INSTITUT FÜR INFORMATIK
der Ludwig-Maximilians-Universität München

Evaluating parameter scaling in single-target learning of Markov Logic Networks with respect to varying domains

Ramona Fabry

Bachelor thesis

Supervisor	Prof. Dr. François Bry
Mentor	Dr. Felix Weitkämper
Date of submission	19.08.2022

Statement

I hereby confirm that I have written the accompanying thesis by myself, without contributions from any sources other than those cited in the text and acknowledgements.

Munich, 19.08.2022 Ramona Fabry

Abstract

Markov logic networks have a commonly known problem: Their unsatisfying behavior for increasing domain sizes. The probabilities they predict for common queries tend to converge to constants independent of their actual weights of Markov logic networks. To overcome this problem, *Mittal* [3] introduced Domain Size aware Markov logic networks. *Ravdin* [8] provides an approach where scaling the weights of Markov logic networks is combined with boosted learning. His approach is presented by learning on samples of popular domains and applies the trained model on the original domains. The results show an improvement in run time, while also having good results in his observed metrics. We want to investigate if this approach strikes similar results when using it on different domain structures. To model such changing structures, we used a for our purposes modified data set called "Friends and Smokers". Our experiments show that this boosted learning approach fails in some cases. We build two types of domains where the data is split into isolated groups. When scaling such data the group structures behave differently: for one domain the groups increase in size, for the other one the group size remains constant but the number of groups grows. We observe how these differences influences the quality of the model predictions. In the latter case, we show that the predictions are not desirable. In contrast, the outcome we get for the domains with growing groups is better. It therefore follows that it is strongly relevant how the domains relations change when changing domain sizes. For domains with a growing number of isolated relation groups using DA-MLN is generally not recommended. Such data may be collected by non-probability sampling techniques such Snowball sampling. However, for scaling domains where the data has been obtained from a random sample, we can confirm the good result *Ravdin* and *Mittal* describe in their work.

Zusammenfassung

Markov-Logik-Netzwerke haben ein allgemein bekanntes Problem: ihr Verhalten bei zunehmender Domänengröße. Die Wahrscheinlichkeiten, die sie für allgemeine Abfragen vorhersagen, tendieren dazu,unabhängig von den tatsächlichen Gewichtungen von Markov-Logiknetzwerken zu Konstanten zu konvergieren. Um dieses Problem zu lösen, wurden von *Mittal* [3] Domänenabhängige Markov-Logik-Netzwerke eingeführt. *Ravdin* [8] bietet einen Ansatz, bei dem die Skalierung der Gewichte von Markov-Logik-Netzwerken mit einer laufzeit-optimierten Lernmethode kombiniert wird. in seinem Ansatz wird auf Teilmengen bekannter Domänen geleernt und dieses Modell wird auf die orginal Datensätze angewendet. Die Ergebnisse zeigen eine Verbesserung der Laufzeit bei gleichzeitig guten Ergebnissen der beobachteten Metriken. Wir wollen untersuchen, ob dieser Ansatz auch bei der Verwendung auf verschiedenen Domänenstrukturen zu ähnlichen Ergebnissen führt. Um solche sich verändernden Strukturen zu modellieren, haben wir einen für unsere Zwecke modifizierten Datensatz namens "Friends and Smokers" verwendet. Unsere Experimente zeigen, dass dieser Boosted-Learning-Ansatz in einigen Fällen fehlschlägt. Wir konstruieren zwei Arten von Domänen, in denen die Daten in isolierte Gruppen aufgeteilt werden. Beim Skalieren solcher Daten verhalten sich die Gruppenstrukturen unterschiedlich: Für die eine Domäne werden die Gruppen größer, für die andere bleibt die Gruppengröße konstant, aber die Anzahl der Gruppen wächst. Wir beobachten, wie dieser Unterschied die Qualität der Modellvorhersagen beeinflusst. Für den letzteren Fall von Domänen zeigen wir, dass die Vorhersagen unbefriedigend sind. Im Gegensatz dazu ist das Ergebnis, das wir für die Domänen mit wachsenden Gruppen erhalten, besser. Daraus folgt, dass es stark relevant ist, wie sich die Domänen-Relationen ändern, wenn sich die Domänen-Größe ändert.Für Domänen mit einer wachsenden Anzahl isolierter Beziehungsgruppen ist die Verwendung von DA-MLN generell nicht zu empfehlen. Solche Datensets können durch Stichprobenverfahren gesammelt werden, z.B durch das Schneeballverfahren. Für Domänen, bei denen die Daten aus einer zufälligen Stichprobe stammen, können wir jedoch die gtuen Ergebnisse bestätigen, die *Ravdin* und *Mittal* in ihren Arbeiten beschreiben.

Contents

CHAPTER 1

Introduction

In recent years, it gets easier and easier to collect huge amounts of statistical data. This results in huge collections of relational data, where statistics can be extracted and on which models can learn on. This amount of data is very costly to evaluate all at once. In statistical relational artificial intelligence (StarAI) there are many different statistical learning methods, which may differ in their models (directed or undirected). Markov logic networks are one among such models. What these methods have in common is a complexity with increasing domain size. To overcome this problem, one may try to learn on small training domains, to reduce computation costs. This model then can be applied to real world data. This also can be an approach to use, when the initially collected data sets may be to small because of a lack of data. One aspect that needs to be considered with such an approach is the change of relations within the data set between larger and smaller collections.

Another aspect that arises when collecting data is the structure of a domain. As an example for collecting data with varying structures, let us consider a school. We distinguish between two methods of taking samples by choosing students and look at their relations between other students. For the first method, we sample a number X of complete school classes. For the second method we sample the same number of students from X different classes. Pupils from the same class usually know each other; they can be considered as one large community. In contrary, in a data set where the pupils are taken from different classes, they are much less likely know each other. Here the sample of each class represents one community, which leads to many small communities. The way how the data collected from such a setup leads to variation in domains: You either focus on collecting data by sampling communities (complete classes) or by sampling representatives from different communities (students from every class). We talk more about domain structures in chapters 3 and 4. We investigate the impact of such different domain structures on learning Markov logic networks and applying them on domains of different sizes.

This thesis is structured as follows: we first give an overview of some important definitions and explain theories and concepts, necessary for the approach used in this work. To understand the functionality of Markov logic networks, the notation of first-order logic is defined as well as the boosted learning approach from *Khot et al.* [2]. Next, the Domain-Size Aware Markov logic networks, introduced by *Mittal* [3] are explained. In the following experiments, we use Domain-Size Aware Markov logic networks as well as boosted Markov logic networks. The next chapter describes our hypothesis. We theoretically explain how we expect the previously presented boosted DA-MLN to act on two different domain struc-

tures and give an example. Following on from that we describe our experimental setup. This includes the algorithm for creating domains as well as the configuration for the Boost-SRL program. To measure the quality of our results we explain some metrics we used. The following chapter describes the results we got from our experiments and describe the resulting graphs. Subsequently, these results are discussed and interpreted in terms of their different setups and our conclusion is provided.

CHAPTER 2

Theory

F_1	F_2	$\neg F_1$	$(F_1 \wedge F_2)$	$(F_1 \vee F_2)$	$(F_1 \Rightarrow F_2)$
true	true	false	true	true	true
true	false	false	false	true	false
false	true	true	false	true	true
false	false	true	false	false	true

Table 2.1: Truth table for F_1 and F_2

2.1 First-order logic

Markov logic networks were first introduced by *Richardson and Domingos* [9] and are based on first-order logic (FOL). For the purpose of this work it is sufficient to define only a subset of FOL, the *quantifier and function free* FOL, which is similar to the subset used by *Mittal et al.*[3].For our exposition we follow *Fitting et al.* [1] defined.

Quantifier and function free FOL contains *constants*, *variables* and *predicate symbols*. Variables are denoted by small Latin letters (e.g. x, y). Boolean relations between objects are expressed by predicate symbols (e.g. friends(Person1,Person2). A *relational language* L is determined by specifying

1. A finite or countable set **R** of function symbols or predicate symbols **P**, each of which has a positive integer */n* associated with it. We say n is the arity of P.

2. A finite or countable set C of constants.

We use the notation $\mathscr{L}$ for the first-order language determined by R and C. A *model* for the language $\mathscr{L}$ is a pair $M = \langle D, I \rangle$. D is a non-empty set of constants called *domain* or *data set* and we think of it as concrete objects (e.g. Alice, Person1, etc.). I is the process of mapping a variable to a constant (e.g. x = Person1,...). This is called *grounding*. It therefore follows, that a ground atom or ground predicate is an atomic formula, where all of those arguments are constants. Assigning a truth value to each possible ground atom yields a so called *interpretation* or *world* ω. An expression representing an object in our domain is called *term*, which can either be a *constant* or a *variable*. Following on from that, a predicate symbol applied to one or more terms is called an *atom*. A *literal* is an *atom* or its negation, called *positive literal* or *negative literal* respectively.

A formula contains one or more literals and can be constructed in a recursive manner. Every atom is a formula. If F_1 and F_2 are formulas, so is

- the negation $\neg F_1$,
- the conjunction $(F_1 \wedge F_2)$
- the disjunction $(F_1 \vee F_2)$
- the implication $(F_1 \Rightarrow F_2)$,

The truth values for each formula can be see in Table 2.1.

As an example, suppose the language $\mathscr{L}$ has relational symbols R={smokes/1, friends/2} and an empty set of constants C={}. The model $M = \langle D, I \rangle$ has a domain D={anna,bob} and an interpretation I such that x=anna, y=bob.

Then a possible world looks like the following:

$$
\begin{aligned}
&\textit{friends(anna, bob)} = \textit{true}\\
&\textit{friends(bob, anna)} = \textit{true}\\
&\textit{friends(anna, anna)} = \textit{false}\\
&\textit{friends(bob, bob)} = \textit{false}\\
&\textit{smokes(anna)} = \textit{true}\\
&\textit{smokes(bob)} = \textit{true}
\end{aligned}
$$

In the following context formulas will appear in the notation of *Horn clauses*. The head of a Horn clause contains a single literal, the target. In the body are 1,..,n literals that are denoted as a conjunction:

$$literal_1 \wedge literal_2 \wedge ... \wedge literal_n \Rightarrow target.$$

2.2 Markov logic networks

Markov logic networks (MLNs) combine probabilities and first-order logic. A MLN is a set of weighed formulas given as $w_i : F_i$, where F_i is a formula and w_i a corresponding real number weight. Clauses with higher weights have more impact on the resulting probability. An example of an MLN formula is:

$$1.2 : smokes(A) \wedge friends(A,B) \Rightarrow smokes(B) \tag{1}$$
$$0.2 : friends(A,B) \wedge friends(B,C) \wedge smokes(C) \Rightarrow smokes(A) \tag{2}$$

In this example the formula (1) means "If A smokes and A and B are friends, then B smokes.". Formula (2) means "If a friend of a friend smokes, the person also smokes" Thinking about this example intuitively, one may say that the impact of a smoking friend on our own smoking habits is higher than the impact of a distant friend. This is also expressed through the weights of the formula. (1) has a higher weight, therefore the instance has more effect on the resulting probability than (2).

From a graphical point of view, MLNs are presented in undirected graphs, where every node represents an atom with its variable. Every edge between a node represents the relation between these two atoms. These edges only exist, when the involved atoms appear in a formula together. Given different domains, an MLN produces different networks, which also can vary in size, but all of them will have similar patterns in their structure and parameters (e.g. all groundings of the same formula will have the same weight). These networks are constructed by grounding all atoms and considering all satisfied formulas of the MLN. All of those atoms included in the satisfied formulas form the nodes and their corresponding connections. This we call a *ground Markov network*. The corresponding Markov network to formula (1) in our given world from chapter 2.1 can be seen in Figure 2.1. The concrete probability distribution for possible interpretations specified by the ground Markov networks is given by:

$$P(X = \omega) = \frac{1}{Z} \exp(\sum_i w_i n_i(\omega)) \qquad \text{with } Z = \sum_X \exp(\sum_i w_i n_i(\omega)) \tag{3}$$

where w_i denotes the weight of a formula, $n_i(\omega)$ is the number of true groundings of F in ω, ω is an interpretation, i is an iterator over the number of FOL formulas and Z is a normalization constant to ensure, that all probabilities sum up to 1.

2.2.1 Downsides of Markov logic networks

When learning an MLN the goal is to learn rules for a given domain and attach a weight to every of these learned FOL formulas. The most approaches are compact and comprehensible and have a mathematical definition that ensures correctness. They efficiently compute models on smaller domains, but computational complexity is exponential in domain size.

Another problem of MLN is that they suffer from generalization issues when training and testing domains differ in their sizes. This results in a growth of probabilities when testing domains are larger than the training domains. *Mittal* [3] used as an example the prediction of an epidemic in a town. A single MLN formula

$$w : sick(x) \Rightarrow epidemic$$

is given where w is learned from some training data. When we now want to use this model to predict the probability of an epidemic in a larger town, it gets more and more likely to have an epidemic, the larger the town is. This is independent of the actual weight of our model but, rather due to the growing number of connections the query atom is involved in. This results in a wrong prediction, which is also not related to the actual domain.

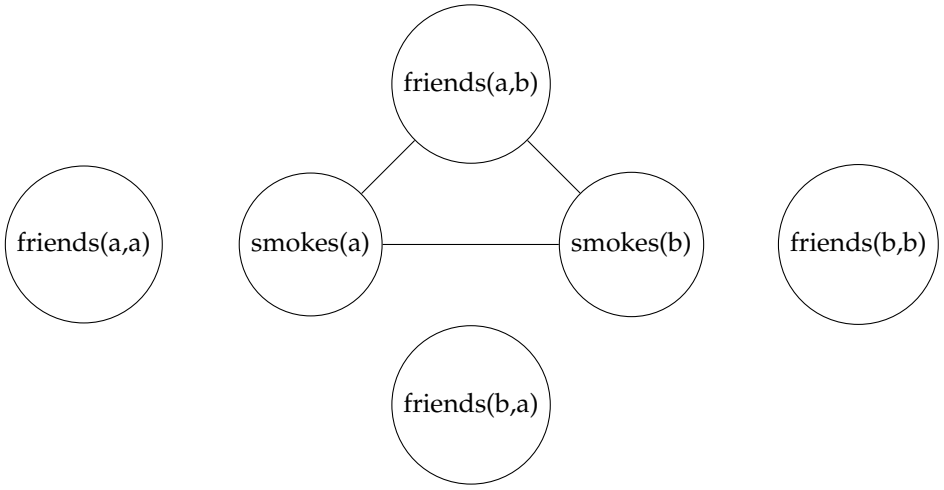


Figure 2.1: Graphical representation of a ground Markov network for formula (1) and a our world from chapter 2.1 with a=anna and b=bob

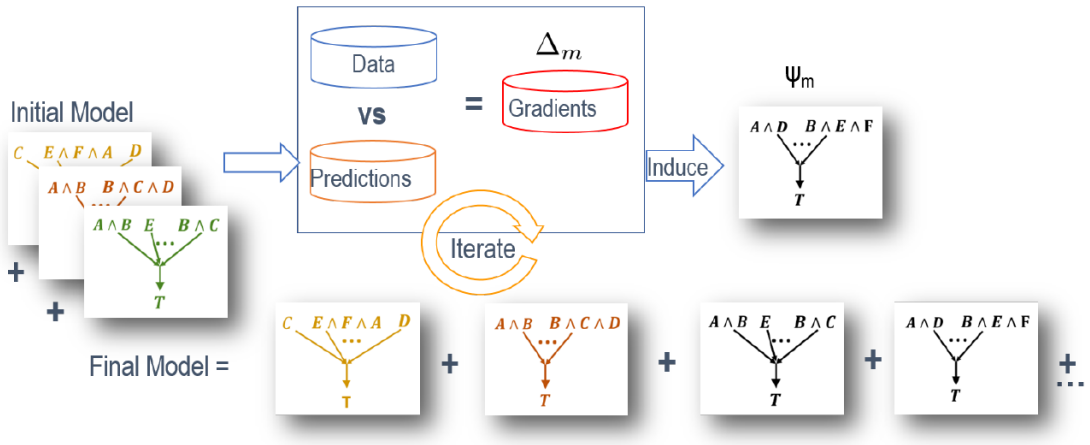


Figure 2.2: graphical representation of the boosting algorithm [7]

2.3 Boosted learning of Markov logic networks

Khot et al. [2] from StarlingAI lab at the university of Dallas provides an approach, where the learning time of MLNs is improved. The main idea of this algorithm is rather than learning huge, complex MLNs and then learn and attach the corresponding weights, smaller, simpler MLNs and their corresponding weights are learned simultaneously. These small MLN are then combined to a large MLN. Every added small MLN improves the expressiveness of the large combined MLN. This may be advantageous, because learning slim MLNs is faster than learning larger ones. For learning, *Khot* is making use of *functional gradient boosting*, represented in figure 2.2.

2.3.1 Functional gradient boosting

The intuition of this approach is to first roughly learn some rules which are then improved for better prediction during a number of iteration steps. Moreover, it is easier to learn a set of approximate rules than a highly accurate model. In every iteration step of the algorithm *inductive logic programming* (ILP) is performed to learn rules in FOL with an associated regression value, which are used to construct either *Relational Regression Trees* (RRT) or a set of *Relational Regression Clauses* (RRC). For our approach, we will focus on learning RRCs. The regression values of the RRCs are the weights of MLNs and are dependent on the number of groundings of the MLN. In general, such RRCs are learned on single targets. An general example for an RRC with the target predicate *target/1* may look like:

$$w_1 : target(X) \Leftarrow p(X) \wedge q(X,Y)$$
$$w_2 : target(X) \Leftarrow p(X)$$
$$w_3 : target(X)$$

Each RRC produced in an iteration is added to either the initial or an already extended model. With the new model, predictions are tested with actual example data. The goal of each iteration is to minimize the error in predictions. It may also happen, that after adding a new RRC, the new predictions gets worse, but since every tree has only a small impact on the whole model this can be corrected by the next iteration step. The number of iteration steps must be previously specified.

2.3.2 BoostSRL program

The code base for the boosted learning of Markov logic networks can be found on Github [4]. The program is split in learning and testing, also called inference. In learning mode a distinction is made between tree-based and clause-based learning. Tree-based learning iterates through each gradient step and learns a regression tree, whereas in clause-bases mode clauses are learned instead of trees. The program also needs some background information containing the predicates and their possible modes, which will be explained in more detail in chapter 4. Then the program learns MLNs via functional gradient boosting and creates output trees as well as the learned MLNs. The output of the testing are AUC-PR, AUC-ROC and CLL values. A more detailed explanation of these metrics will be given in chapter 4. This approach is limited insofar as that it is only possible to test on the same domain size the model was trained on to get meaningful results.

2.4 Domain size aware Markov logic networks

In chapter 2.2 MLNs were introduced as having a problem: with increasing domain sizes, probabilities tend to converge to constants independent of their weights. Therefore, the predictions lose their relation to the initial MLNs. *Mittal* [3] introduced an approach where scaling weights of every MLN formula bypasses the convergence. This makes it possible to scale on various domain sizes without losing any expressiveness.

For scaling, the number of connections c of a target predicate P in every formula F of our MLN T needs to be known. This number increases with growing domains, hence this leads to an increasing sum of weights and ends in convergence of probabilities to constants. At first predicate occurrences of every F in T need to be counted. When we revisit our example from 2.2(1), a single occurrence for predicate *friends* is present, *smokes* can be counted twice. Thus, it is also possible to count multiple occurrences of a predicate in a single formula. The appearances are saved in *Preds(T)*. Afterwards, the number of connections c of each $P \in Preds(T)$ is needed. For this each variable that does not appear in P, but in the MLN as $Vars(P)^{-}$ is considered. The scaling down factor s_i is derived from c by aggregating those values with the *max* function, as *Mittal* suggests in his work. The scaled probability distribution is now defined as:

$$P(X = \omega) = \frac{1}{Z} \exp\left(\sum_i \frac{w_i}{s_i} n_i(\omega)\right) \tag{4}$$

where the used symbols are the same as in 2.2 and s_i denotes the scaling down factor of T_i. Also the normalization constant Z updates to

$$\sum_X \exp\left(\sum_i \frac{w_i}{s_i} n_i(\omega)\right)$$

2.5 Boosted domain size aware Markov logic networks

Previously, the approach of boosted learning of MLNs as well as scaling their weights to apply the learned MLNs on different domain sizes was presented. *Ravdin* [8] provides an extension of BoostSRL where the concept of DA-MLN is implemented. The scaling happens in clause-base mode, since it is much simpler to apply the scaling down factor in this mode. When learning MLNs the size of the domain is saved, by counting all different constants in the data set. Since it is only possible to learn on a single-target, just the number of connections of this target is needed. This is done by counting all predicates in the body of a Horn clause. When it comes to inference, the scaling factor s is defined as

$$s = \prod_{x \in Vars(P)^-} \frac{|\Delta_X|_{test}}{|\Delta_X|_{train}}$$

The results of his work show that this scaling behaves well on different domains, naming IMDB, WebKB and a specific configuration of Friends and Smokers.

CHAPTER 3

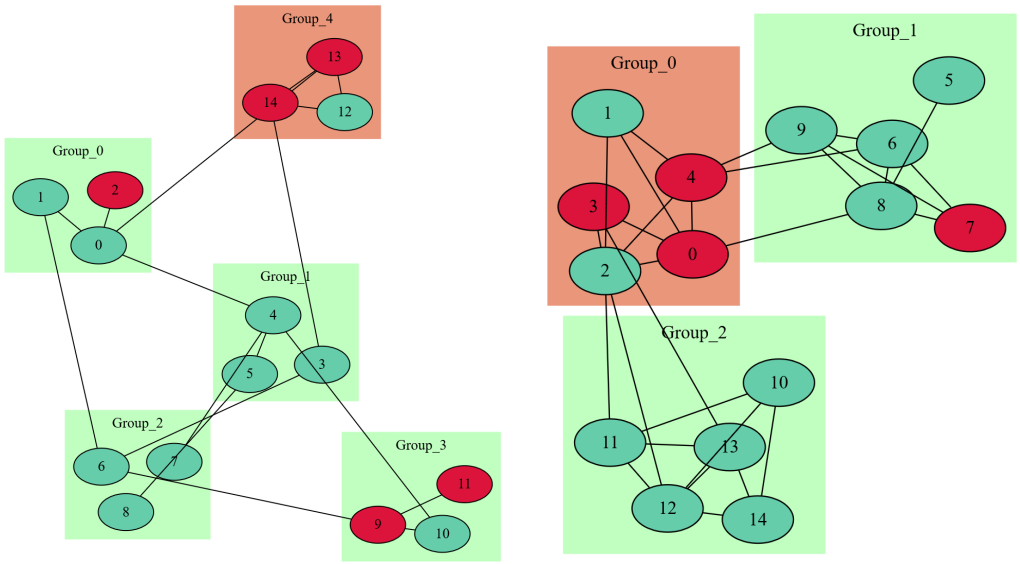
Hypothesis

This thesis investigates the effect of the introduced boosted scaling approach on varying domain structures. To observe the behavior, we perform experiments on a synthetic domain which we modify to model different domain structures. To explain what is meant by different structures, two example data sets are provided in figure 3.1, where the corresponding clause-files can be found in the appendix. We reconsider our example from chapter 1 to better illustrate the presented data. Data set (a) maps to the case where pupils are taken from a few classes, but the number of classes considered stays the same (in this case the number of classes is 5). This leads to growth within each group as the data set increases. In data set (b) we have a similar case as taking a sample of students from every class. There are more and more small communities (groups), from each of which a fixed amount of pupils is picked (here 5 students are picked). Red groups are labeled as "smoker" groups. In these groups the members are more likely smokers than members of non-smoker groups. Nodes represent different persons, whereby red nodes are smoker, green are non-smoker. We assume that it is more likely for two persons to be friends if they are in the same group, than in different groups.

For data set (a) let the number of groups remain constant at 5 with increasing domain sizes. Here we would expect that the total number of friendships increases with greater domain sizes. In contrast, if we look at data set (b) and fix the size of groups to 5 we would assume that the number of friendships does not grow as fast as we assume for data set (a), since the number of **likely** friends connections within the group stays constant at 4. We expect the scaling of these two data sets behaves different. For (a) the number of connections approximately matches the actual connections. Therefor the scaling works well. On the other hand the approximate number of connections for (b) is less than the expected connections, which leads to a debasement of out MLNs after scaling. In general the two main points we want to observe are:

(1) Learning of DA-MLN outperforms regular MLN on Domains with growing connection relations

(2) Learning of regular MLN outperforms DA-MLN on Domains with mainly constant connection relations



(a) Data set of size 15 with 5 groups

(b) Data set of size 15 with 5 persons per group

Figure 3.1: Example for domains varying in their structure

To show this theoretically *Weitkämper* [10] introduced the following formula, to model the probabilities of a mean of possible worlds independent of the weights of MLNs.:

$$\lim_{n\to\infty} P_{T_w,n}(Q(x)) = \lim_{n\to\infty} \int_{\mu_{T_w,n}} sigmoid(\delta_{Q(x)}^{T_{w,n}}) \tag{1}$$

Where the sigmoid(δ) is the weighted mean of true groundings in the considered possible worlds with $\omega = \frac{1}{Z}\exp(\frac{w_i}{s_i}n_i(\omega))$ the weighed probability distribution from formula 4 and Q(x) is the target predicate e.g *smokes(x)*.

As an example, let's assume the weight of every MLN is 1.0, since formula 1 is independent of weights. We also forbid friends connections outside a group to focus on the varying domains. Further, we assume the probability of being a smoker is 80%. For the domains, we consider a fixed training size of 10 and increase the testing domain. The scaling factor s is calculated as $s = \frac{trainsize=10}{testsize}$. Now, we distinguish between data sets with a fixed number of groups and a fixed size for groups like we did in figure 3.1.

In the first case we have a fixed number of 5 groups. The number of possible friends grows with increasing domain sizes and can be at a maximum group size - 1, which we use for every one of our example worlds in table 3.1. In the second case, where we have a fixed number of persons per group, the behavior when approaching infinity is different. While s_i is also growing with increasing domains, n_i stays constant between 0-4. Therefore, only the denominator grows, which makes the whole fraction smaller and approaches 0. The result *sigmoid(0)* = $\frac{1}{2}$ is not meaningful, since there is no correlation to the input values.

domain size	number of friends	smokers n_i	scaling factor s_i	$\delta = \frac{n_i}{s_i}$	*sigmoid*(n_i)	*sigmoid*(δ)
10	2	2	1	2	0,88	0,88
20	4	3	2	1,5	0,88	0,81
30	6	5	3	1,67	0,95	0,84
40	8	6	4	1,5	0,98	0,82
50	10	8	5	1,6	0,99	0,83
60	12	10	6	1,67	1,0	0,84
70	14	11	7	1,57	1,0	0,83
80	16	13	8	1,63	1,0	0,84
90	18	16	9	1,56	1,0	0,83
100	20	13	10	1,58	1,0	0,83

Table 3.1: Possible worlds for domains of size 10-100 and a group count of 5 and probability of 80% of being a smoker.

domain size	number of friends	smokers n_i	scaling factor s_i	$\delta = \frac{n_i}{s_i}$	*sigmoid*(n_i)	*sigmoid*(δ)
10	4	3	1	3	0,95	0,95
20	4	3	2	1,5	0,95	0,81
30	4	3	3	1	0,95	0,73
40	4	3	4	0,75	0,95	0,68
50	4	3	5	0,6	0,95	0,65
60	4	3	6	0,5	0,95	0,62
70	4	3	7	0,43	0,95	0,61
80	4	3	8	0,38	0,95	0,59
90	4	3	9	0,33	0,95	0,59
100	4	3	10	0,3	0,95	0,57

Table 3.2: Possible worlds for domains of size 10-100 and a group size of 5 and probability of 80% of being a smoker.

CHAPTER 4

Methods

We want to observe how a change in the structure of a domain affects the quality of scaling MLN. To model the varying domains, we changed the size of the training domain and keep the domain size for testing fixed to 100. We look at four different metrics, naming Area Under Curve of Receiver Operating Characteristic (AUC-ROC), Area Under Curve Precision and Recall (AUC-PR), Conditional Log Likelihood (CLL) and the run time, which we compare in different setups.

4.1 Data set "friends and smokers"

"Friends and smokers" is a synthetic data set, which models smoking habits of different people and their relationship to other smokers or non-smokers as well as their health status. We use it to learn and predict a person's smoking habit, depending on the smoking habits of their friends. Since this data set is synthetic, it comes in handy that it can be modified for our purpose. This means we can change the condition of two people being friends to create a data set with varying numbers of friends connections. This represents the changing structure in our data set we want to investigate.

Our created data set for "friends and smokers" is similar to the one *Mittal* [3] defined. We omit the predicate *cancer(a)* and just keep *smokes(a)* and *friends(a,b)*. As we are in single-target leaning, we choose *smokes(a)* as our target predicate. To be able to learn recursively, another predicate *s_smokes(a)* is introduced as well as *not_s_smokes(b)* accordingly. A person, who is labeled as *s_smokes(a)* is the same as a person labeled with *smokes(a)*. A person labeled with *not_s_smokes(b)* does not smoke. So it can be predicted how likely this person smokes, depending on the smoking habits of his friends.

Building the adjusted data set works as follows: First of all, we choose the size d for our domain. This data set is split into several groups which are randomly labeled either as a "smoker" or a "not smoker". The probability to be labeled as "smoker" is 40%. The number of groups g is either a fixed size $g = G$ or depends on the size s of groups $g = \frac{n}{s}$. The size of a group is either set to $s = \frac{d}{g}$ or a fixed size $s = S$. For every data set we either set a fixed group number (in figures often referred to as COUNT) $g = G$ or a fixed group size (in figures often referred to as SIZE) $s = S$. For setting the friends-relation it is considered if people are in the same group or not. People from the same group have a 80% chance to be friends. The probability of a friends-relation outside a group changes during our

experiments in a range from 0% to 30%. The smoking habits of each person depends on the label of his group. If someone is in a "smoker" group there is a 70% chance for this person to be a smoker, otherwise its 10%. Finally, we create a file with all previously collected facts: Friends relations, smokers, s_smokers and not_s_smokers. Information about the groups are not saved in this file. Groups can only be guessed through the number of smoking friends or non smoking friends respectively, since this is what we want to learn with our model. The script for creating data sets can be found on Github [6].

Algorithm 1 Friends and Smokers data set generator

if $group\_count != Null$ **then** ▷ create group sizes/number of members
  $group\_size = \frac{domain\_size}{group\_count}$
else $group\_size = \sqrt{domain\_size}$
end if
if $group\_size != Null$ **then**
  $group\_count = \frac{domain\_size}{group\_size}$
else $group\_size = \frac{domain\_size}{group\_count}$
end if
if 40% chance **then** ▷ randomly lable groups
  label = smoke
else label = not_smoke
end if
if persons in same group and 80% chance **then**
  friends(person_1, person_2)
else if persons not in same group and x% chance
  friends(person_1, person_2)
end if
if in group "smoke" and 70% chance **then** ▷ set smoking habits
  smokes(person)
  s_smokes(person)
else if 10% chance **then**
  smokes(person)
  s_smokes(person)
else not_s_smokes(person)
end if

We ran the script starting on a domain size of 10 for training and a domain size of 100 for testing. In the further course we increased the training domain size up to 100 in steps of 5. For each training size 10 repetitions were made where a new training and testing data set pair was created for each repetition. If the training of a model somehow failed in one step, e.g. because of too small data sets or too many connections, we retried learning in the same step with a new generated data set up to 30 times. From these 10 results we form Interdecile range (IDR) where the highest 10% of values and lowest 10% are discarded to get rid of extreme outliers. From the cleared up values we take the Median to get a representative value. This was repeated for a fixed group size and fixed group member number of 5 and 10. As mentioned, we use "friends and smokers" to vary the number of friends-connections, which we did by changing the probability of having a friend outside a group from 0%-30% In total we made 8 setups.

- fixed group size/group count of 5, with a probability of 0% to have a friend outside a group
- fixed group size/group count of 5, with a probability of 10% to have a friend outside

a group

- fixed group size/group count of 5, with a probability of 20% to have a friend outside a group
- fixed group size/group count of 5, with a probability of 30% to have a friend outside a group
- fixed group size/group count of 10, with a probability of 0% to have a friend outside a group
- fixed group size/group count of 10, with a probability of 10% to have a friend outside a group
- fixed group size/group count of 10, with a probability of 20% to have a friend outside a group
- fixed group size/group count of 10, with a probability of 30% to have a friend outside a group

For results, where the group size/count was 10, we omitted every second value, since it is not always possible to divide without rest. All experiments were run on the virtual machine "Vanuabalavu" with 32 Cores and 228 GB RAM from the Institute for Informatics of the LMU Munich. The results will be presented in chapter 5 and discussed in chapter 6.

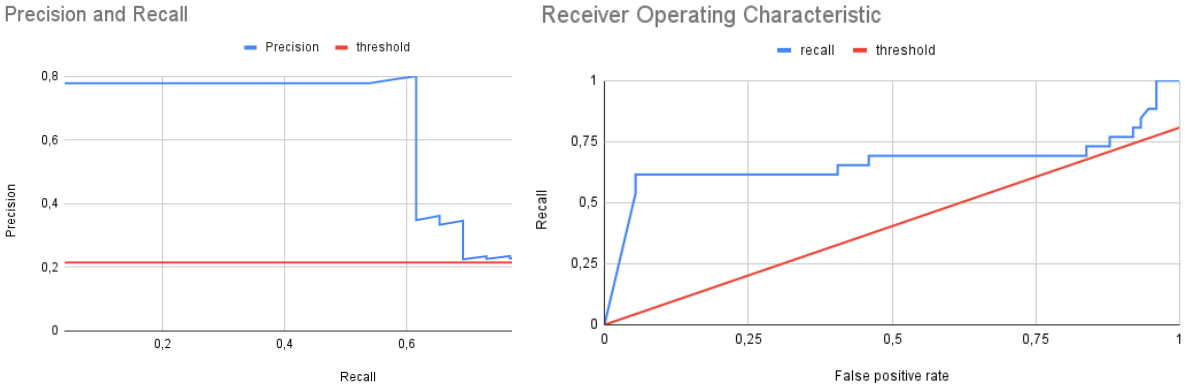
4.2 Configuration BoostSRL

There are various settings, that can be individually set by the user. A manual for the main usage can be found in the BoostSRLWiki [5]. The variable inputs we set are as follows:

- The number of trees remains constant to 10, since this is the best setting for acceptable learning times as well as enough learned models. With more trees, more ILP iterations are done and learning time increases, but the model improves only marginally. On the other hand, with less trees the quality of learning decreases, but results in a reduction in run time.
- Our provided mode file includes our used predicates *smokes(Person)* and *friends(Person,Person)* and the types of their variables. When introducing recursive clauses it is now possible to learn identity clauses like$s_smokes(a) \Rightarrow smokes(a)$. To avoid a generation of such clauses, we disabled that the same variable can appear in the head an in the body
- We did not provide any negative examples, just a file containing facts. Negative examples are learned during the ILP loop by using the default *negative/positive ratio*. With this setting program learned twice as many negative examples as positive examples.

4.3 Metrics

We want to measure the quality of the probability prediction of our learned model for the created data sets. To explain the used measure tools, we introduce some terminology. The *threshold* is a value, where all probabilities between 0 and 1 are split in two groups. Every prediction smaller than the threshold is considered false, the remaining ones are considered to be true. In general predictions can be divided in four different categories:



(a) Example for a PR curve

(b) Example for a ROC curve

Figure 4.1: Example for typical PR and ROC curves

- *true positive* (TP) or *recall*: correctly indicates the presence of a condition
- *true negative* (TN): correctly indicates the absence of a condition
- *false positive* (FP) or *type error I*: incorrectly indicates the presence of a condition
- *false negative* (FN) or *type error II*: incorrectly indicates the absence of a condition

The subset of true predictions contains TP and FP probabilities, whereas the TN and FP predictions form the set of false predictions. A smaller threshold leads to an increasing number of TP, but also FP. Further, the proportion of TP in all positive predictions is called *precision* and is defined as $\frac{TP}{(TP+FP)}$ and measures the relevancy of predicted results. In contrary the *true positive rate* (TPR) or *recall* measures if our model correctly identifies TPs and is defined as TPR=$\frac{TP}{(TP+FN)}$.

The *false positive rate* (FPR) indicates the proportion of FP of all negative examples and is defined as FPR = $\frac{FP}{(FP+TN)}$. AUC-ROC and AUC-PR based on thresholds. The AUC-ROC plots the recall against the FPR on a fixed threshold value of 0.5 and integrates it. Higher values are desirable, since it indicates how good the model predicts actual true values as true and actually false values as false. The AUC-ROC is maximized by having the FPR near 0 and TPR near 1.

The PR is a trade off between precision and recall for different thresholds. This threshold depends on the predicted positive examples and therefore changes for every model. To maximize the PR values we look for a low false negative rate and a low false positive rate. The higher the values of the AUC-PR curve, the more accurate are the predicted results.

CLL values are logarithmic probabilities supposing that our trained model is correct. It is a model likelihood estimation. The better the model predicts, the closer CLL values are to 0. Since we take the logarithm of probabilities (between 0 and 1), CLL values are negative. CLL indicates the quality of our models predictions. The closer positvie predicted probabilities are to 1, the smaller is our CLL value. The other way round it is for negative probabilities predicted close to 0. CLL is calculated as follows: For our model we want the joint probability distribution, where every example is conditioned to all other examples. We approximate the likelihood, since we should not use a probability prediction as

condition for another examples prediction.

$$CLL = \frac{\sum_{ex_{pos}} \log(P(ex_{pos})) + \sum_{ex_{neg}} \log(1 - P(ex_{pos}))}{|ex|}$$

where ex_{pos} consists of all positive examples (TP + FP) and ex_{neg} of all negative exmples (TN + FN). The denominator ex is an average log likelihood of all examples.

CHAPTER 5

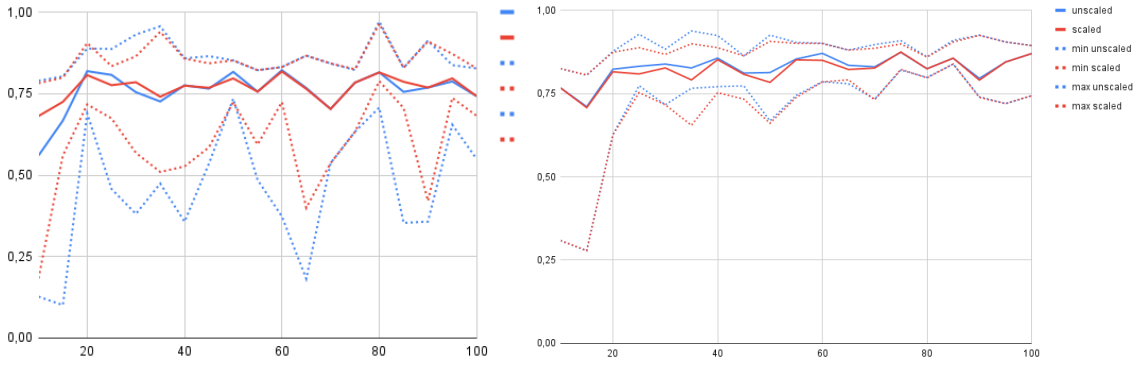
Observations and results

For all our setups we plotted the IDR of AUC-ROC, AUC-PR and CLL curves, as well as the run time, each in its own diagram. In every plot we compare a run of the boosted MLN with a run of the boosted DA-MLN. They both ran on the same data set and on same learned model. The maximum and minimum values of the IDR of each setup are displayed as well.

The first setup we want to investigate has a fixed size/count of 5 and 0% probability of having a friend outside a group. For AUC-ROC curves we encounter a similar trend for scaled and unscaled outputs displayed in Figure 5.1. All Values are around 0,75, which means the predictions are quite good. For the AUC-ROC curves in 5.1(b) we encounter a slightly better prediction of the unscaled model. The same trend we have for the PR curves in figure 5.2. The values for scaled and unscaled models are rather similar. Again in 5.2(b) there is a slightly better prediction for unscaled models. The run time for testing the trained models is in both cases 5.3(a) and 5.3(b) better for unscaled models. For the CLL curves, we encounter the most significant difference between the different domain structures. For domains with a fixed number of groups in 5.4(b) the CLL values of unscaled models are getting worse, the higher the scaling value is. Only at a domain size of 60 the CLL of unscaled models starts to adjust to the scaled ones. For domains with a fixed group size, displayed in 5.4(b), both scaled and unscaled models perform poorly on CLL. But for unscaled models the CLL values perform better up to a domain size of 90.

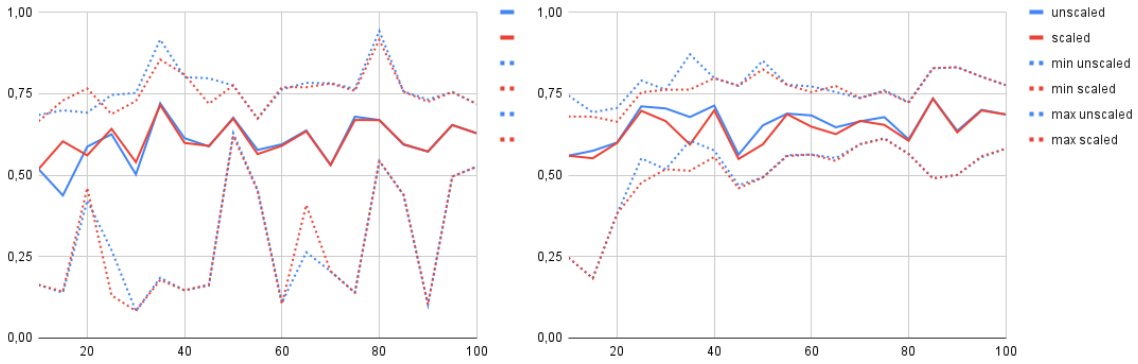
When changing the setup to a fixed size of 10, while still isolating groups from each other, we observe a similar behavior for CLL values in figure 5.8. In general the values are not as good as for case of a fixed size of 5 in 5.4 where the scaled values are around -0.5, but unscaled values performing worse up to a domain size of 80. For AUC-ROC and AUC-PR the plots have similar trends like in 5.1 and 5.2. The median AUC-ROC values in 8.1 (appendix) are 0,75 or better for the first case, whereas for the second case they are similar except for the outlieres at very small domains. When looking at AUC-PR curves the trend is similar. PR values for both cases are similar, for case two a small outlier at small domains can be seen in 8.2.

When we allow a 10% chance of having a friend outside a group the AUC-ROC and AUC-PR curves don't change significantly. In the appendix the corresponding plots can be found in figure 8.3 and 8.4. Interestingly, we observe a new behavior in CLL plots in figure 5.5. For the case of a fixed group count both graphs behave similar to the previous setup in figure 5.4, but for fixed group sizes, the unscaled model now behaves worse than the



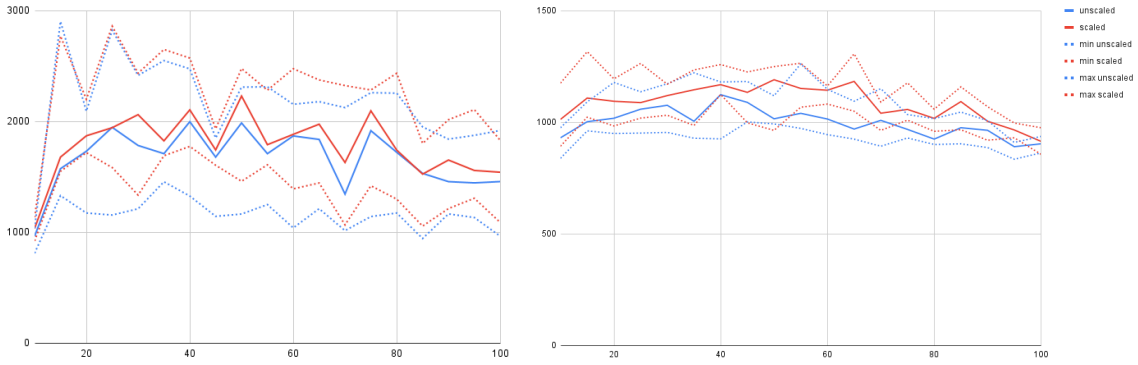
(a) fixed group count of 5 (b) fixed group size of 5

Figure 5.1: AUC-ROC curves with 0% probability having a friend outside a group



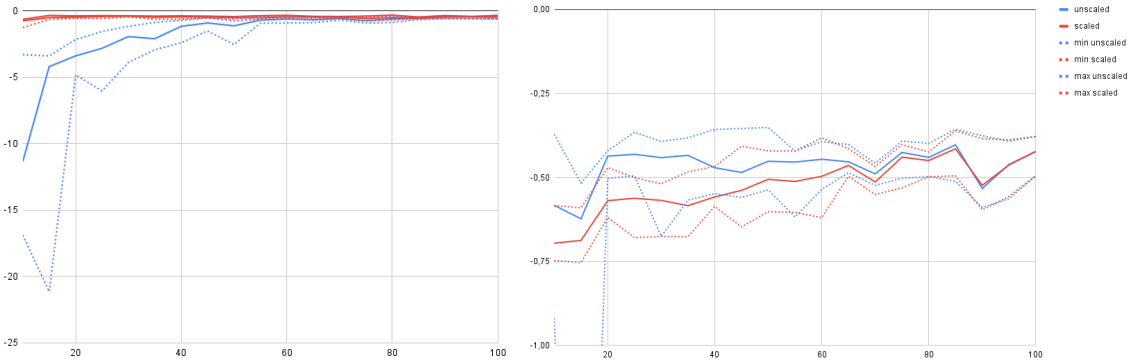
(a) fixed group count of 5 (b) fixed group size of 5

Figure 5.2: PR curves with 0% probability having a friend outside a group



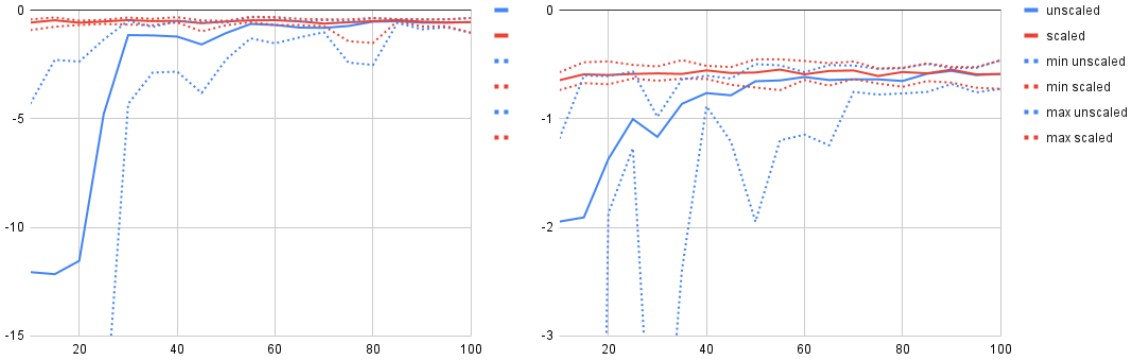
(a) fixed group count of 5 (b) fixed group size of 5

Figure 5.3: Runtime curves with 0% probability having a friend outside a group



(a) fixed group count of 5 (b) fixed group size of 5

Figure 5.4: CLL curves with 0% probability having a friend outside a group

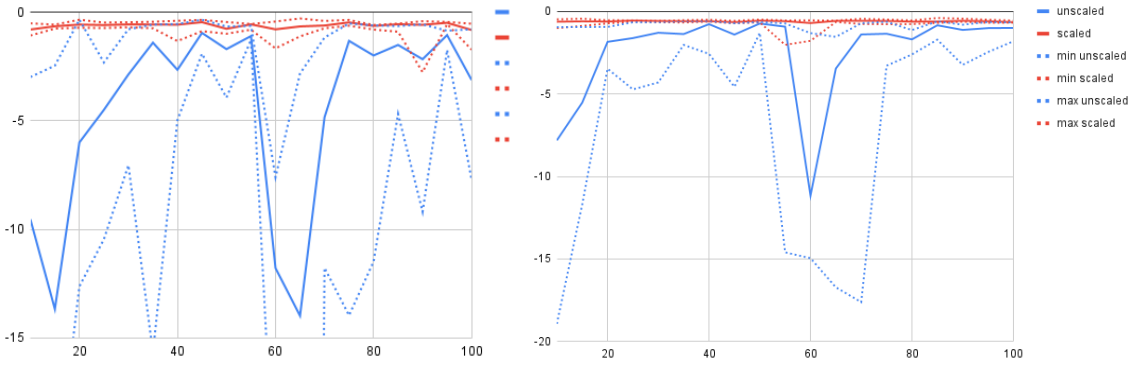


(a) fixed group count of 5 (b) fixed group size of 5

Figure 5.5: CLL curves with 10% probability having a friend outside a group

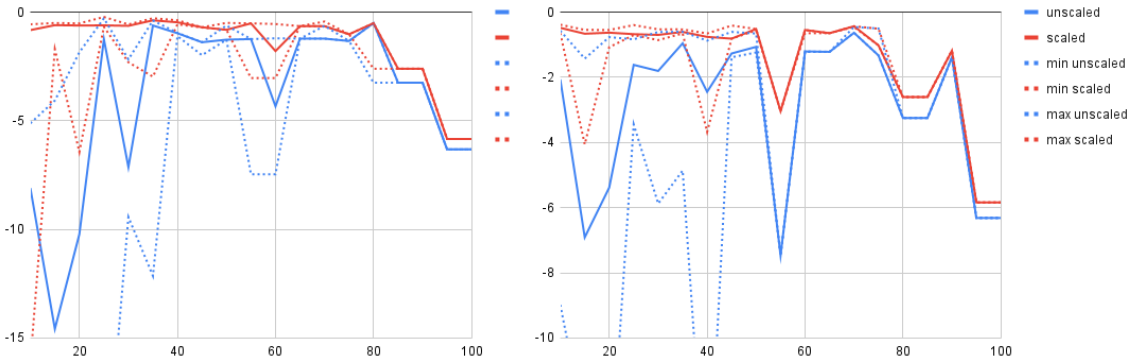
scaled model. We again have the case, that for a fixed group size the CLL values in general are worse than in the model for a fixed group count. But still, in general the CLL values for unscaled models for a fixed size are better than CLL values for a scaled model.

By increasing the probability to 20% we can see in figure 5.6(a) CLL values for unscaled models get worse than in 5.5, while for scaled models the values stay constantly near 0. In figure 5.6(b) ths range of CLL increases up to -10. But when comparing (b) to (a) the unscaled values are better models with a fixed group size.



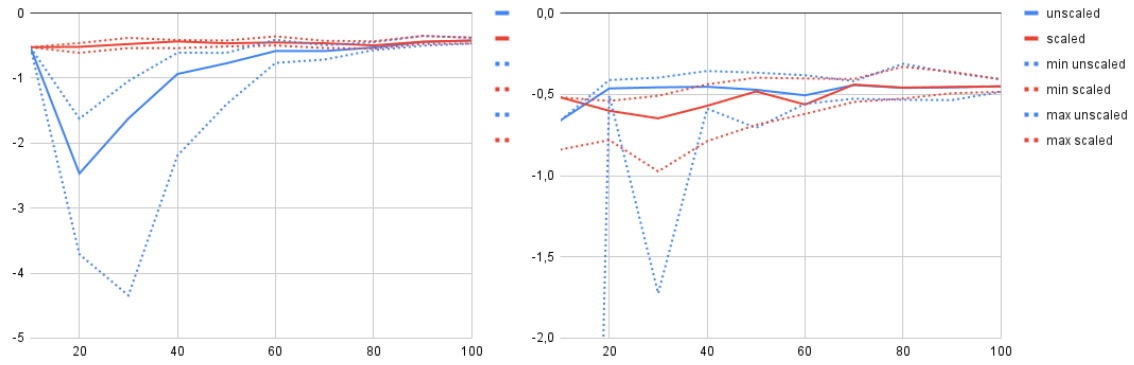
(a) fixed group count of 5 (b) fixed group size of 5

Figure 5.6: CLL curves with 20% probability having a friend outside a group



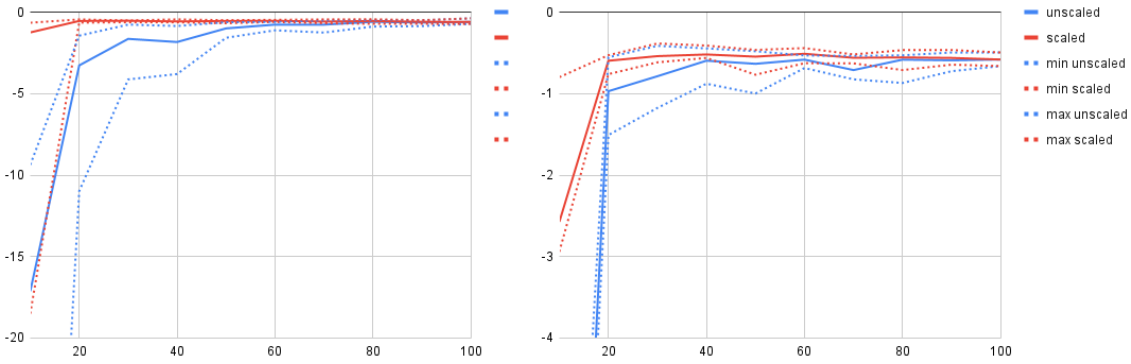
(a) fixed group count of 5 (b) fixed group size of 5

Figure 5.7: CLL curves with 30% probability having a friend outside a group



(a) fixed group count of 10 (b) fixed group size of 10

Figure 5.8: CLL curves with 0% probability having a friend outside a group



(a) fixed group count of 10

(b) fixed group size of 10

Figure 5.9: CLL curves with 10% probability having a friend outside a group

CHAPTER 6

Discussion

The most meaningful metric for our observation is the CLL value. On the one hand, we see the biggest difference here, on the other hand they are the most expressive. Since the PR and ROC values are very similar for DA-MLN and boosted MLN they are not very informative. They just show that our models' predictions are quite good which is a good sign.

When looking at the graphs in chapter 5 we notice that our assumptions (1) and (2) from chapter 3 hold for isolated setups. Scaling works well on domains with varying group sizes, because the number of connections grows as we would expect. In contrary scaling the values of domains with fixed group sizes is counterproductive. The expectation was that the scaled values are worse and our results confirm this. The scaling vector reduces the expected number of friends for increasing domains as more connections are expected, which are in fact not-existent. With the scaling vector a wrong number of friends is predicted. This explains why the unscaled model performs better in the case of CLL values.

As soon as we allow connections outside a group, hypothesis (2) does not apply anymore. As we see in 5.5(b) the scaled model performs better than the unscaled one and CLL values are rather constant around the value 0.6. When comparing the CLL curves from 5.5(a) and 5.5(b) we encounter a similar trend, but CLL values in 5.5(a) get rapidly better as soon as the domain grows. However the CLL values for small domains are about 5x worse than for fixed group size. This is due to the fact that small domains with a fixed group count only contain small groups (starting from 1-2 members per group). Therefore only few connections are inside the group and scaling has only an effect on connections outside a group. Since it is unlikely having connections outside a group the total count of connections is rather small. When looking at fixed group size, groups are rather big for small domains. Such data contains more connections and therefore CLL values are better for small domains (compared to 5.5(a)). As soon as domains are larger than 30 scaling results for fixed group count and fixed group size get similar.

When we increase the probability for connections to 20% and again consider the CLL curves, the behavior is similar to models with 10% probability. The scaling here works very well for 5.8(a) and 5.8(b) where the all values are around 0.5. This is, because the number of connections again increases for outside friendships. In contrary the unscaled values for both graphs 5.8(a) and 5.8(b) are even worse. Only the constant connections for fixed groups inside a group buffer the downward trend of the curve in 5.8(b).

For probabilities of 30% we have a worsening of unscaled CLL curves, again because of

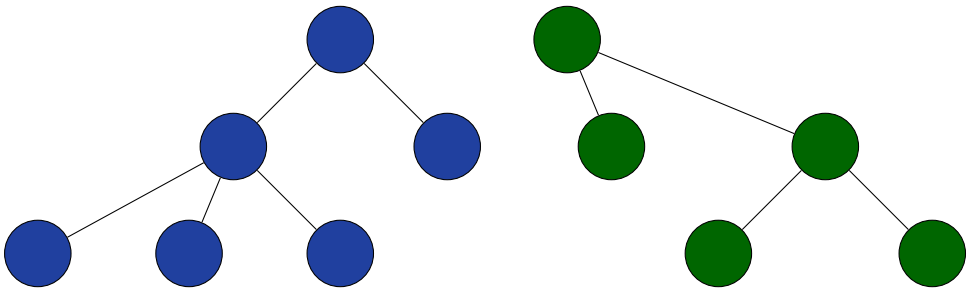


Figure 6.1: Graphical representation of snowball sampling where edges represent some kind of relation between nodes (persons)

the growing relation. This is also one reason why we considered domains with maximal 30%. We don't get any new information, just an amplification of the effect. Additionally, training models became more difficult. With increasing domains, the number of potential friends also increases. At that point the program failed to train a model. Since the number of connections grew so large it was no longer possible to draw any conclusions about the groups. An example visualisation of such a data set can be seen in figure 8.5 in the appendix. Therefore, the influence of each individual friend became smaller and smaller and it was hardly possible to predict the smoking habits. The outcome was hardly better than guessing.

Our observations show, that DA-MLN does not work well for transferring data between domains containing isolated groups whose size is not correlated with the size of the domain. Such data can be collected by snowball sampling. This method is often used, when it is hard to find a simple random sample. Having an initially small set of samples, other participants are reached through the social networks of the first samples. These other samples may have similar traits, because of their social network. As an example reconsider the case from chapter 1 where data is collected in a school class. It is more practical to collect date from a whole class rather than picking single students from different classes. This collected data has some bias which leads to an isolation of these social network groups in the database as exemplary displayed in figure 6.1.

As an example, let's consider an epidemic and observe how the virus spread through the schools. The best way to collect data would be picking random pupils from random classes and schools. But in reality, there are often only corporations with certain schools, where data can be collected. Additionally, data is normally collected from whole classes instead of a few randomly chosen individuals. This class can be considered as an isolated group. Therefore, one could not scale the contagion rate from this class up for the whole school using DA-MLN, but should use MLN.

CHAPTER 7

Conclusion

We observed, that scaling boosted DA-MLN works pretty well in a lot of domain setups. But there are indeed some constellations of data sets for which MLN should be preferred. This holds for data sets, which consist of isolated groups with relations within in the group, where relations are independent of the domain size. These relations of the targets don't grow in parallel to the increase of the domain size, or even stay constant. We showed this with concrete examples of our synthetic data set "friends and smokers", which we modified by our self, to model different domain structures. In every scenario, AUC-ROC and AUC-PR values were very similar for scaled and unscaled domains. Only in extreme cases we could achieve a worsening of CLL curves in scaling. Meaning, only when groups were isolated and group sizes stayed constant, the re-parameterization of MLN weights made predictions worse. By knowing this, we could say boosted DA-MLN does not work well for data sets, that naturally vary in their domain structure but relations are independet of the domain size, e.g. data sets created by snowball sampling.

CHAPTER 8

Appendix

dataset (a)

```
useStdLogicVariables: true.
friendsPerson_0,Person_1.
friendsPerson_1,Person_0.
friendsPerson_0,Person_2.
friendsPerson_2,Person_0.
friendsPerson_0,Person_4.
friendsPerson_4,Person_0.
friendsPerson_0,Person_13.
friendsPerson_13,Person_0.
friendsPerson_1,Person_6.
friendsPerson_6,Person_1.
friendsPerson_3,Person_6.
friendsPerson_6,Person_3.
friendsPerson_3,Person_14.
friendsPerson_14,Person_3.
friendsPerson_4,Person_5.
friendsPerson_5,Person_4.
friendsPerson_4,Person_7.
friendsPerson_7,Person_4.
friendsPerson_4,Person_10.
friendsPerson_10,Person_4.
friendsPerson_5,Person_8.
friendsPerson_8,Person_5.
friendsPerson_6,Person_9.
friendsPerson_9,Person_6.
friendsPerson_9,Person_10.
friendsPerson_10,Person_9.
friendsPerson_9,Person_11.
friendsPerson_11,Person_9.
friendsPerson_12,Person_13.
friendsPerson_13,Person_12.
friendsPerson_12,Person_14.
friendsPerson_14,Person_12.
friendsPerson_13,Person_14.
friendsPerson_14,Person_13.
smokesPerson_2.
smokesPerson_9.
smokesPerson_11.
smokesPerson_13.
```

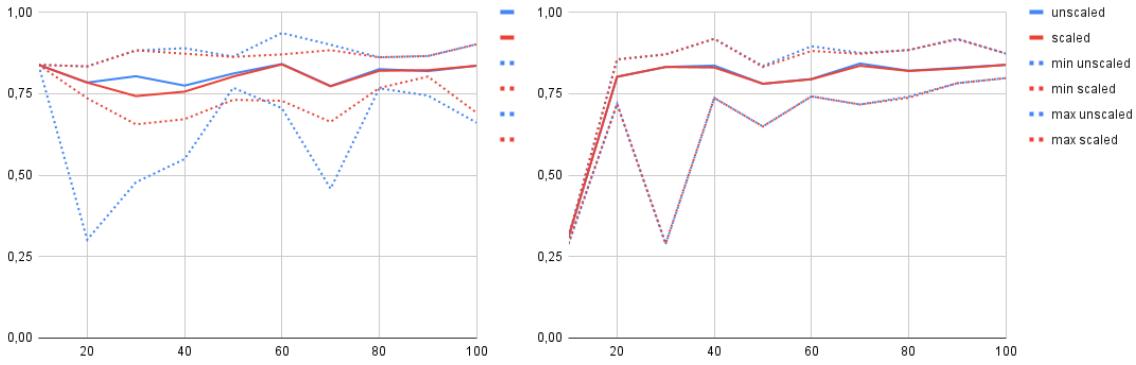
```
smokesPerson_14.
s_smokesPerson_2.
s_smokesPerson_9.
s_smokesPerson_11.
s_smokesPerson_13.
s_smokesPerson_14.
not_s_smokesPerson_0.
not_s_smokesPerson_1.
not_s_smokesPerson_3.
not_s_smokesPerson_4.
not_s_smokesPerson_5.
not_s_smokesPerson_6.
not_s_smokesPerson_7.
not_s_smokesPerson_8.
not_s_smokesPerson_10.
not_s_smokesPerson_12.
```

---

dataset (b)

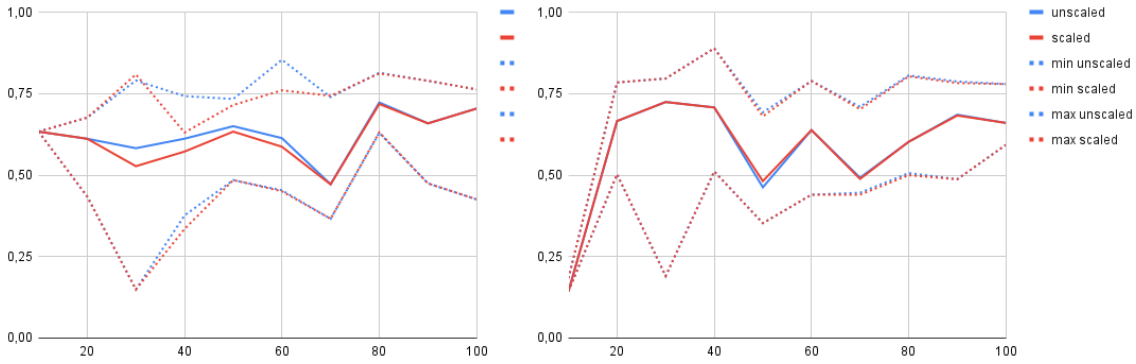
```
useStdLogicVariables: true.
friendsPerson_0,Person_1.
friendsPerson_1,Person_0.
friendsPerson_0,Person_2.
friendsPerson_2,Person_0.
friendsPerson_0,Person_3.
friendsPerson_3,Person_0.
friendsPerson_0,Person_4.
friendsPerson_4,Person_0.
friendsPerson_0,Person_8.
friendsPerson_8,Person_0.
friendsPerson_1,Person_2.
friendsPerson_2,Person_1.
friendsPerson_1,Person_4.
friendsPerson_4,Person_1.
friendsPerson_2,Person_3.
friendsPerson_3,Person_2.
friendsPerson_2,Person_4.
friendsPerson_4,Person_2.
friendsPerson_2,Person_11.
friendsPerson_11,Person_2.
friendsPerson_2,Person_12.
friendsPerson_12,Person_2.
friendsPerson_3,Person_13.
friendsPerson_13,Person_3.
friendsPerson_4,Person_6.
friendsPerson_6,Person_4.
friendsPerson_4,Person_9.
friendsPerson_9,Person_4.
friendsPerson_5,Person_8.
friendsPerson_8,Person_5.
friendsPerson_6,Person_7.
friendsPerson_7,Person_6.
friendsPerson_6,Person_8.
friendsPerson_8,Person_6.
friendsPerson_6,Person_9.
friendsPerson_9,Person_6.
friendsPerson_7,Person_8.
friendsPerson_8,Person_7.
friendsPerson_7,Person_9.
friendsPerson_9,Person_7.
friendsPerson_8,Person_9.
friendsPerson_9,Person_8.
friendsPerson_10,Person_11.
```

```
friendsPerson_11,Person_10.
friendsPerson_10,Person_12.
friendsPerson_12,Person_10.
friendsPerson_10,Person_14.
friendsPerson_14,Person_10.
friendsPerson_11,Person_12.
friendsPerson_12,Person_11.
friendsPerson_11,Person_13.
friendsPerson_13,Person_11.
friendsPerson_12,Person_13.
friendsPerson_13,Person_12.
friendsPerson_12,Person_14.
friendsPerson_14,Person_12.
friendsPerson_13,Person_14.
friendsPerson_14,Person_13.
smokesPerson_0.
smokesPerson_3.
smokesPerson_4.
smokesPerson_7.
s_smokesPerson_0.
s_smokesPerson_3.
s_smokesPerson_4.
s_smokesPerson_7.
not_s_smokesPerson_1.
not_s_smokesPerson_2.
not_s_smokesPerson_5.
not_s_smokesPerson_6.
not_s_smokesPerson_8.
not_s_smokesPerson_9.
not_s_smokesPerson_10.
not_s_smokesPerson_11.
not_s_smokesPerson_12.
not_s_smokesPerson_13.
not_s_smokesPerson_14.
```



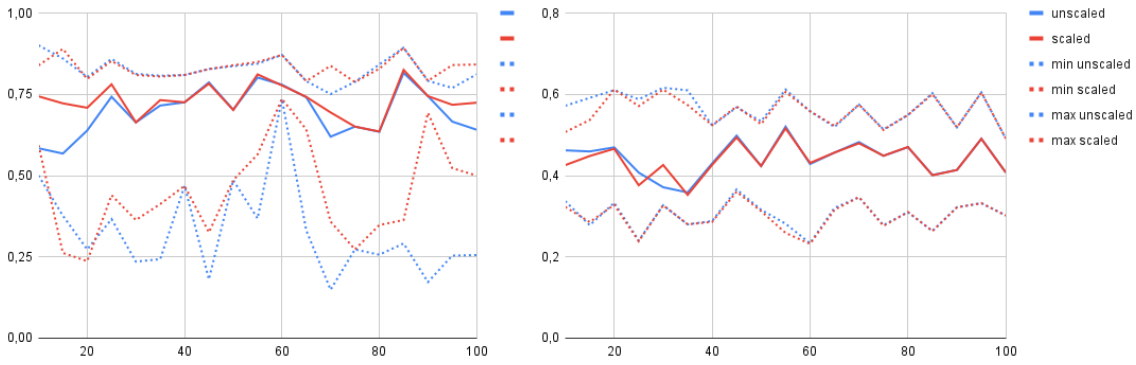
(a) fixed group count of 10 (b) fixed group size of 10

Figure 8.1: AUC-ROC curves with 0% probability having a friend outside a group



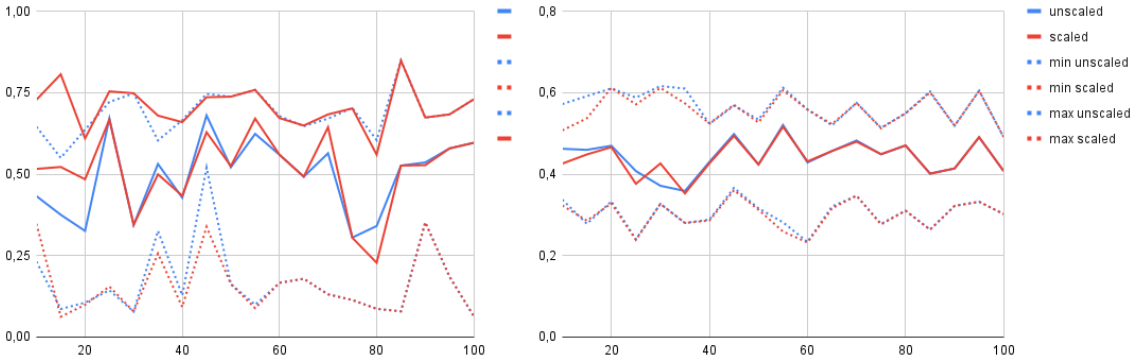
(a) fixed group count of 10 (b) fixed group size of 10

Figure 8.2: AUC-PR curves with 0% probability having a friend outside a group



(a) fixed group count of 5 (b) fixed group size of 5

Figure 8.3: AUC-ROC curves with 10% probability having a friend outside a group



(a) fixed group count of 5

(b) fixed group size of 5

Figure 8.4: AUC-PR curves with 10% probability having a friend outside a group

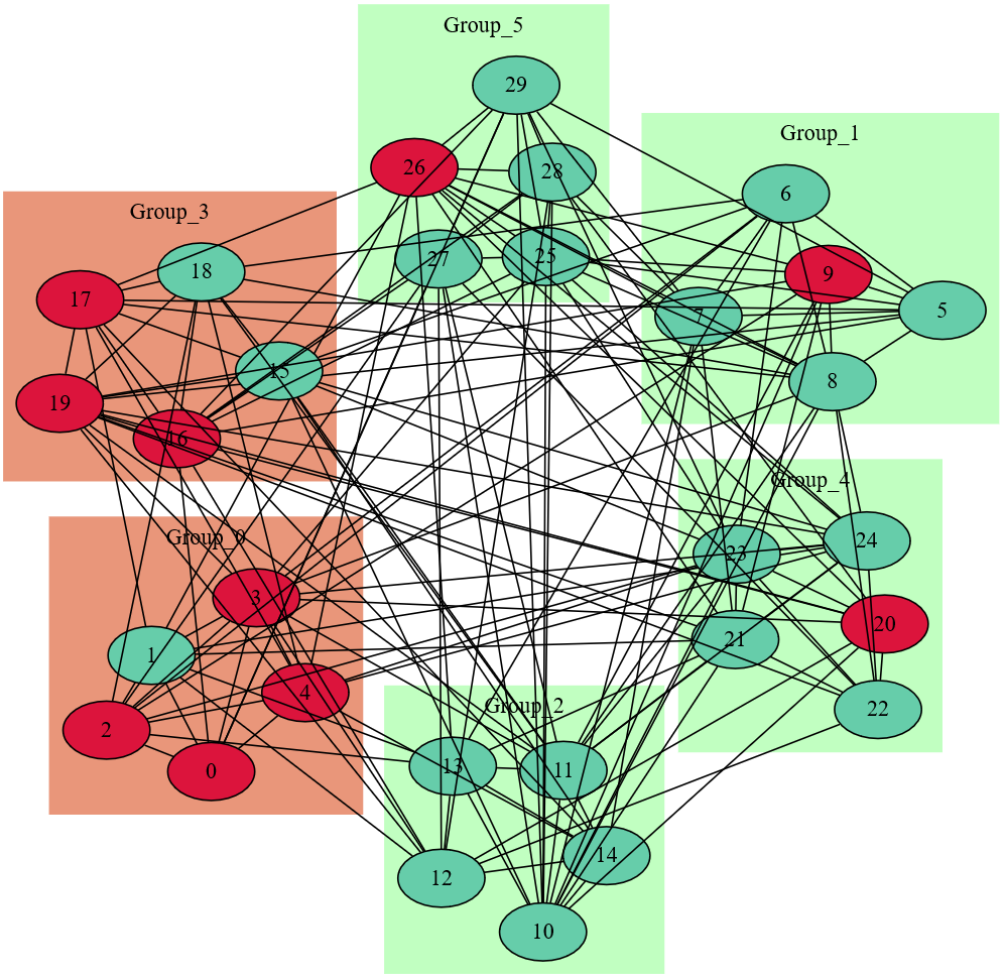


Figure 8.5: Example for a Data set of size 30 with a fixed group size of 5 and 30% probability of having a friend outside a group.

List of abbreviations

StarAI Relational statistic artificial intelligence

FOL first-order logic

MLN Markov logic network

DA-MLN Domain size aware Markov logic networks

ILP inductive logic programming

RRT relational regression tree

RRC Relational Regression Clauses

AUC-ROC Area Under Curve of Receiver Operating Characteristic

AUC-PR Area Under Curve Precision and Recall

CLL Conditional Log Likelihood

IDR Interdecile range

TP True positive

TN True negative

FP False positive

FN False negative

TPR True positive rate

FPR False positive rate

Bibliography

[1] Melvin Fitting and David Gries, *First-order logic and automated theorem proving*, Springer-Verlag, Berlin, Heidelberg, 1990.

[2] Tushar Khot, Sriraam Natarajan, Kristian Kersting, and Jude W. Shavlik, *Learning Markov logic networks via functional gradient boosting*, 11th IEEE International Conference on Data Mining, ICDM 2011, Vancouver, BC, Canada, December 11-14, 2011 (Diane J. Cook, Jian Pei, Wei Wang, Osmar R. Zaïane, and Xindong Wu, eds.), IEEE Computer Society, 2011, pp. 320–329.

[3] Happy Mittal, Ayush Bhardwaj, Vibhav Gogate, and Parag Singla, *Domain-size aware Markov logic networks*, The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS 2019, 16-18 April 2019, Naha, Okinawa, Japan (Kamalika Chaudhuri and Masashi Sugiyama, eds.), Proceedings of Machine Learning Research, vol. 89, PMLR, 2019, pp. 3216–3224.

[4] Sriraam Natarajan and Tushar Khot, *BoostSRL github*, https://github.com/booststarai/BoostSRL-Misc, 2017, Online accessed 2022-01-30.

[5] Sriraam Natarajan, Tushar Khot, Kristian Kersting, and Jude Shavlik, *BoostSRL wiki*, https://starling.utdallas.edu/software/boostsrl/wiki/, 2015, Online accessed 2022-01-30.

[6] Sriraam Natarajan, Tushar Khot, Dmitriy Ravdin, and Ramona Fabry, *Boosted DA-MLN github*, https://github.com/fabrr/boosted-DA-MLN, 2020, Online accessed 2022-03-20.

[7] Nandini Ramanan, Gautam Kunapuli, Tushar Khot, Bahare Fatemi, Seyed Mehran Kazemi, David Poole, Kristian Kersting, and Sriraam Natarajan, *Structure learning for relational logistic regression: An ensemble approach*, Principles of Knowledge Representation and Reasoning: Proceedings of the Sixteenth International Conference, KR 2018, Tempe, Arizona, 30 October - 2 November 2018 (Michael Thielscher, Francesca Toni, and Frank Wolter, eds.), AAAI Press, 2018, pp. 661–662.

[8] Dmitriy Ravdin, *Parameter scaling for boosted singe-target learning of Markov logic networks*, Bachelor thesis, LMU Munich, 2022.

[9] Matthew Richardson and Pedro M. Domingos, *Markov logic networks*, Mach. Learn. **62** (2006), no. 1-2, 107–136.

[10] Felix Weitkämper, *Scaling the weight parameters in Markov logic networks and relational logistic regression models*, CoRR **abs/2103.15140** (2021).