assigned as weights for each of the rules, e.g., $V = \{0.01, 0.1, 1.0\}$. If a model contains r rules, then we can define $\mathbf{W}$ to be the r-ary Cartesian product of V, $\mathbf{W} = V \times \cdots \times V$, defining a grid with the intersections representing different weight configurations. Then, for each configuration $\mathbf{w} \in \tilde{\mathbf{W}}$, $\gamma(\mathbf{w})$ is evaluated after performing MAP inference in the SRL model. Finally, the weight configuration with the highest $\gamma(\mathbf{w})$ is selected.

However, a comprehensive grid search is usually infeasible due to the combinatorial explosion in the size of the grid; if a model contains r rules where each rule can take on one of v possible values, then $|\mathbf{W}| = v^r$. Thus, to make the approach tractable (as mentioned in Algorithm 2), we uniformly draw t unique samples from $\mathbf{W}$ to stay within an established budget of resources; this approach is referred to as *random grid search for weight learning* (RGS). It is important to ensure that the resources are used to evaluate distinct weight configurations, therefore at the time of sampling we ensure that every weight configuration chosen has a possibility of yielding a distinct solution. In Sect. 4, we show how two weight configurations that look distinct can yield the same solution at the time of MAP inference (and hence for the value of γ as well) and how we can identify and avoid wasting resources on such inherently identical weight configurations. Therefore, to ensure uniqueness of weight configurations explored, we keep track of a set $\mathbf{W}_{explored}$ which contains all the weight configurations explored so far and the configurations that are inherently the same as an explored weight configuration and sample new weight configuration from $\mathbf{W} \setminus \mathbf{W}_{explored}$.

Algorithm 2: Random grid search for weight learning

```
Result: $\mathbf{w}^*$ : weight configuration with best evaluation metric among samples
$\mathbf{W}$ = set of weight configurations from the user-defined weight grid;
$t$ = maximum number of weight configurations to explore;
$\mathbf{W}_{explored}$ = weight configurations explored so far;
for $iter \in \{1, \ldots, t\}$ do
    $\mathbf{w}_{next} = Random(\mathbf{W} \setminus \mathbf{W}_{explored})$;
    perform MAP inference to compute $\gamma(\mathbf{w}_{next})$;
    $\forall \dot{\mathbf{w}} \in \mathbf{W}$ such that $\gamma(\dot{\mathbf{w}}) = \gamma(\mathbf{w}_{next})$, add to $\mathbf{W}_{explored}$ ;
    if $\gamma(\mathbf{w}_{next}) > \gamma(\mathbf{w}^*)$ then
        $\mathbf{w}^* = \mathbf{w}_{next}$;
    end
end
```

3.4 Continuous random search for weight learning

A primary drawback of RGS is the need to define a grid over the space which captures weights that will lead to a good model. While specifying a grid might seem straightforward, several unique properties of the weight space makes the process of specifying the right grid non-trivial (details in Sect. 4). Further, specifying grids can easily result in unexpected biases. For instance, one may be tempted to simply define a grid of evenly spaced points in a unit hypercube. However, this leads to a sampling bias towards configurations with moderate ratios which might not be ideal (more details in Sect. 4.4).

Continuous random search for weight learning (CRS) is similar to RGS in that, rather than exploring the entire space, t weight configurations are chosen for evaluation and the highest performing configuration is returned. The difference is that CRS does not define a discrete grid of weights but samples continuously from the search space. Therefore, it is crucial for the search space to be an accurate representation of the true weight space. In this approach (as mentioned in Algorithm 3), we sample t weight configurations from a r dimensional Dirichlet distribution and return the weight configuration with the best value obtained for the γ function. In Sect. 4.4, we discuss in detail on why using a Dirichlet distribution to sample weights is an appropriate choice. For CRS the Dirichlet distribution is parametrized by a r-dimensional hyperparameter $\mathrm{A} \in \mathbb{R}^r_+$, which can be tuned to obtain the best approximation of the space based on the application and prior knowledge.

Algorithm 3: Continuous random search for weight learning

```
Result: $\mathbf{w}^*$ : weight configuration with best evaluation metric among samples
$t$ = maximum number of weight configurations to explore;
$\mathbf{W}_{explore} \sim Dirichlet(\mathrm{A})^t$, $t$ distinct samples from a Dirichlet distribution;
for $\mathbf{w}_{next} \in \mathbf{W}_{explore}$ do
    perform MAP inference to compute $\gamma(\mathbf{w}_{next})$;
    if $\gamma(\mathbf{w}_{next}) > \gamma(\mathbf{w}^*)$ then
        $\mathbf{w}^* = \mathbf{w}_{next}$;
    end
end
```

3.5 Hyperband for weight learning

So far, the search-based methods we have discussed iteratively select a set of t weight configurations to explore, $\mathbf{W}_{explore}$, from a set of weight configurations, $\mathbf{W}$ and then run inference until completion for each $\mathbf{w}_i \in \mathbf{W}_{explore}$ in order to calculate $\gamma(\mathbf{w}_i)$. Then the weight configuration $\mathbf{w}^* = \mathrm{argmax}_{\mathbf{w}_i \in \mathbf{W}_{explore}} \gamma(\mathbf{w}_i)$ is chosen as the model weights. Ideally, in these methods, we want t to be as large as possible, as increasing the number of weight configurations explored can potentially improve the $\gamma(\mathbf{w}^*)$ obtained. However, it is generally infeasible to have a large t due to limited resources. In order to maximize our gain with the limited resources, we make use of a practical observation that the weight configurations which initially show a slow rate of improvement during inference will tend to converge to a poor $\gamma(\cdot)$ evaluation. For instance, let $\mathbf{w}_1$ and $\mathbf{w}_2$ be two weight configurations for a SRL model. If MAP inference for both the weight configurations takes $\hat{t}$ iterations to converge to the final solution and results in $\gamma(\mathbf{w}_1) > \gamma(\mathbf{w}_2)$, then it is highly likely that the $\gamma(\cdot)$ computed by interrupting the MAP inference at $\frac{\hat{t}}{\hat{s}}$ number of iterations, where $\hat{s} > 1$, will still result in

$\gamma(\mathbf{w}_1) > \gamma(\mathbf{w}_2)$. Therefore, running MAP inference to convergence for all weight configurations could be wasteful. By early termination of unpromising weight configurations a larger number of configurations can be explored resulting in a better overall solution. This idea has been exploited in other areas of machine learning to tune hyperparameters and is referred to as *Hyperband* (Li et al. 2018). Here, we adapt this approach in the context of weight learning and refer to it as *Hyperband for weight learning* (HBWL).

HBWL allows for more exploration while still operating within a budget (generally time or iterations) through adaptive resource allocation and early-stopping. To understand how the algorithm operates, we will first describe *SuccessiveHalving*, a critical subroutine of HBWL , and then describe how SuccessiveHalving is used in HBWL . SuccessiveHalving (as mentioned in Algorithm 4) requires two input parameters, namely t the total number of weight configurations we wish to explore and B the total number of iterations for MAP inference.[1] Initially t configurations $\mathbf{W}_{explore} = \{\mathbf{w}_1, \cdots, \mathbf{w}_t\}$ are sampled from the search space $\mathbf{W}$. Then, SuccessiveHalving proceeds in rounds. At the start of each round, a fraction of the budget ($b = \frac{B}{t}$) is allocated to the t weight configurations to perform MAP inference and compute $\gamma'(\cdot, b)$ where $\gamma'(\cdot, b)$ is the evaluation metric computed after b iterations of MAP inference. Finally, the weight configurations are ranked based on $\gamma'(\cdot, b)$, the bottom half of the weight configurations are removed, and $|\mathbf{W}_{explore}|$ is reduced to $\frac{t}{\eta}$ where $\eta > 1$ is the proportion of configurations to be removed (for classic SuccessiveHalving $\eta = 2$). This process is repeated for multiple rounds until only one weight configuration remains which is chosen as the $\mathbf{w}^*$.

The hyperparameters B and t of SuccessiveHalving can trade-off between having a large number of weight configurations with a small amount of resource allocated to each configuration (a.k.a. exploration), or a small number of weight configurations with a large amount of resource allocated to each weight configuration (a.k.a. exploitation). The best trade-off between exploration and exploitation is typically unknown. HBWL (as mentioned in Algorithm 5) extends SuccessiveHalving by trying several possible values for the $\frac{t}{B}$ ratio to choose the best explore exploit trade-off. The possible values for $\frac{t}{B}$ are constructed strategically from the two user provided parameters for HBWL , $\hat{R}$, the maximum amount of resource that can be allocated to a single weight configuration, and $\eta \in (1, \infty)$, the proportion of weight configurations to be removed in each round of SuccessiveHalving. Each complete execution of SuccesiveHalving in HBWL is referred to as a bracket. Every bracket is parameterized by the values (t, B, s) that are constructed uniquely for each bracket using $\hat{R}$ and η, where s is the number of initial configurations being tested in that bracket. HBWL chooses a bracket size of $s = \lfloor \log_\eta(\hat{R}) \rfloor + 1]$ and decrements it every round until one. Finally, the weight configuration with the best value for function γ is returned.

[1] Note that, for simplicity, in the algorithm we overload B to be number of iterations in MAP inference in the first round instead of maximum number of iterations. The maximum number of iterations in Algorithm 4 is $B\eta^{log_\eta(t)}$, where η is a parameter defined in HBWL which represents the proportion of weights removed every round in SuccessiveHalving.

```
Algorithm 4: SuccessiveHalving
Result: $\mathbf{w}^*, \gamma^*$ : weight configuration with the best evaluation metric and the metric value $\gamma(\cdot)$
$B =$ number of iterations in MAP inference for first round;
$\eta =$ the proportion of weight configurations to be removed in each round;
$\mathbf{W}_{explore} =$ weight configurations to be evaluated;
$t = |\mathbf{W}_{explore}|$;
$b = B$ ;
$iter = 0$;
while $t > 1$ do
    $iter$++ ;
    $\forall \mathbf{w} \in \mathbf{W}_{explore}$ perform MAP inference with max iterations set to $b$;
    if $\mathrm{argmax}_{\mathbf{w} \in \mathbf{W}_{explore}} \gamma'(\mathbf{w}, b) > \gamma^*$ then
        $\gamma^* = \max_{\mathbf{w} \in \mathbf{W}_{explore}} \gamma'(\mathbf{w}, b)$;
        $\mathbf{w}^* = \mathrm{argmax}_{\mathbf{w} \in \mathbf{W}_{explore}} \gamma'(\mathbf{w}, b)$;
    end
    $t = \lfloor \frac{t}{\eta} \rfloor$;
    $b = B\eta^{iter}$;
    $\mathbf{W}_{explore} = top_t(\mathbf{W}_{explore})$, ;
    // $top_t$ ranks $\mathbf{W}_{explore}$ based on $\gamma'$ and returns top $t$ weights;
end
```

```
Algorithm 5: Hyperband for weight learning
Result: $\mathbf{w}^*$ : weight configuration with best evaluation metric
$\hat{R} =$ maximum number of resources to be allocated;
$\eta =$ the proportion of weight configurations to be removed in each round;
$s_{max} = \lfloor \log_\eta(\hat{R}) \rfloor$, maximum bracket size;
$\gamma^* = -\infty$, the best $\gamma$ value obtained so far;
for $s \in \{s_{max}, s_{max} - 1, \cdots, 1\}$ do
    $t = \lfloor \frac{(s_{max}+1)\eta^s}{(s+1)} \rfloor$;
    $B = \hat{R}\eta^{-s}$;
    $\mathbf{W}_{explore} \sim Dirichlet(\mathrm{A})^t$, $t$ distinct samples from a Dirichlet distribution;
    $\mathbf{w}_s, \gamma_s = SuccessiveHalving(B, W_{explore})$;
    if $\gamma_s > \gamma^*$ then
        $\gamma^* = \gamma_s$;
        $\mathbf{w}^* = \mathbf{w}_s$;
    end
end
```

3.6 Bayesian optimization for weight learning

Next, we introduce BOWL (Bayesian Optimization for Weight Learning), which uses GPR to perform weight learning in the BO framework.[2] Previously discussed approaches make no assumptions about the search space and the evaluation function (γ). They randomly sample weight configurations from the search space and evaluate the function γ. This

[2] Note that, while BO can refer to many different approaches, in this paper we loosely refer to a specific way of using GPR in BO framework to learn weights as BOWL .

process can be made more efficient by assuming that the metric obtained by two weight configurations $\mathbf{w}_1$ and $\mathbf{w}_2$ are likely to be similar ($\gamma(\mathbf{w}_1) \approx \gamma(\mathbf{w}_2)$) if the distance between them is small. This implies that when searching the space we can make use of previous observations and either explore more diverse weight configurations or exploit and choose weight configurations closer to previous configurations that performed well. This is exactly the goal of the new approach that we introduce here, BOWL . Next we explain BOWL and our choices for both the kernel function in GPR and the acquisition function in BO which are key in determining the method's performance.

A high-level sketch of BOWL is as follows: first, a weight configuration $\mathbf{w} \in \mathbf{W}$ is chosen using an acquisition function α (discussed in Sect. 3.6.2). Next, inference is performed using the current weight configuration $\mathbf{w}$, and $\gamma(\mathbf{w})$ is computed. Then the GPR is updated with $\mathbf{w}$ and $\gamma(\mathbf{w})$. Finally, after t iterations, the weight configuration that resulted in the highest value for γ is returned. See Algorithm 6 for details. As mentioned earlier, there are two primary components of BOWL that need to be defined: the kernel function used in GPR and the acquisition function α.

Algorithm 6: Bayesian optimization for weight learning

```
Result: $\mathbf{w}^*$ : weight configuration with best evaluation metric among samples
$\mathbf{W} = Dirichlet(A)$;
$t$ = maximum number of weight configurations to explore;
for $iter \in \{1, \ldots, t\}$ do
    $\mathbf{w}_{next} = \text{argmax}_{\mathbf{w} \in \mathbf{W}}\, \alpha(\mathbf{w})$; $\alpha$//an acquisition function chosen from Section 3.6.2;
    perform MAP inference to compute $\gamma(\mathbf{w}_{next})$;
    update GPR with $\gamma(\mathbf{w}_{next})$;
    if $\gamma(\mathbf{w}_{next}) > \gamma(\mathbf{w}^*)$ then
        $\mathbf{w}^* = \mathbf{w}_{next}$;
    end
end
```

3.6.1 Kernel function

In this work we employ the *squared exponential kernel*, k_{SE}, over a r' dimensional projection of the weight space, $\mathcal{E} : \mathbb{R}^r_+ \to \mathbb{R}^{r'}$:

$$k(\mathbf{w}_i, \mathbf{w}_j) = k_{SE}(\mathcal{E}(\mathbf{w}_i), \mathcal{E}(\mathbf{w}_j)) = \tilde{\sigma} \cdot exp\{\frac{\|\mathcal{E}(\mathbf{w}_i) - \mathcal{E}(\mathbf{w}_j)\|_2^2}{2\rho^2}\} \tag{16}$$

where $\tilde{\sigma}$ is the *amplitude* parameter and ρ is the *characteristic length-scale* parameter of k_{SE}. Leveraging the fact that input projections preserve the positive semi-definiteness of the kernel (15), we have that k is a valid covariance function. Furthermore, the parameters of k_{SE} have the same interpretation for k. The scaling factor ρ affects the smoothness of the approximation (the larger the value, the more smooth the approximation) and the number of iterations required to explore the space. We choose ρ such that a reasonable exploration of the space is possible in t iterations. The value of $\tilde{\sigma}$ is chosen based on the range of the metric being learned.

As mentioned in Sect. 2.4.2, the squared exponential kernel is a stationary kernel. In fact, the kernel can be written as a function of only the absolute value of the Euclidean distance of its inputs. Kernels with this property are called isotropic, or radial basis functions

(Rasmussen and Williams 2005). The isotropic stationarity property is not typically observed in practice for weight configurations. However, this assumption can be satisfied in the projected weight space defined by the function $\mathcal{E}$. Ideally, the weight space projection should be such that if the distance between two projected weight configurations is zero then the output of the function γ should be equal. Further, as the distance between the two projected weight configurations increases, the correlation between the output of the γ function should go to zero. In Sect. 4.2, we introduce a projection for the weights and show that if distances measured in the projected space are zero, then the output of the γ function is the same. This indicates better correlation among function values and distances measured in the projected space.

The squared exponential kernel makes a smoothness assumption on γ. Specifically, it assumes that γ is infinitely differentiable at every point in its domain. While this might seem restrictive, we argue that this is a reasonable assumption in the context of weight learning in SRL. To do this, we constrain ourselves to only those metrics ($\omega(\mathbf{y}, \mathbf{y}^*)$) that are smooth with respect to the random variables (such as MSE). Note, our assumption is that the function γ is smooth as a function of weight configurations $\mathbf{w}$ and not the random variables $\mathbf{y}$. Hence, it is non-trivial to prove smoothness of γ. However, with the above constraint on the possible metrics, we know that if a small change in $\mathbf{w}$ leads to a small change in $\mathbf{y}$, then the function γ is also smooth. This directly leads to the conditioning of the problem. If a problem is well-conditioned then our assumption about the smoothness of γ is precise. If the problem is ill-conditioned then this assumption fails to hold and the function learned in BOWL could be a poor approximation of γ. Further, in practice we observe that small changes in weights generally do not affect the γ function significantly which indicates smoothness. Even though we assume $\omega(\mathbf{y}, \mathbf{y}^*)$ is a smooth function to justify the squared exponential kernel, in our empirical evaluation we observe this is effective even on non-smooth evaluation functions such as accuracy.

3.6.2 Acquisition function

Another crucial component of our algorithm is the acquisition function α. The function α determines the next weight configuration on which to evaluate the function γ, i.e., $\mathbf{w}_{next} = \text{argmax}_{\mathbf{w} \in \mathbf{W}}\, \alpha(\mathbf{w})$. The evaluation of γ updates the distribution described by the underlying GP of BOWL . We would like to explore a variety of weights to allow BOWL to develop a complete picture of the weight learning objective γ while also exploiting previous observations to target promising regions of the weight space. To achieve this, we consider four well studied acquisition functions in the context of BO.

Upper confidence bound (UCB) (Srinivas et al. 2010): is an optimistic policy with provable cumulative regret bounds. The acquisition function can be written as:

$$\alpha(\mathbf{W}) = \mu(\mathbf{w}) + \psi \cdot \sigma(\mathbf{w})$$

where μ and σ are the mean and variance predicted by the GP and $\psi \geq 0$ is a hyperparameter set to achieve optimal regret bounds.

Thompson sampling (TS) (Thompson 1933): is an information-based policy that considers the posterior distribution over the weights $\mathbf{W}$. The acquisition function can be written as:

$$\alpha(\mathbf{W}) = \tilde{p}(\mathbf{w})$$
$$\tilde{p}(\mathbf{w}) \sim \mathcal{N}(\mu(\mathbf{w}), \sigma(\mathbf{w}))$$

where $\tilde{p}$ are samples obtained from the distribution computed at the point $\mathbf{w}$.

Probability of improvement (PI) (Kushner 1964): is an improvement-based policy that favors points that are likely to improve an incumbent target τ. The acquisition function can be written as:

$$\alpha(\mathbf{W}) = \mathbb{P}(\gamma(\mathbf{w}) > \tau) = \mathcal{F}\Big(\frac{\mu(\mathbf{w}) - \tau}{\sigma(\mathbf{w})}\Big)$$

where $\mathcal{F}$ is the standard normal cumulative distribution function and τ is set adaptively to the current best observed value for γ.

Expected improvement (EI) (Mockus et al. 1978): is an improvement-based policy similar to PI. Instead of probability, it measures the expected amount of improvement. The acquisition function can be written as:

$$\alpha(\mathbf{W}) = \Big\{(\mu(\mathbf{w}) - \tau)\mathcal{F}\Big(\frac{\mu(\mathbf{w}) - \tau}{\sigma(\mathbf{w})}\Big) + (\sigma(\mathbf{w}))\mathcal{F}\Big(\frac{\mu(\mathbf{w}) - \tau}{\sigma(\mathbf{w})}\Big)\Big\}$$

where $\mathcal{F}$ is the probability density function of a standard normal distribution function.

3.7 Efficiency of search-based approaches

In practice, it is inefficient to use search-based approaches for high-dimensional problems. For instance, GPR has been shown to work best when the number of dimensions is less than 50 (Wang et al. 2016). This makes weight learning an ideal use case for search-based approaches, because typically SRL models have just tens of rules and most often the number of rules does not exceed 50.

The success of all of the search-based weight learning approaches discussed so far rely on the weight configurations, $\mathbf{W}_{explore}$, being an accurate representative sample of the weight space. Furthermore, for BOWL , the distances measured between weight configurations should correlate to the MAP estimates of the corresponding parameterizations of the LMRF distribution. In the next section, we first show that the weight space in both PSL and MLN are redundant and weight configuration distances do not correlate to distances in MAP estimates. Next, assuming positive weights for rules, we introduce a novel projection that address these challenges. Finally, we also provide an efficient strategy to approximately sample from the projected space ensuring the effectiveness of the search-based approaches introduced.

4 Efficient space to search for weights

The weights of a SRL model consisting of r rules is represented via a vector in an r-dimensional space and is referred to as *original space* (OS). However, this is an inefficient space to perform weight learning using a search-based approach. This is because there exists many weight configurations which yield the same solution when performing MAP inference in SRL models. The main reason for this is because weights in SRL models are relative and scale invariant at the time of MAP inference. We can show that any SRL model with weights in $\mathbb{R}$ can be re-scaled with a positive constant $\tilde{c}$ without any change to the solution obtained through the objective in Eq. 3. This shows that weights in SRL models are scale invariant when performing MAP inference.

Theorem 1 *Consider any SRL model with r rules and weights* $\mathbf{w} = \{w_1, \dots, w_r\}$ *such that for all* $i = 1, \cdots, r$, $w_i \in \mathbb{R}^+$ ($w_i \in \mathbb{R}$ *for MLN*). *For all weight configurations* $\tilde{c} \cdot \mathbf{w}$ *where* $\tilde{c} > 0$, *the solution obtained for* $\mathbf{y}$ *by performing MAP inference using weights* $\mathbf{w}$ *and* $\tilde{c} \cdot \mathbf{w}$ *are the same*, *i.e.*, $\text{argmax}_{\mathbf{y}}\, \hat{s} \cdot \mathbf{E}(\mathbf{y}|\mathbf{x}, \mathbf{w}) = \text{argmax}_{\mathbf{y}}\, \hat{s} \cdot \mathbf{E}(\mathbf{y}|\mathbf{x}, \tilde{c} \cdot \mathbf{w})$.

Proof The MAP inference objective generated by using weights $\tilde{c} \cdot \mathbf{w}$ can be written as:

$$\begin{aligned}
\underset{\mathbf{y}}{\text{argmax}}\, \hat{s} \cdot \mathbf{E}(\mathbf{y}|\mathbf{x}, \tilde{c} \cdot \mathbf{w}) &= \underset{\mathbf{y}}{\text{argmax}}\, \hat{s} \cdot \sum_{i}^{l} \tilde{c} \cdot w_i \phi_i(\mathbf{x}, \mathbf{y}) \\
&= \underset{\mathbf{y}}{\text{argmax}}\, \tilde{c} \cdot \hat{s} \cdot \sum_{i}^{l} w_i \phi_i(\mathbf{x}, \mathbf{y}) \\
&= \underset{\mathbf{y}}{\text{argmax}}\, \tilde{c} \cdot \hat{s} \cdot \mathbf{E}(\mathbf{y}|\mathbf{x}, \mathbf{w}) \\
&= \underset{\mathbf{y}}{\text{argmax}}\, \hat{s} \cdot \mathbf{E}(\mathbf{y}|\mathbf{x}, \mathbf{w})
\end{aligned}$$

□

Since in our search-based approaches we optimize with respect to a user-defined evaluation metric, and this function depends only on the random variables obtained by MAP inference, we have that the user-defined evaluation metric function is also scale invariant.

4.1 Challenges in the original space

OS for weights has two fundamental challenges for search-based approaches: (1) OS is redundant and (2) the distance between weights in OS does not translate to true correlation of the MAP solution obtained by using these weights. The redundancy of space is clear from Theorem 1 as the weights on any line intersecting origin in OS will have the same solution. As a result, the Euclidean distance, $\delta_{i,j}$, between two weight configurations, $\mathbf{w}_i$ and $\mathbf{w}_j$, can be extremely large and still yield the exact same solution. The example below clearly illustrates this phenomenon:

Example 2 Consider a model with two rules $\mathbf{w} = \{w_1, w_2\}$. Let us assume three possible weight configurations for this problem: $\mathbf{w}_1 = \{0.1, 0.1\}$, $\mathbf{w}_2 = \{1.0, 1.0\}$, and $\mathbf{w}_3 = \{0.1, 0.0001\}$. Assuming that the number of groundings for both rules are the same, the weights of the rules in $\mathbf{w}_1$ and $\mathbf{w}_2$ indicate that both rules are equally important and lie on a line intersecting origin, while in $\mathbf{w}_3$ the first rule is 1000 times more important than the second rule and is not on the same line. This results in the function γ producing the same output for $\mathbf{w}_1$ and $\mathbf{w}_2$, and potentially a different value for $\mathbf{w}_3$. Based on this, the weight configuration $\mathbf{w}_3$ should be significantly different from the weight configurations $\mathbf{w}_1$ and $\mathbf{w}_2$, while $\mathbf{w}_1$ and $\mathbf{w}_2$ should be similar. Unfortunately, the Euclidean distances measured between the weight configurations, $\delta_{1,2} = 1.27$, $\delta_{1,3} = 0.09$, and $\delta_{2,3} = 1.34$, do not behave in this manner. The distance $\delta_{1,2}$ is much larger than distance $\delta_{1,3}$. Therefore, some of the search-based approaches such as BOWL would incorrectly infer that the function value of $\gamma(\mathbf{w}_1)$ is more correlated with $\gamma(\mathbf{w}_3)$ than $\gamma(\mathbf{w}_2)$. However, as argued above, we want the opposite behavior.

It is important to note that the redundancy illustrated in the example is an inherent property of the OS, and will also arise when using metrics alternative to the Euclidean distance. In fact, any metric that is not scale invariant will be redundant with respect to the MAP inference problem. Certainly there are distance measures such as the cosine similarity metric which are scale invariant. Measuring distances between weight configurations using their cosine would resolve the scale dependence redundancy of the OS, however, this metric may still struggle with the second challenge of distances not translating to true correlations between MAP solutions. Indeed, one part of this challenge is overcoming the redundancies of the OS, but another is the being robust to the number of groundings of the rules. In this section we introduce a new projection for weight configurations such that the standard Euclidean distance is both not redundant with respect to the scale dependence of MAP inference and is not effected by the number of groundings of the rules, resulting in a metric that is more representative of the true correlation of weights. For BOWL , this projection supports the application of isotropic kernels.

For the remainder of this section we assume weights of the rules to be positive. While this does not restrict PSL which supports only positive weights, it does constrain MLNs which support negative weights. However, it has been shown that a negative weighted rule in MLNs can be replaced with a negated rule and positive weight with the same magnitude. Further in Sect. 5, we show negative weight learning can be supported by search-based approaches.

4.2 Scaled space

In order to perform efficient and effective search, we define a new space for the weight configurations called *scaled space* (SS). SS is a projection of weights onto a relative space. We use the ratio of weights between the rules to define the relative importance of weights in the configuration. This projection eliminates redundancies due to scale invariance and results in distances that more closely correspond to the actual correlation between the weight configurations. Formally, we define SS as:

Definition 3 Given a set of weights $\mathbf{w} = \{w_1, \dots, w_r\} \in (0, \infty]^r$, SS $\mathcal{E}$ is a projection defined on $\mathbf{w}$ such that $\mathcal{E}(\mathbf{w}) \in \mathbb{R}^{(r-1)}$ is given by:

$$\mathcal{E}(\mathbf{w}) = \{\forall_{i=2}^{r}(ln(w_i) - ln(w_1))\} \tag{17}$$

Given the definition of SS, we next define the distance between two weight configurations in SS.

Definition 4 The distance between two weight configurations $\mathbf{w}_i$ and $\mathbf{w}_j$, $\Delta_{i,j}$, in SS is defined as:

$$\Delta_{i,j} = ||\mathcal{E}(\mathbf{w}_i) - \mathcal{E}(\mathbf{w}_j)||_2^2 \tag{18}$$

Given the definition of SS, we can now show how SS addresses the two challenges mentioned for OS. We first show that any weight configuration on a line intersecting the origin in OS will be represented by the same point in SS. This eliminates the redundancy that exists in OS. Next we show that in SS a distance of zero ($\Delta_{i,j} = 0$) between two weight configurations $\mathbf{w}_i$ and $\mathbf{w}_j$, implies that the two weight configurations yield the same MAP inference solution. Hence SS yields a more accurate representation of distances between weight configurations.

Theorem 2 *Given two weight configurations* $\mathbf{w}_1$ *and* $\mathbf{w}_2$, *if* $\mathbf{w}_1 = c \cdot \mathbf{w}_2$, *i.e.*, $\mathbf{w}_1$ *and* $\mathbf{w}_2$ *lie on a line intersecting the origin in OS then the resultant value in SS will be the same*, *i.e.*, $\mathcal{E}(\mathbf{w}_1) = \mathcal{E}(\mathbf{w}_2)$

Proof Given, $\mathbf{w}_1 = c \cdot \mathbf{w}_2$ implies:

$$w_{1,i} = c \cdot w_{2,i}; i \in 1, \dots, r$$

$\mathcal{E}(\mathbf{w}_1)$ by definition is given by $\{\forall_{i=2}^{r}(ln(w_{1,i}) - ln(w_{1,1}))\}$. By replacing $w_{1,i}$ with $cw_{2,i}$ we get:

$$\mathcal{E}(\mathbf{w}_1) = \{\forall_{i=2}^{r}(ln(w_{2,i}) - ln(w_{2,1}))\} = \mathcal{E}(\mathbf{w}_2)$$

Therefore, if $\mathbf{w}_1 = c \cdot \mathbf{w}_2$ then $\mathcal{E}(\mathbf{w}_1) = \mathcal{E}(\mathbf{w}_2)$. □

Theorem 3 *Given two weight configurations* $\mathbf{w}_1$ *and* $\mathbf{w}_2$, *if* $\mathcal{E}(\mathbf{w}_1) = \mathcal{E}(\mathbf{w}_2)$ (*i.e.*, $\Delta_{1,2} = 0$) *then the solution obtained for* $\mathbf{y}$ *by optimizing Eq.* 3 *with both the weight configurations are the same*.

Proof Let $\mathbf{w}_1 = \{w_{1,1}, \dots, w_{1,r}\}$, $\mathbf{w}_2 = \{w_{2,1}, \dots, w_{2,r}\}$ and $\mathcal{E}(\mathbf{w}_1) = \mathcal{E}(\mathbf{w}_2)$. As the two weight configurations are the same in SS, the equality can be written as:

$$ln(\mathbf{w}_1) - ln(w_{1,1}) = ln(\mathbf{w}_2) - ln(w_{2,1})$$
$$\mathbf{w}_1 = \frac{w_{1,1}}{w_{2,1}}\mathbf{w}_2$$

Since $w_{1,1} \in (0, 1]$ and $w_{2,1} \in (0, 1]$ are constants, the resulting optimization problems are equivalent:

$$\begin{aligned} \underset{y}{\text{argmax}}\, \hat{s} \cdot \mathbf{E}(\mathbf{y}|\mathbf{x}, \mathbf{w}_1) &= \underset{y}{\text{argmax}}\, \frac{w_{1,1}}{w_{2,1}} \hat{s} \cdot \mathbf{E}(\mathbf{y}|\mathbf{x}, \mathbf{w}_2) \\ &= \underset{y}{\text{argmax}}\, \hat{s} \cdot \mathbf{E}(\mathbf{y}|\mathbf{x}, \mathbf{w}_2) \end{aligned}$$

Therefore, if the distance between two weight configurations is 0 in SS, then the solutions of their corresponding SRL model by optimizing Eq. 3 are the same. □

Theorem 2 shows that SS overcomes redundancies due to scale invariance of MAP inference and Theorem 3 proves the distances in SS are a more accurate representation of correlation than OS. These theorems show that SS is a more efficient space for weight learning using search-based approaches. Note that in the definition of SS we use the weight of the first rule to compute the projection. This choice is arbitrary and can be switched to any rule without affecting the space.

Example 3 (Continued) Consider our earlier example. The weights and the distances of our running example in SS $\mathcal{E}$ using Eqs. 17 and 18 are: $\mathcal{E}(\mathbf{w}_1) = \{0\}$, $\mathcal{E}(\mathbf{w}_2) = \{0\}$, $\mathcal{E}(\mathbf{w}_3) = \{6.907\}$, $\Delta_{1,2} = 0$, $\Delta_{1,3} = 47.7$, and $\Delta_{2,3} = 47.7$.

A drawback of SS is that it does not support a weight of zero for any rule in the model. This means that all rules in the configuration must participate in the model. However, in practice, we mitigate this by using SS only to measure distances between weight configurations and add a small positive value ϱ to all weights. In order to generate weight configurations for the search, we sample from a hypersphere in OS which has similar properties as SS. We discuss this in detail in the Sect. 4.4.

4.3 The effect of varied number of groundings in the scaled space

Our discussion on SS so far has made a very important simplifying assumption, that the number of groundings for each rule in the model is the same. However, the number of groundings produced by different rules are seldom the same and the number of groundings produced by a rule has an impact on the inference of the random variables. The weight associated with each rule is repeated for each ground instance of that rule. This leads to the weight of each rule having varied influence on the minimization of the energy function. For instance, if a model has two equally weighted rules, but one rule produces 10 times more groundings than the other, then that rule implicitly becomes 10 times more important in the model.

Next we show that while the number of groundings has an impact on the solution obtained by SRL models, it does not impact the correctness of SS. We can modify the weights to accommodate the number of groundings of the rules in the model. Consider a model with r rules and let $\boldsymbol{\beta} = \{\beta_1, \dots, \beta_r\}$ be the number of groundings for each of the r rules. We define a grounding factor κ for each rule. For rule z, the grounding factor $\kappa_z = \frac{\beta_z}{\max(\boldsymbol{\beta})}$, where $\boldsymbol{\kappa} = \{\kappa_1, \dots, \kappa_r\}$ is the vector of grounding factors. Therefore, the true weight associated with the z^{th} rule is $\kappa_z \cdot w_z$ and the grounding adjusted weight configuration can be represented as an element-wise product between $\boldsymbol{\kappa}$ and $\mathbf{w}$, i.e., $\tilde{\mathbf{w}} = \boldsymbol{\kappa} \odot \mathbf{w}$. The distance between two weight configurations i and j in OS can be re-written as $||\tilde{\mathbf{w}}_i - \tilde{\mathbf{w}}_j||_2^2$. Similarly the distance in SS can be re-written as $||\mathcal{E}(\tilde{\mathbf{w}}_i) - \mathcal{E}(\tilde{\mathbf{w}}_j)||_2^2$. However, the scaling factor $\boldsymbol{\kappa}$ does not affect the distance in SS as $\boldsymbol{\kappa}$ is constant for both weight configurations and cancels when computing the distance leaving the distance in SS unchanged.

Theorem 4 *Given two weight configurations* $\mathbf{w}_i$ *and* $\mathbf{w}_j$*, a set of grounding factors of* $\boldsymbol{\kappa}$*, and grounding adjusted weight configurations* $\tilde{\mathbf{w}}_i = \boldsymbol{\kappa} \odot \mathbf{w}_i$ *and* $\tilde{\mathbf{w}}_j = \boldsymbol{\kappa} \odot \mathbf{w}_j$*, the distance measured between both* $(\mathbf{w}_i, \mathbf{w}_j)$ *and* $(\tilde{\mathbf{w}}_i, \tilde{\mathbf{w}}_j)$ *in SS are equal, i.e.* $||\mathcal{E}(\mathbf{w}_i) - \mathcal{E}(\mathbf{w}_j)||_2^2 = ||\mathcal{E}(\tilde{\mathbf{w}}_i) - \mathcal{E}(\tilde{\mathbf{w}}_j)||_2^2$.

Proof To prove the above theorem we consider the difference between the weight configurations $\mathcal{E}(\tilde{\mathbf{w}}_i) - \mathcal{E}(\tilde{\mathbf{w}}_j)$:

$$\begin{aligned}
\mathcal{E}(\tilde{\mathbf{w}}_i) - \mathcal{E}(\tilde{\mathbf{w}}_j) &= (ln(\boldsymbol{\kappa} \odot \mathbf{w}_i) - ln(\kappa_1 \cdot w_{i,1})) - (ln(\boldsymbol{\kappa} \odot \mathbf{w}_j) - ln(\kappa_1 \cdot w_{j,1})) \\
&= (ln(\mathbf{w}_i) - ln(w_{i,1})) - (ln(\mathbf{w}_j) - ln(w_{j,1})) \\
&= \mathcal{E}(\mathbf{w}_i) - \mathcal{E}(\mathbf{w}_j)
\end{aligned}$$

Since $\mathcal{E}(\tilde{\mathbf{w}}_i) - \mathcal{E}(\tilde{\mathbf{w}}_j) = \mathcal{E}(\mathbf{w}_i) - \mathcal{E}(\mathbf{w}_j)$, the distances are also equal. □

Theorem 4 shows that the distance measured between two weight configurations in SS is robust while considering the size of their groundings.

4.4 Sampling weight configurations for search

Most search-based approaches work by sampling weight configurations in order to explore and search for a set of weights with the best evaluation score. In order to do this effectively, we must ensure that the samples generated for exploration are representative of the space. While one could sample points from the SS directly, this introduces three challenges. First, as mentioned earlier, points in SS cannot represent rules with zero weight. Second, to perform MAP inference weights need to be in OS and there is a one-to-many mapping between SS and OS. Third, sampling uniformly from the SS would require defining minimum and maximum ratio hyperparameters which would bound the search space and be potentially non-intuitive to set. It is much simpler to sample configurations in the OS and measure distances in the SS. Further, we will see that this approach leads to a set of easy to use hyperparameters that give the modeler a principled method for concentrating the weight search to regions of interest.

One straightforward approach to sampling weight configurations is to uniformly points from the positive orthant of a unit hypercube in OS and then project the points onto SS to measure distances. The approach can be summarized as:

$$\mathbf{w} \sim \mathit{Unif}([0,1])^r$$

where *Unif* generates uniform random numbers between $[0,1]^r$. Since, the influence of weights in SRL models are scale invariant for MAP inference, we can ensure all possible weight configurations that can be represented in the $\mathbb{R}^{+r}$ can be represented in $[0,1]^r$. However, the main problem with this approach is that the resulting configurations will have a low spread in SS as OS has many redundancies. Specifically, the projection will result in more probability mass around configurations with ratios near 1. This is not desirable as we would prefer to have a uniform/large spread of possible weight configurations in SS.

As mentioned earlier, MAP inference in SRL models are scale invariant and hence weights along a line passing through the origin, 0, in OS are equivalent, i.e., the direction of a vector starting at the origin in OS is sufficient to represent all weight configurations in SRL models. This implies that the weight configurations obtained by sampling from the surface of an r-dimensional unit hypersphere in the positive quadrant represents all possible weight configurations in OS. Therefore, uniformly sampling from the surface of this hypersphere (we only refer to the positive orthant of the unit hypersphere) is a close approximation of SS. It is trivial to show that Theorem 2 and 3 apply to this space (full proofs are in "Appendix"). However, Theorem 4 does not hold true for this space. This is because the grounding factor generally does not preserve the orientation of the vector, and as a result will modify the distances in this space (full proof shown in "Appendix"). Therefore, we treat the hypersphere as an approximation of SS and sample weight configurations from the hypersphere but compute distances in SS. Uniform samples of weight configurations from the surface of the hypersphere can be obtained by first sampling points from a standard multivariate-normal distribution and projecting the values to the hypersphere (Muller 1959; Marsaglia 1972) (since we want samples only from the positive orthant we project all samples on to this orthant by taking the absolute value):

$$\mathbf{s} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$
$$\mathbf{w} = |\frac{\mathbf{s}}{||\mathbf{s}||}|$$

0 is a r-dimensional zero vector and **I** is an r-dimensional identity matrix.

While uniform sampling from the surface of a hypersphere ensures that every orientation of the weight vector is equiprobable, in practice this might not always be desirable. The primary reason for this is that for any weight configuration sampled from the hypersphere, if we choose two weights $w_i < w_j$ the $P(\frac{w_i}{w_j} < 0.1) \approx 0.11$ (assuming $r = 2$). This implies that the ratio between weights will be typically be close to 1 which is not always ideal as we would expect the evaluation metric to show larger variance with larger ratios. In order to circumvent this, we propose another sampling strategy which gives full control over the distribution of the weight configurations. We sample from an r-dimensional Dirichlet distribution which generates samples from the probability simplex and then project the samples onto the surface of the positive orthant of the r-dimensional unit hypersphere. There is a one-to-one mapping between the points on the probability simplex and the positive surface of the hypersphere, i.e., all possible weight configurations can be generated by this procedure. The *concentration* hyperparameter, A of the Dirichlet distribution can be modified to generate samples concentrated towards the center (near equally valued weights) or the poles (extreme ratios) or anywhere in between. The sample generation process is as follows:

$$\mathbf{w} \sim Dirichlet(\mathrm{A}) \tag{19}$$

where $\mathrm{A} \in R^{+r}$ is the hyperparameter that defines the Dirichlet distribution. The visualization of different densities obtained using different A in three dimensions is shown in "Appendix" along with its impact on the empirical evaluation in Sect. 6.3.2.

5 Accommodating negative weights in Markov logic networks

In this section we discuss how the sampling strategy discussed in Sect. 4.4 is modified for MLNs to accommodate negative weights when using CRS , HBWL , and BOWL . Since the Dirichlet distribution samples from the probability simplex, the weights sampled are strictly non-negative. To introduce the possibility of negative weights, we first sample weight configurations from the Dirichlet distribution and then randomly select an orthant in the r-dimensional Euclidian space. This random selection of an orthant has the same effect as independently flipping the sign of each weight in the sampled configuration so every orthant is equiprobable. We further apply the positive scale invariance property for weight configurations to ensure the uniqueness of samples. While no sampling is needed for RGS in MLNs, we ensure uniqueness of explored configurations by projecting the values onto a unit hypersphere instead of SS. In order to use BOWL in MLNs, instead of computing distance in SS, we compute the Euclidean distances of the weight configurations by projecting the weights onto the hypersphere.[3]

[3] Note this may lead to over generalization of the function as two points might be very close to each other but have significantly different outcome on the γ function (as mentioned in Example 2).

6 Empirical evaluation

In this section, we evaluate the search-based approaches for weight learning on various real-world datasets. We investigate four research questions through our experiments:

Q1 How do search-based approaches perform on real-world datasets compared to the existing methods?
Q2 Which search-based approach performs better weight learning in SRL?
Q3 Are search-based approaches scalable?
Q4 Are search-based approaches robust?

In order to answer these questions we selected five real-world datasets from different domains for which SRL models have promising results (Bach et al. 2017; Kouki et al. 2015).[4] Details of these datasets are as follows:

Jester: contains 2,000 users and 100 jokes (Goldberg et al. 2001). The task is to predict user's preference to jokes.

LastFM: contains 1,892 users and 17,632 artists. The task is to recommend artists to users by predicting the ratings for user-artist pairs.

Citeseer: contains 2,708 scientific documents, seven categories, and 5,429 directed citations. The task is to assign a category to each document.

Cora: is similar to Citeseer dataset, but contains 3,312 documents, six categories and 4,591 directed citations.

Epinions: contains 2,000 users and 8,675 directed links which are positive and negative trust links between users. The task is to predict the trust relation.

We evaluate the three search-based methods (RGS, CRS , and BOWL) for both MLNs and PSL, and HBWL for PSL only. While the implementation of RGS, CRS , and BOWL are efficient to run for both MLNs and PSL (more details in Sect. 6.2), HBWL with MLNs on our datasets is not as efficient and each experiment was estimated to take over a month. While the search-based approaches have been fully integrated into PSL codebase, for MLNs, we implement search-based methods as wrappers, i.e., we use an external script to generate weights and use Tuffy to perform MAP inference. Because of this, in HBWL , the SuccessiveHalving step requires regrounding of the same model many times, substantially increasing the time required to run. We use MLE , MPLE , and LME as baseline non-search-based methods for PSL and use LME (implemented in PSL) and DN (implemented in Tuffy), as the non-search-based baseline for MLN. The LME implementation for both MLN and PSL is optimized using Frank-Wolfe optimization. The prediction and separation subproblems of LME is performed with WalkSAT for MLNs and with ADMM for PSL. As MLNs are discrete, we perform evaluations only on the three discrete datasets, Citeseer, Cora, and Epinions. For each dataset we use two domain appropriate evaluation metrics. We report *MSE* and *AUROC* for the Jester and LastFM datasets, *categorical accuracy (CA)* and *F1* for the Cora and the Citeseer datasets, and *AUROC* and *F1* for the Epinions dataset.

[4] Models, code, and data: https://github.com/linqs/srinivasan-mlj20.

Table 1 Mean and standard deviation of mean square error (MSE), AUROC, and F1 performance of PSL weight learning methods on Jester, LastFM, and Epinions

	Jester		LastFM		Epinions	
	MSE	AUROC	MSE	AUROC	F1	AUROC
MLE	0.053 (2.3e−4)	0.731 (1.4e−3)	0.061 (1.5e−3)	0.548 (3.2e−3)	0.712 (1.2e−2)	0.811 (1.5e−2)
MPLE	0.068 (5.6e−4)	0.700 (1.5e−3)	**0.060** (6.6e−4)	0.552 (2.4e−3)	0.711 (1.2e−2)	0.792 (2.3e−2)
LME	0.063 (4.4e−4)	0.702 (1.6e−3)	0.061 (5.7e−4)	0.549 (2.0e−3)	0.712 (1.2e−2)	0.807 (3.0e−2)
RGS	0.053 (5.6e−4)	**0.762** (5.8e−3)	0.061 (2.0e−3)	0.574 (1.2e−3)	0.711 (1.2e−2)	**0.814** (2.0e−2)
CRS	0.053 (1.3e−3)	0.744 (1.2e−2)	0.062 (3.3e−3)	0.572 (6.5e−3)	0.712 (1.2e−2)	0.797 (2.0e−2)
HBWL	**0.052** (5.9e−4)	0.756 (6.6e−3)	0.064 (2.8e−3)	0.556 (6.6e−3)	0.711 (1.4e−2)	0.810 (1.6e−2)
BOWL	0.053 (6.4e−4)	0.761 (5.0e−3)	0.063 (1.5e−3)	**0.587** (3.7e−3)	**0.713** (1.1e−2)	0.780 (2.1e−2)

The best scoring methods are shown in bold

6.1 Performance analysis

To address [Q1] and [Q2], we compare the performance of all search-based approaches RGS, CRS , HBWL , and BOWL with MLE , MPLE , and LME in PSL and DN and LME in MLN on several metrics. For the datasets, we use the 8 folds generated by Bach et al. (2017) for Citeseer, Cora, Epinions and Jester and 5 folds generated by Kouki et al. (2015) for LastFM and perform cross validation. In RGS , we use the PSL distribution's default set of possible weight values $V = \{0.001, 0.01, 0.1, 1, 10\}$, while for MLNs we use $V \cup -V$. For CRS , HBWL , and BOWL , we specify each dimension of A to be 0.05 for PSL (more experiments with different values can be found in Sect. 6.3.2) and 0.1 for MLNs. Note, while we require non-zero weights for the use of SS in BOWL , we do not explicitly choose a lower bound for the weights, i.e., the SS parameter ϱ is set to 0, as the probability of sampling a weight of 0 from a Dirichlet distribution is 0. In HBWL for PSL, $\hat{R}$ represents the number of iterations of ADMM inference and is set to the default value of 25000 and similarly η is set to 4. For RGS, CRS , HBWL , and BOWL , the maximum number of weight configurations to explore in order to approximate the user-defined evaluation metric function is set to $t = 50$. Although in our experiments we show that the best metric value is usually obtained at $t < 25$ (especially for BOWL , this is likely because the function we intend to learn has several flat regions). We also use a stopping criterion for BOWL which terminates the exploration if the standard deviation at all sampled weight configurations is less than 0.5. MLE , MPLE , LME , and DN are allowed to run for 100 iterations or until convergence whichever is smaller. For LME in both PSL and MLN, the parameter C is set to 1. For BOWL , we use UCB as our acquisition function with $\psi = 1$ to favor modest exploration. However in Sect. 6.3.3 we show that similar performance can be obtained with PSL by using the other acquisition functions discussed in Sect. 3.6.2. Other hyperparameters that we use for BOWL are: $\tilde{\sigma} = 0.5, \rho = 1$, and the mean function is a constant 0.5. We set the value of ρ to 1.0 after exploring different values in $[10^5, 10^{-5}]$ using a validation set, and we set $\tilde{\sigma}$ to 0.5 as our metrics are mostly in the range [0, 1].

Tables 1, 2, 3, and 4 show the comparison between search-based approaches and other methods across the different datasets for both PSL and MLN respectively. We first analyze the performance results of PSL in Tables 1 and 2. Here we show search-based approaches are consistently among the best performing methods across all the datasets and metrics. Notably, BOWL is the best or within one standard deviation from the best performing

Table 2 Mean and standard deviation of categorical accuracy (CA) and F1 performance of PSL weight learning methods on Citeeseer and Cora

	Citeseer		Cora	
	CA	F1	CA	F1
MLE	0.835 (1.5e−4)	0.283 (6.4e−2)	0.881 (1.7e−2)	0.404 (8.5e−2)
MPLE	0.837 (7.6e−4)	0.293 (3.3e−2)	0.888 (3.1e−2)	0.439 (2.1e−2)
LME	0.838 (8.6e−3)	0.303 (3.5e−2)	**0.889** (4.0e−3)	0.442 (2.1e−2)
RGS	0.844 (2.8e−3)	0.321 (1.1e−2)	0.888 (4.1e−3)	0.438 (1.9e−2)
CRS	0.844 (2.6e−3)	**0.322** (1.3e−2)	0.887 (3.9e−3)	0.437 (1.8e−2)
HBWL	**0.845** (3.0e−3)	0.321 (1.3e−2)	0.888 (3.9e−3)	0.438 (1.8e−2)
BOWL	0.844 (2.1e−3)	0.319 (1.5e−2)	**0.889** (3.8e−3)	**0.443** (1.8e−2)

The best scoring methods are shown in bold

Table 3 Mean and standard deviation of categorical accuracy (CA) and F1 performance of MLN weight learning methods on Citeeseer and Cora

	Citeseer		Cora	
	CA	F1	CA	F1
DN	0.829 (2.0e−3)	0.267 (1.5e−2)	0.867 (3.1e−3)	0.339 (2.1e−2)
LME	0.820 (5.6e−3)	0.214 (1.5e−2)	0.864 (5.2e−3)	0.311 (2.9e−2)
RGS	0.827 (2.8e−3)	0.251 (1.1e−2)	0.865 (2.8e−3)	0.333 (2.8e−2)
CRS	0.829 (1.6e−3)	0.253 (1.7e−2)	0.866 (6.0e−3)	0.343 (1.5e−2)
BOWL	**0.833** (2.6e−3)	**0.281** (1.1e−2)	**0.871** (3.5e−3)	**0.354** (1.7e−2)

The best scoring methods are shown in bold

Table 4 Mean and standard deviation of F1 and AUROC performance of MLN weight learning methods on Epinions

	Epinions	
	F1	AUROC
DN	0.704 (1.2e−2)	0.608 (3.7e−2)
LME	0.704 (1.3e−2)	0.577 (1.7e−2)
RGS	0.708 (1.2e−2)	0.689 (3.5e−2)
CRS	**0.711** (1.0e−2)	**0.699** (1.7e−2)
BOWL	0.710 (1.1e−2)	0.659 (2.8e−2)

The best scoring methods are shown in bold

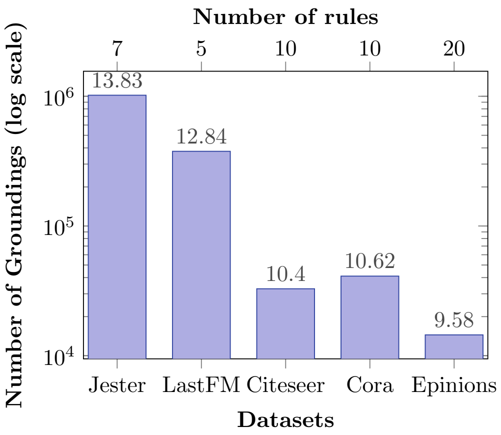
method on all datasets and metrics other than LastFM MSE. For the Epinions and Cora datasets, there is no large difference among all approaches on both metrics. However, on the Citeseer dataset, all search-based approaches perform better than other approaches on CA. An interesting observation is that on the Citeseer dataset, the weights found by MLE and LME perform about as well (within one standard deviation) as search-based approaches on F1 but not on CA. This can potentially be because the optimal weights for F1 are more aligned with maximum likelihood weights in this special case. This further strengthens our motivation to directly optimize for the evaluation metric rather than the likelihood. In LastFM, BOWL is significantly better than all other approaches on AUROC, however, RGS and the non-search based approaches outperform BOWL in the MSE metric on LastFM. One reason for likelihood-based approaches to perform better in LastFM MSE could be because the rating values mentioned in LastFM was constructed by fitting a negative binomial distribution (Kouki et al. 2015). This could potentially imply that maximizing the likelihood in PSL could directly lead to minimizing the MSE in this special and partially synthetic case. Overall, as mentioned earlier, search-based approaches perform better than other approaches and in general even the worst performing search-based method is better than likelihood-based approaches (like in Jester and LastFM AUROC). These experiments clearly show that search-based approaches are better suited for weight learning in PSL. Further, of the search-based weight learning approaches BOWL most consistently performs the best. This is because BOWL performs smart exploration of weight space to approximate the evaluation function.

Next, in Tables 3 and 4 we analyze the performance of the MLN weight learning methods on all datasets and metrics. We again see that search-based methods achieved the best performance for every dataset and metric. Except for F1 score in the Cora dataset, where DN performs as well as RGS, the non-search based methods, DN and LME perform worse than search-based approaches on all datasets and metrics. This observation further supports the use of search-based weight learning methods across SRL frameworks. Similar to the PSL experiments, BOWL most consistently outperforms its alternatives. BOWL is either the best method and or within one standard deviation from the best method on all but the Epinions dataset when evaluating the AUROC metric. The reason for poor performance of BOWL in Epinions with AUROC could be because Epinions has the largest number of rules (20 rules) compared to all other datasets. Further, since distances for BOWL in MLN are computed in OS, the covariance function used to approximate the AUROC is not ideal for the weight space, potentially leading to a poor exploration strategy. On the same dataset for F1 BOWL performs nearly as well as CRS which is the best. This indicates that the approximation of the evaluation function through BOWL is dependent on the function being approximated as well.

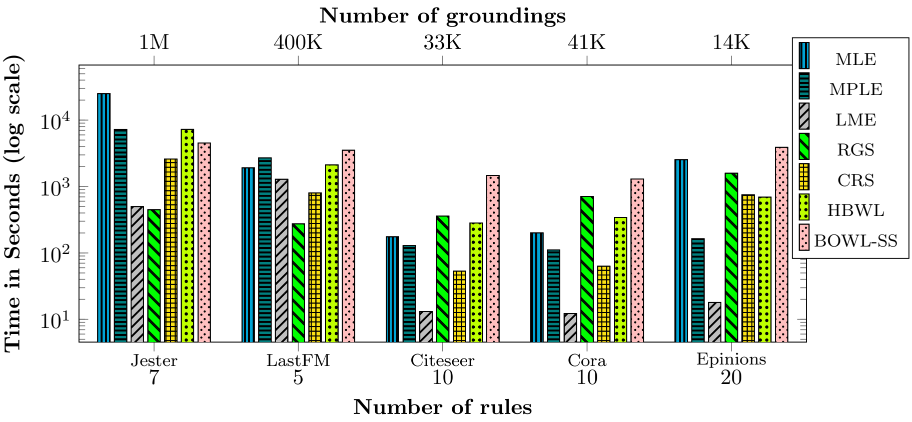
6.2 Scalability

In this section, we compare the runtimes of MLE , MPLE , LME , RGS, CRS , HBWL , and BOWL in PSL to measure the scalability of search-based approaches and address [Q3]. We do not compare runtimes of experiments with Tuffy. This is because the search-based approaches are implemented as wrappers around the MLN framework. Hence, the runtime numbers of different approaches will not be a fair comparison.[5] The number of parameters to learn in PSL and MLN is equal to the number of rules in the model and the data size translates to the number of groundings generated by the model. Figure 3a shows the number of groundings generated by each of the datasets along with the number of rules in each model. The Jester dataset produces the largest number of groundings ($\sim 1M$) using seven rules and the Epinions dataset produces the least number of groundings ($\sim 14K$) using the largest model (20 rules).

[5] Even though we reground the model every iteration for search-based methods in Tuffy, the search-based approaches were at least two times faster than DN in Citeseer and Cora datasets and due to early-stopping, BOWL in MLN was at least 5 times than DN .



(a) Groundings generated by different datasets.



(b) Time to learn vs. # of rules and groundings in datasets.

Fig. 3 Analyzing the scalability of different approaches on the number of rules and groundings. With number of iterations fixed, search-based approaches scale better with both the number of rules and groundings

Figure 3b shows the average runtimes measured accross all folds for all approaches on all datasets. Runtimes for some approaches depend on number of groundings while some others depend on number of rules. The runtime for MPLE primarily depends on number of groundings and as the number of groundings increases the runtime also increases by a factor of ~45 from Epinions to Jester dataset. The time taken to run LME depends not only on the number of groundings, but also the complexity of finding the margin. Therefore, the runtime of LME on the LastFM dataset is higher than the Jester dataset. MLE , RGS, CRS , HBWL , and BOWL depend on number of groundings through the time taken to perform inference on larger models. Since inference in PSL is efficient due to its convex objective, these approaches can scale better with the number of groundings compared to the other approaches. Further, inference time in PSL depends on both the number of groundings (which affects per iteration cost in solving) and ease of solving the optimization (which affects the number of iterations required to converge). This can be observed

Table 5 The table shows the mean and standard deviation of the metrics obtained by running search-based approaches with varied initialization

Datasets	Varied initializations			
	RGS	CRS	HBWL	BOWL
Jester (MSE)	0.053 (0.001)	0.053 (0.001)	0.052 (0.001)	0.053 (0.001)
LastFM (MSE)	0.067 (0.008)	0.067 (0.006)	0.065 (0.002)	0.067 (0.006)
Citeseer (F1)	0.319 (0.007)	0.321 (0.007)	0.322 (0.008)	0.320 (0.007)
Cora (F1)	0.412 (0.006)	0.409 (0.005)	0.411 (0.005)	0.411 (0.005)
Epinions (F1)	0.714 (0.002)	0.713 (0.002)	0.714 (0.002)	0.716 (0.001)

BOWL is the least affected by initialization

when we compare runtimes of the above mentioned five methods on the Epinions dataset and Cora dataset. The time taken to run Epinions is two times greater for MLE even though the number of groundings of Epinions is three times smaller. Overall, MLE has the largest increase in runtime (~150 fold increase) from Cora to Jester dataset. The runtimes of search-based approaches also depend on the number of evaluations (or resources allocated) they are allowed. For a good approximation of the evaluation function the number of evaluation points required increases exponentially with number of rules in the model. Since we fix the maximum number of iterations to 50 for all models, it does not affect our approaches. Specifically, the runtimes of BOWL across datasets does not vary as much as other approaches. BOWL has the smallest increase in runtime (~3 fold) between the Cora and Jester datasets across all methods. This is because BOWL has a constant overhead for performing updates in the GP and retrieving the best next set of weight configurations. This also ensures bad weights that converge poorly are chosen less often (as they are likely to yield poor evaluation metric value), hence making it efficient and scalable with number of groundings. Overall, search-based approaches can scale with a large number of groundings and produce the best results on the evaluation metric function.

6.3 Robustness

To address [Q4], we ran three sets of experiments: the first experiment is to check how robust search-based methods are w.r.t. different initializations, the second is to evaluate the robustness of CRS , HBWL , and BOWL to the hyperparameter A from the Dirichlet distribution, and the third experiment is to test the effects of choosing an acquisition function on the performance of BOWL .

6.3.1 Varied initialization

For the first experiment, we perform weight learning with all four search-based approaches using 30 random initializations and report the mean and standard deviation (std) of a metric per dataset in Table 5. Note that for this experiment we use UCB as our acquisition function in BOWL . Further, we use only one fold (of the eight folds) per dataset as we intend to measure the variance introduced by different initialization. In

Table 6 The mean and standard deviation of the performance of different search-based approaches with varying the A parameter in the Dirichlet distribution

Datasets	Methods	A = 10	A = 1	A = 0.1	A = 0.01
Citeseer - CA	CRS	0.844 (2.8*e*−3)	0.844 (2.3*e*−3)	0.843 (2.4*e*−3)	0.844 (2.6*e*−3)
	HBWL	0.843 (2.3*e*−3)	0.844 (2.3*e*−3)	0.843 (2.4*e*−3)	0.843 (2.3*e*−3)
	BOWL	0.844 (2.3*e*−3)	0.844 (3.0*e*−3)	0.845 (2.1*e*−3)	0.843 (2.1*e*−3)
Jester - MSE	CRS	0.064 (1.8*e*−3)	0.054 (5.6*e*−4)	0.053 (9.4*e*−4)	0.056 (2.2*e*−3)
	HBWL	0.061 (1.4*e*−3)	0.053 (6.8*e*−3)	0.052 (3.0*e*−4)	0.054 (5.4*e*−4)
	BOWL	0.053 (2.3*e*−4)	0.053 (4.0*e*−4)	0.053 (7.3*e*−4)	0.053 (2.3*e*−4)

Table 5, the standard deviation is very small for all methods on the Jester dataset. Then on Citeser, Cora, and Epinions, the standard deviation across different folds, shown in Tables 1 and 2, is much higher than it is across random initializations, indicating robustness to initialization. Finally, on the LastFM dataset, on all approaches except HBWL , there is a standard deviation larger than the standard deviation across folds obtained in Table 1. This is likely because HBWL explores more weight configurations using smart resource allocation. We can conclude that while the search-based approaches are reasonably robust to initialization, this can depend on the dataset and the number of weight configurations they are allowed to try.

6.3.2 Impact of hyperparameter A

Here we evaluate the robustness of CRS , HBWL , and BOWL to the hyperparameter A in the Dirichlet distribution when sampling for the weight configurations. We chose four different values for A = $\{10, 1, 0.1, 0.01\}$ and evaluated on one discrete dataset, Citeseer, and one continuous dataset, Jester. For Citeseer, we use CA as the evalaution metric and MSE for the Jester dataset. We evaluate the three approaches, CRS , HBWL , and BOWL , that are impacted by A in PSL. Table 6 shows the metrics obtained for different values of A on both datasets and all methods. The effect of A on the Citeseer datasest is minimal in all methods. This is likely because the CA function w.r.t. weights is reasonably flat and small changes in weights have minimal impact. Therefore as long as there is at least one sampled point in a region, it is sufficient to get an optimal value and thus all approaches seem robust. Next, considering the Jester dataset, we see the parameter A has a large impact on CRS and HBWL . For a value of A = 0.1, CRS and HBWL perform the best, then, as it is increased to 10, both approaches produce the worst MSE value. This is because the space generated by the Dirichlet distribution using these parameter values is not representative of the true weight space for these two approaches. The impact of A is smaller on HBWL than CRS as HBWL explores more weight configurations via smart exploration. Finally,BOWL is shown to be robust to the parameter A as it uses an acquisition function to choose the next point.

Table 7 The table shows the effect of metrics obtained by using different acquisition function with BOWL

Datasets	Different acquisition functions			
	UCB	TS	PI	EI
Jester (MSE)	0.053	0.055	0.053	0.053
LastFM (MSE)	0.065	0.065	0.066	0.067
Citeseer (F1)	0.324	0.323	0.323	0.323
Cora (F1)	0.440	0.437	0.442	0.439
Epinions (F1)	0.711	0.712	0.713	0.713

BOWL is unaffected by both acquisition functions

6.3.3 Impact of acquisition function in BOWL

Our third experiment measures the robustness of BOWL to different acquisition functions. In Table 7 we compare the performance of BOWL using four different acquisition functions, UCB, TS, PI, and EI, for all five datasets in PSL (one metric per dataset). On all datasets and metrics, BOWL is relatively robust to acquisition function and performs similarly for all.

Finally, based on these experiments we observe that BOWL is the most robust to initialization, hyperparameter, and acquisition function and produces the best evaluation metric.

7 Conclusion and future work

In this paper, we introduced four approaches to learn weights in SRL: continuous random search, random grid search, Hyperband, and BOWL . To the best of our knowledge this, together with our prior work (Srinivasan et al. 2020b), is the first application of black-box algorithms for the task of weight learning in SRL. We proposed a novel projection that results in an efficient search over the best weight configuration that maximizes a user-defined evaluation metric. And finally, we showed that search-based approaches improved performance across several metrics on a variety of different real-world datasets. There are many avenues for expanding our work, including examining SRL approaches other than MLNs and PSL, avoiding full grounding, exploration of additional kernel functions, and structure learning. Here, we evaluated our new weight learning on two well-known SRL frameworks, MLNs and PSL; while we believe our approaches are generic enough to apply to other SRL frameworks, the effect of our proposed search-based approaches on the performance of other frameworks remains open for exploration. Further, to perform weight learning using search-based approaches, the SRL model needs to be fully grounded. There are a variety of approaches for avoiding full grounding (Sarkhel et al. 2015, 2016) that would be interesting to integrate into our approach. In addition, the performance of GP is highly dependent on the kernel function used. Therefore, in BOWL , which uses GP, an exploration of different kernels could further improve the performance. Finally, another enticing direction for future work is to apply search-based algorithms for identifying accurate dependency structures of SRL models, i.e., structure learning. Similar to weight learning, many existing approaches to structure learning aim to maximize a likelihood-based objective (Kok and Domingos

2005; Mihalkova and Mooney 2007; Natarajan et al. 2012; McCallum 2003; Shu et al. 2010). Extending the search-based methods proposed in this work to optimize problem-specific metrics rather than likelihood-based objectives may improve structure learning performance.

A. Appendix: Surface of a unit hypersphere as search space

Here, we discuss the effectiveness of the surface of a unit hypersphere as the search space in weight learning. To show that this is a reasonable space to sample weights from we need to show: first, the surface of the hypersphere is complete and non-redundant and second, if two weight configurations in OS are represented by the same weight configurations on the hypersphere then the solution obtained for $\mathbf{y}$ by both the weight configurations are the same. Every weight configuration given in OS can be easily projected on to the hypersphere in the following way:

$$\mathcal{H}(\mathbf{w}) = \frac{\mathbf{w}}{||\mathbf{w}||_2} \tag{20}$$

where $||\mathbf{w}||_2$ is the L-2 norm of the vector.

Theorem 5 *Every weight configuration* $\mathbf{w}$ *in OS has an equivalent weight configuration* $\tilde{\mathbf{w}}$ *on the hypersphere such that* $\operatorname{argmax}_{\mathbf{y}} \hat{s} \cdot \mathbf{E}(\mathbf{y}|\mathbf{x}, \mathbf{w}) = \operatorname{argmax}_{\mathbf{y}} \hat{s} \cdot \mathbf{E}(\mathbf{y}|\mathbf{x}, \tilde{\mathbf{w}})$.

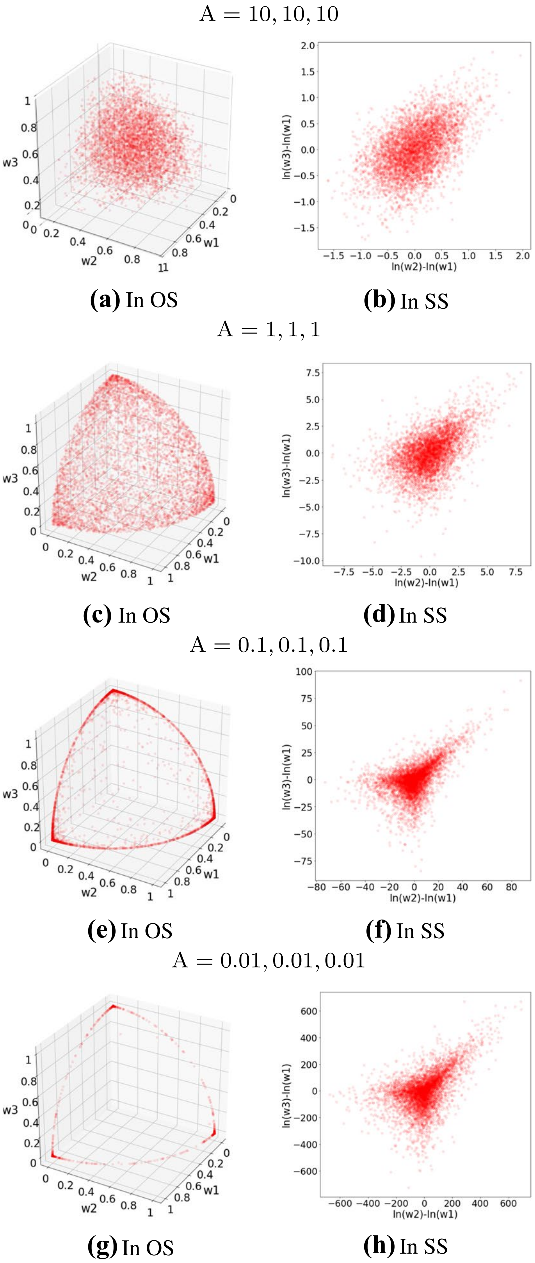
Proof From Theorem 1 we know that weights are scale invariant when performing MAP inference and this implies that the magnitude of the vector generated by a weight configuration does not affect the solution obtained when performing inference in PSL and MLNs. This implies that all r-dimensional unit vectors in the positive quadrant is sufficient to represent all possible weight configuration for MAP inference in PSL and MLNs. All unit vectors can be represented using the surface of a hypersphere. Therefore, surface of an r-dimensional hypersphere removes redundancies of OS and is complete. □

Theorem 6 *Given two weight configurations* $\mathbf{w}_1$ *and* $\mathbf{w}_2$ *such that the projection* $\mathcal{H}$ *of the weight configurations are equal, i.e.,* $\mathcal{H}(\mathbf{w}_1) = \mathcal{H}(\mathbf{w}_2)$, *then the optimization solution obtained for* $\mathbf{y}$ *by performing inference is the same.*

Proof This is easy to show as the definition of the projection defined in Eq. 20 is very similar to the projection defined in Theorem 1. The weights are simply rescaled and therefore, by definition and from Theorem 1, it is clear to see that if $\mathcal{H}(\mathbf{w}_1) = \mathcal{H}(\mathbf{w}_2)$ then $\operatorname{argmax}_{\mathbf{y}} \hat{s} \cdot \mathbf{E}(\mathbf{y}|\mathbf{x}, \mathbf{w}_1) = \operatorname{argmax}_{\mathbf{y}} \hat{s} \cdot \mathbf{E}(\mathbf{y}|\mathbf{x}, \mathbf{w}_2)$. □

This shows that the projection on to the surface of a unit hypersphere has properties similar to SS. However, these two are not entirely equivalent as the grounding factor $\boldsymbol{\kappa}$ plays an important role in the actual impact of weights at the time of inference. While SS is ideal and distances between weights are preserved even after adjusting for grounding, the distances are not preserved after adjusting for grounding.

Fig. 4 Visualization of Dirichlet distribution with different A of a model with three rules. Visualization shown both in OS and SS



$\mathrm{A} = 10, 10, 10$

(a) In OS **(b)** In SS

$\mathrm{A} = 1, 1, 1$

(c) In OS **(d)** In SS

$\mathrm{A} = 0.1, 0.1, 0.1$

(e) In OS **(f)** In SS

$\mathrm{A} = 0.01, 0.01, 0.01$

(g) In OS **(h)** In SS

Theorem 7 *Given two weight configurations* $\mathbf{w}_1$ *and* $\mathbf{w}_2$ *and their grounding adjusted weights* $\boldsymbol{\kappa} \cdot \mathbf{w}_1$ *and* $\boldsymbol{\kappa} \cdot \mathbf{w}_2$ (*an element-wise dot*), *the distance between the two weights before and after grounding adjustment are not the same*, *i.e.*, $||\mathbf{w}_1 - \mathbf{w}_2|| \neq \boldsymbol{\kappa} \cdot \mathbf{w}_1 - \boldsymbol{\kappa} \cdot \mathbf{w}_2$.

Therefore, we can conclude that while the surface of a hypersphere does not have all the properties of SS it is a reasonable approximation of SS and weights can be samples from the hypersphere.

A.1. Density of samples generated by projecting points from probability simplex

While sampling from a hypersphere with different densities can be tricky one easy way to do this is to sample from a Dirichlet distribution which samples from the probability simplex and project the values on to the sphere. The Dirichlet distribution accepts the hyperparameter $\mathrm{A} \in \mathbb{R}^{+r}$ which controls the spread of the distribution. It is easy to see that every point on a probability simplex can be uniquely projected on to a hypershpere. Here in Fig. 4 we visualize the distribution generated by using different values of A for a model with three rules. We vary the value of alpha form all 10s to all 0.001 and plot the projection on to a sphere which is in OS on the left and SS (which is 2-dimensional space) on the right. We can observe that as the value of A is reduced the samples in OS are getting concentrated to the poles and in SS we see the samples spread wider and choosing larger ratios. We can choose this value based on our application. For instance when the task is to perform rule pruning, it is more desirable for the ratio of weights to be more extreme and hence choose a small value for A and if the task is to fine tuning weights without making large changes, then a higher value of A might be more suitable.

Acknowledgements This work was partially supported by the National Science Foundation Grants CCF-1740850, CCF-2023495, and IIS-1703331. Golnoosh Farnadi was supported by postdoctoral scholarships from IVADO through the Canada First Research Excellence Fund (CFREF) grant.

References

Ahmadi, B., Kersting, K., & Natarajan, S. (2012). Lifted online training of relational models with stochastic gradient methods. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*.

Alshukaili, D., Fernandes, A. A. A., & Paton, N. W. (2016). Structuring linked data search results using probabilistic soft logic. In *The International Semantic Web Conference*.

Bach, S. H., Broecheler, M., Huang, B., & Getoor, L. (2017). Hinge-loss Markov random fields and probabilistic soft logic. *Journal of Machine Learning Research, 18*, 109:1-109:67.

Bach, S. H., Huang, B., London, B., & Getoor, L. (2013). Hinge-loss Markov random fields: Convex inference for structured prediction. In *The Conference on Uncertainty in Artificial Intelligence*.

Beltagy, I., Chau, C., Boleda, G., Garrette, D., Erk, K., & Mooney, R. (2013). Montague meets Markov: Deep semantics with probabilistic logical form. In *Second Joint Conference on Lexical and Computational Semantics*.

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research, 13*, 281–305.

Bergstra, J. S., Bardenet, R., Bengio, Y., & Kégl, B. (2011). Algorithms for hyper-parameter optimization. In *The Neural Information Processing Systems*.

Besag, J. (1975). Statistical analysis of non-lattice data. *Journal of the Royal Statistical Society, 24*, 179–195.

Boyd, S. P., Parikh, N., Chu, E., Peleato, B., & Eckstein, J. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. Foundations and trends. *Machine Learning*.

Brochu, E., Brochu, T., & de Freitas, N. (2010). A Bayesian interactive optimization approach to procedural animation design. In *The ACM Special Interest Group on Computer Graphics and Interactive Techniques*.

Chen, H., Ku, W., Wang, H., Tang, L., & Sun, M. (2017). Scaling up Markov logic probabilistic inference for social graphs. *IEEE Transactions on Knowledge and Data Engineering, 29*(2), 433–445.

Choi, J., Choi, C., Lee, E., & Kim, P. (2015). Markov logic network based social relation inference for personalized social search. In *New trends in computational collective intelligence* (pp. 195–202). Springer.

Chou, L., Sarkhel, S., Ruozzi, N., & Gogate, V. (2016). On parameter tying by quantization. In *The Association for the Advancement of Artificial Intelligence*.

Chowdhury, R., Srinivasan, S., & Getoor, L. (2020). Joint estimation of user and publisher credibility for fake news detection. In *The Conference on Information and Knowledge Management*.

Claesen, M., & De Moor, B. (2015). Hyperparameter search in machine learning. arXiv:1502.02127.

Collins, M. (2002). Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Empirical Methods in Natural Language Processing*.

Das, M., Dhami, D. S., Kunapuli, G., Kersting, K., & Natarajan, S. (2019). Fast relational probabilistic inference and learning: Approximate counting via hypergraphs. In *The Association for the Advancement of Artificial Intelligence*.

Das, M., Wu, Y., Khot, T., Kersting, K., & Natarajan, S. (2016). Scaling lifted probabilistic inference and learning via graph databases. In *SIAM International Conference on Data Mining*.

De Raedt, L., & Kersting, K. (2011). Statistical relational learning. In *Encyclopedia of machine learning* (pp. 916–924). Springer.

De Raedt, L., Kimmig, A., & Toivonen, H. (2007). Problog: A probabilistic prolog and its application in link discovery. In *The International Joint Conference on Artificial Intelligence*.

Ebrahimi, J., Dou, D., & Lowd, D. (2016). Weakly supervised tweet stance classification by relational bootstrapping. In *Empirical Methods in Natural Language Processing*.

Farabi, K. M. A., Sarkhel, S., & Venugopal, D. (2018). Efficient weight learning in high-dimensional untied mlns. In *Society for Artificial Intelligence and Statistics*.

Farnadi, G., Bach, S. H., Moens, M., Getoor, L., & Cock, M. D. (2017). Soft quantification in statistical relational learning. *Machine Learning Journal*.

Fierens, D., Van den Broeck, G., Renkens, J., Shterionov, D., Gutmann, B., Thon, I., Janssens, G., & De Raedt, L. (2015). Inference and learning in probabilistic logic programs using weighted boolean formulas. *Theory and Practice of Logic Programming, 15*(3), 358–401.

Friedman, N., Getoor, L., Koller, D., & Pfeffer, A. (1999). Learning probabilistic relational models. In *The International Joint Conference on Artificial Intelligence*.

Genton, M. (2001). Classes of kernels for machine learning: A statistics perspective. *Journal of Machine Learning Research, 2*, 299–312.

Getoor, L., & Taskar, B. (2007). *Introduction to statistical relational learning*. The MIT Press.

Goldberg, K., Roeder, T., Gupta, D., & Perkins, C. (2001). Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval Journal, 4*, 133–151.

Huynh, T. N., & Mooney, R. (2009). Max-margin weight learning for Markov logic networks. In *The ACM Special Interest Group on Knowledge Discovery and Data Mining*.

Huynh, T. N., & Mooney, R. J. (2010). Online max-margin weight learning with Markov logic networks. In *The Association for the Advancement of Artificial Intelligence*.

Islam, M. M., Mohammad Al Farabi, K., Sarkhel, S., & Venugopal, D. (2018). Scaling up inference in mlns with spark. In *Big data*.

Jaeger, M. (1997). Relational Bayesian networks. In *The Conference on Uncertainty in Artificial Intelligence*.

Joachims, T., Finley, T., & Yu, C.-N.J. (2009). Cutting-plane training of structural svms. *Machine Learning Journal, 77*, 27–59.

Johnson, K., Lee, I., & Goldwasser, D. (2017). Ideological phrase indicators for classification of political discourse framing on twitter. In *Workshop on NLP and Computational Social Science (NLP+CSS) at Association for ComputationalLinguistics*. https://aclanthology.org/venues/nlpcss/.

Kautz, H., Selman, B., & Jiang, Y. (1996). A general stochastic approach to solving problems with hard and soft constraints. In *The Satisfiability Problem: Theory and Applications*.

Khot, T., Balasubramanian, N., Gribkoff, E., Sabharwal, A., Clark, P., & Etzioni, O. (2015). Exploring Markov logic networks for question answering. In *Empirical Methods in Natural Language Processing*.

Kok, S., & Domingos, P. (2005). Learning the Structure of Markov Logic Networks. In *The International Conference on Machine Learning*.

Kouki, P., Fakhraei, S., Foulds, J., Eirinaki, M., & Getoor, L. (2015). Hyper: A flexible and extensible probabilistic framework for hybrid recommender systems. In *RecSys*.

Kouki, P., Pujara, J., Marcum, C., Koehly, L. M., & Getoor, L. (2017). Collective entity resolution in familial networks. In *The IEEE International Conference on Data Mining*.

Kushner, H. J. (1964). A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering, 86*(1), 97–106.

Lacoste-Julien, S., Jaggi, M., Schmidt, M., & Pletscher, P. (2013). Block-coordinate Frank–Wolfe optimization for structural svms. In *The International Conference on Machine Learning*.

Lalithsena, S., Perera, S., Kapanipathi, P., & Sheth, A. P. (2017). Domain-specific hierarchical subgraph extraction: A recommendation use case. In *Big data*.

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., & Talwalkar, A. (2018). Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research, 18*, 1–52.

Lizotte, D., Wang, T., Bowling, M., & Schuurmans, D. (2007). Automatic gait optimization with Gaussian process regression. In *The International Joint Conference on Artificial Intelligence*.

Lowd, D., & Domingos, P. (2007). Efficient weight learning for Markov logic networks. In *The ACM Special Interest Group on Knowledge Discovery and Data Mining*.

Marsaglia, G. (1972). Choosing a point from the surface of a sphere. *Annals of Mathematical Statistics, 43*(2), 645–646.

Martinez-Cantin, R., de Freitas, N., Brochu, E., Castellanos, J. A., & Doucet, A. (2009). A Bayesian exploration–exploitation approach for optimal online sensing and planning with a visually guided mobile robot. *Autonomous Robots, 27*(2), 93–103.

Matérn, B. (1960). *Spatial variation*. Springer.

McCallum, A. (2003). Efficiently inducing features of conditional random fields. In *The Conference on Uncertainty in Artificial Intelligence*.

Mehran Kazemi, S., Buchman, D., Kersting, K., Natarajan, S., & Poole, D. (2014). Relational logistic regression. In *The Association for the Advancement of Artificial Intelligence*.

Mihalkova, L., & Mooney, R. (2007). Bottom-up learning of Markov logic network structure. In *The International Conference on Machine Learning*.

Mockus, J. (1977). On Bayesian methods for seeking the extremum and their application. In *IFIP congress*.

Mockus, J., Tiesis, V., & Zilinskas, A. (1978). The application of Bayesian methods for seeking the extremum. In *Towards Global Optimisation*.

Muller, M. E. (1959). A note on a method for generating points uniformly on n-dimensional spheres. *Communications of the ACM, 2*(4), 19–20.

Natarajan, S., Khot, T., Kersting, K., Gutmann, B., & Shavlik, J. (2012). Gradient-based boosting for statistical relational learning: The relational dependency network case. *Machine Learning Journal*

Neville, J., & Jensen, D. (2007). Relational dependency networks. *Journal of Machine Learning Research, 8*, 653–692.

Niu, F., Ré, C., Doan, A., & Shavlik, J. W. (2011). Tuffy: Scaling up statistical inference in Markov logic networks using an rdbms. *Very Large Data Bases, 4*, 373–384.

Noessner, J., Niepert, M., & Stuckenschmidt, H. (2013). Rockit: Exploiting parallelism and symmetry for map inference in statistical relational learning. In *The Association for the Advancement of Artificial Intelligence*.

Platanios, E., Poon, H., Mitchell, T. M., & Horvitz, E. J. (2017). Estimating accuracy from unlabeled data: A probabilistic logic approach. In *The Neural Information Processing Systems*.

Poole, D. (1993). Probabilistic horn abduction and Bayesian networks. *Artificial Intelligence, 64*, 81–129.

Poon, H., & Domingos, P. (2006). Sound and efficient inference with probabilistic and deterministic dependencies. In *The Association for the Advancement of Artificial Intelligence*.

Rasmussen, C. E., & Williams, C. K. I. (2005). *Gaussian processes for machine learning (adaptive computation and machine learning)*. The MIT Press.

Richardson, M., & Domingos, P. M. (2006). Markov logic networks. *Machine Learning Journal, 62*(1–2), 107–136.

Sarkhel, S., Singla, P., & Gogate, V. (2015). Fast lifted map inference via partitioning. In *The Neural Information Processing Systems*.

Sarkhel, S., Venugopal, D., Pham, T. A., Singla, P., & Gogate, V. (2016). Scalable training of Markov logic networks using approximate counting. In *The Association for the Advancement of Artificial Intelligence*.

Sato, T. (1995). A statistical learning method for logic programs with distribution semantics. In *International Conference on Logic Programming*.

Schölkopf, B., & Smola, A. (2002). *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. MIT Press.

Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., & de Freitas, N. (2016). Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE, 104*(1), 148–175.

Shavlik, J., & Natarajan, S. (2009). Speeding up inference in Markov logic networks by preprocessing to reduce the size of the resulting grounded network. In *The International Joint Conference on Artificial Intelligence*.

Shu, J., Lao, N., & Xing, E. (2010). Grafting-Light: Fast, Incremental Feature Selection and Structure Learning of Markov Random Fields. In *The ACM Special Interest Group on Knowledge Discovery and Data Mining*.

Singla, P., & Domingos, P. (2005). Discriminative training of Markov logic networks. In *The Association for the Advancement of Artificial Intelligence*.

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical Bayesian optimization of machine learning algorithms. In *The Neural Information Processing Systems*.

Sridhar, D., Fakhraei, S., & Getoor, L. (2016). A probabilistic approach for collective similarity-based drug–drug interaction prediction. *Bioinformatics, 32*(20), 3175–3182.

Srinivas, N., Krause, A., Kakade, S., & Seeger, M. (2010). Gaussian process optimization in the bandit setting: No regret and experimental design. In *The International Conference on Machine Learning*.

Srinivas, N., Krause, A., Kakade, S. M., & Seeger, M. W. (2012). Information-theoretic regret bounds for Gaussian process optimization in the bandit setting. *IEEE Transactions on Information Theory, 58*, 3250–3265.

Srinivasan, S., Augustine, E., & Getoor, L. (2020a). Tandem inference: An out-of-core streaming algorithm for very large-scale relational inference. In *The Association for the Advancement of Artificial Intelligence*.

Srinivasan, S., Farnadi, G., & Getoor, L. (2020b). BOWL: Bayesian optimization for weight learning in probabilistic soft logic. In *The Association for the Advancement of Artificial Intelligence*.

Srinivasan, S., Rao, N., Subbian, K., & Getoor, L. (2019). Identifying facet mismatches in search via micrographs. In *The Conference on Information and Knowledge Management*.

Taskar, B., Abbeel, P., & Koller, D. (2002). Discriminative probabilistic models for relational data. In *The Conference on Uncertainty in Artificial Intelligence*.

Thompson, W. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika, 25*(3–4), 285–294.

Van Haaren, J., Van den Broeck, G., Mert, W., & Davis, J. (2015). Lifted generative learning of Markov logic networks. *Machine Learning Journal*.

Venugopal, D., Sarkhel, S., & Gogate, V. (2016). *Magician: Scalable inference and learning in Markov logic using approximate symmetries*. UofM, Memphis: Technical report.

Wang, Z., Hutter, F., Zoghi, M., Matheson, D., & De Freitas, N. (2016). Bayesian optimization in a billion dimensions via random embeddings. *Journal of Artificial Intelligence Research, 55*(1), 361–387.

Authors and Affiliations

Sriram Srinivasan[1] · Charles Dickens[1] · Eriq Augustine[1] · Golnoosh Farnadi[2] · Lise Getoor[1]

Charles Dickens
cdickens@ucsc.edu

Eriq Augustine
eaugusti@ucsc.edu

Golnoosh Farnadi
farnadig@mila.quebec

Lise Getoor
getoor@ucsc.edu

[1] UC Santa Cruz (CSE), Santa Cruz, CA, USA

[2] Mila, Universite de Montreal, Montreal, QC, Canada