Adversarial Sets for Regularising Neural Link Predictors

Pasquale Minervini[1] **Thomas Demeester**[2] **Tim Rocktäschel**[3] **Sebastian Riedel**[1]
University College London, London, United Kingdom[1]
Ghent University - iMinds, Ghent, Belgium[2]
University of Oxford, Oxford, United Kingdom[3]

Abstract

In adversarial training, a set of models learn together by pursuing competing goals, usually defined on *single* data instances. However, in relational learning and other non-*i.i.d* domains, goals can also be defined over *sets* of instances. For example, a link predictor for the IS-A relation needs to be consistent with the *transitivity* property: if IS-A(x_1, x_2) and IS-A(x_2, x_3) hold, IS-A(x_1, x_3) needs to hold as well. Here we use such assumptions for deriving an *inconsistency loss*, measuring the degree to which the model violates the assumptions on an adversarially-generated set of examples. The training objective is defined as a minimax problem, where an *adversary* finds the most offending adversarial examples by maximising the inconsistency loss, and the model is trained by jointly minimising a supervised loss and the inconsistency loss on the adversarial examples. This yields the first method that can use function-free Horn clauses (as in Datalog) to regularise any neural link predictor, with complexity independent of the domain size. We show that for several link prediction models, the optimisation problem faced by the adversary has efficient closed-form solutions. Experiments on link prediction benchmarks indicate that given suitable prior knowledge, our method can significantly improve neural link predictors on all relevant metrics.

1 INTRODUCTION

Adversarial training [Dalvi et al., 2004, Goodfellow et al., 2014, Szegedy et al., 2014] is a learning setting where two or more models learn together by pursuing competing goals. Adversarial learning has received increasing attention in recent years, mainly motivated by the observation that neural networks can be vulnerable to carefully-crafted adversarial examples [Goodfellow et al., 2014]. Adding additional loss terms to training objectives for enforcing models to be robust to adversarial examples can be very beneficial, for instance in the context of image synthesis [Dosovitskiy et al., 2015].

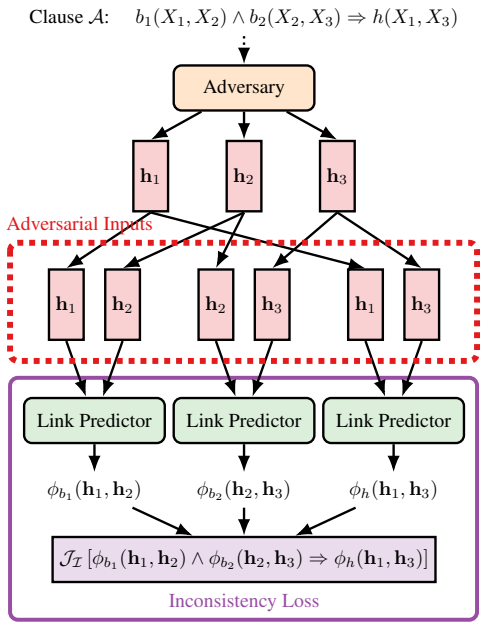


Figure 1: Overview of regularisation via adversarial sets for neural link prediction. Given a clause $\mathcal{A}$ with variables $\{X_1, X_2, X_3\}$, an adversary maps each variable to an adversarial entity embedding that maximises an inconsistency loss, while the discriminator is trained by minimising the link prediction and inconsistency loss.

Such competing goals are generally defined on *single* data instances. For example, in Generative Adversar-

ial Networks (GANs) [Goodfellow et al., 2014], a generator is trained to synthesise single *fake* data instances (*e.g.* images) that are classified as *real* by a discriminator, while the discriminator is trained to discriminate between single *real* and *fake* data instances. However, for relational tasks such as *link prediction* or *knowledge base population* [Nickel et al., 2012, Riedel et al., 2013b, Socher et al., 2013, Chang et al., 2014, Toutanova et al., 2015, Neelakantan et al., 2015], where objects can interact with each other, such goals can also be defined in terms of several instances.

For instance, consider the hypernymy relation IS-A. Since state-of-the-art link predictors rank or score pairs in isolation [Bordes et al., 2013, Toutanova et al., 2015, Trouillon et al., 2016], they cannot explicitly account for the transitivity property of IS-A. This means that a predictor might infer IS-A(x_1, x_2) and IS-A(x_2, x_3), but not IS-A(x_1, x_3). In this case, encouraging transitivity to hold for such models can be achieved by defining a goal on three related inputs rather than a single input: $\{(x_1, x_2), (x_2, x_3), (x_1, x_3)\}$. The goal of the adversary would be to find inputs that lead to inconsistent predictions, while the predictor's goal would be restoring consistency on such sets of inputs.

In this paper, we introduce Adversarial Set Regularisation (ASR), a general and scalable method for regularising neural link prediction models by using background knowledge. Our method is summarised by the *adversarial training* architecture in Fig. 1. In ASR, we first define a set of constraints on multiple problem instances in the form of function-free First-order Logic (FOL) clauses. From these clauses we then derive an *inconsistency loss* that measures the extent to which constraints are violated.

The learning architecture is composed of two models, an *adversary* and a *discriminator*, having two competing goals. The *adversary* tries to find a set of adversarial input representations for which, according to the discriminator (a link prediction model), the constraints do not hold. Such a set is found by maximising the inconsistency loss. The *discriminator*, on the other hand, uses the inconsistency loss on the adversarial input representations for regularising its training process.

Our proposed training algorithm, described in Sect. 3, can be seen as a zero-sum game involving two players, where: i) one player, the link predictor, has to predict a target graph over a set of *real* inputs, while ensuring global consistency over a set of *generated adversarial* inputs; and ii) the other player, the adversary, has to generate adversarial input representations such that the link predictor fails at constructing consistent graphs.

The link prediction model is trained by jointly minimising both the data loss and the inconsistency loss on the adversarial input sets by updating the model, while the adversary maximises the inconsistency loss by changing the input sets. Another interpretation is that the adversary acts as an *adaptive regulariser* for training neural link predictors.

Our method is related to Rocktäschel et al. [2015] and its variant KALE [Guo et al., 2016], which also minimise an inconsistency loss over sets of instances. A core difference with the proposed approach is that, instead of generating adversarial examples in representation space, they select random entities from the training and test sets. This generally leads to less effective updates, since the random entities can already satisfy the imposed constraints. Furthermore, they also enforce consistency not just by changing the relation-specific parameters, but also by changing the distributed representations of the entity inputs. This can lead to poor generalisation on entities less frequently sampled during training. Demeester et al. [2016] overcome this problem, but their approach only works for a single type of model and simple clauses. Our approach has no such restrictions, and we formally show that it is a generalisation of their work.

We show empirically that, by training on adversarial input sets, the link prediction model becomes indeed more robust. In experiments, we use our approach to inject prior assumptions into several state-of-the-art models, indicating the general nature of our approach. The regularised models outperform the original models on real-world data sets, namely WN18 and FB122. Moreover, using the same constraints and experimental setup, we outperform KALE both in terms of absolute performance, and in terms of relative improvements when compared to their underlying base models.

Our contributions are threefold: i) we introduce a novel approach to regularise neural link prediction models based on prior relational assumptions, such as transitivity – this is the first work that uses adversarial input sets for doing so; ii) we present an optimisation algorithm for solving the underlying minimax problem; and iii) we derive closed form solutions for the inner maximisation problem that enable faster training, provide intuitive insights into the goal of the adversary, and show that the method of Demeester et al. [2016] can be derived from our framework.

In the next sections we first briefly introduce the link prediction problem and state-of-the-art link prediction models. Then we present our adversarial approach to injecting prior knowledge. We conclude with experimental results and a discussion.

2 LINK PREDICTION

In this work, we focus on the problem of *predicting missing links* in large, multi-relational networks such as FREEBASE. In the literature, this problem is referred to as *link prediction*. We specifically focus on *knowledge graphs*, *i.e.*, graph-structured knowledge bases where factual information is stored in the form of relationships between entities. Link prediction in knowledge graphs is also known as *knowledge base population*. We refer to Nickel et al. [2016] for a recent survey on approaches for this problem.

A knowledge graph $\mathcal{G} \triangleq \{(r, a_1, a_2)\} \subseteq \mathcal{R} \times \mathcal{E} \times \mathcal{E}$ can be formalised as a set of triples (facts) consisting of a relation type $r \in \mathcal{R}$ and two entities $a_1, a_2 \in \mathcal{E}$, respectively referred to as the *subject* and the *object* of the triple. Each triple (r, a_1, a_2) encodes a relationship of type r between a_1 and a_2, represented by the fact $r(a_1, a_2)$.

Link prediction in knowledge graphs is often simplified to a *learning to rank* problem, where the objective is to find a score or ranking function $\phi_r^{\boldsymbol{\theta}} : \mathcal{E} \times \mathcal{E} \mapsto \mathbb{R}$ for a relation r that can be used for ranking triples according to the likelihood that the corresponding facts hold true.

2.1 Neural Link Prediction

Recently, a specific class of link predictors received a growing interest [Nickel et al., 2016]. These predictors can be understood as multi-layer neural networks. Given a triple $\mathbf{x} = (r, a_1, a_2)$, the associated score $\phi_r^{\boldsymbol{\theta}}(a_1, a_2)$ is given by a neural network architecture encompassing an *encoding layer* and a *scoring layer*.

In the encoding layer, the subject and object entities a_1 and a_2 are mapped to low-dimensional vector representations (embeddings) $\mathbf{h}_1 \triangleq \mathbf{h}(a_1) \in \mathbb{R}^k$ and $\mathbf{h}_2 \triangleq \mathbf{h}(a_2) \in \mathbb{R}^k$, produced by an encoder $\mathbf{h}^{\boldsymbol{\gamma}} : \mathcal{E} \rightarrow \mathbb{R}^k$ with parameters $\boldsymbol{\gamma}$. This layer can be pre-trained [Vylomova et al., 2016] or, more commonly, learnt from data by back-propagating the link prediction error to the encoding layer [Bordes et al., 2013, Nickel et al., 2016, Trouillon et al., 2016].

In the scoring layer, the entity representations $\mathbf{h}_1$ and $\mathbf{h}_2$ are scored by a function $\phi_r^{\boldsymbol{\theta}}(\mathbf{h}_1, \mathbf{h}_2)$, parametrised by $\boldsymbol{\theta}$.

Summarising, the high-level architecture is defined as:

$$\begin{aligned} \mathbf{h}_1, \mathbf{h}_2 &\triangleq \mathbf{h}^{\boldsymbol{\gamma}}(a_1), \mathbf{h}^{\boldsymbol{\gamma}}(a_2) \\ \phi_r(a_1, a_2) &\triangleq \phi_r^{\boldsymbol{\theta}}(\mathbf{h}_1, \mathbf{h}_2) \end{aligned}$$

Ideally, more likely triples should be associated with higher scores, while less likely triples should be associated with lower scores.

While the literature has produced a multitude of encoding and scoring strategies, for brevity we overview only a small subset of these. However, we point out that our method makes no further assumptions about the network architecture other than the existence of an argument encoding layer.

2.2 Encoding Layer

Given an entity $a \in \mathcal{E}$, the entity encoder $\mathbf{h}^{\boldsymbol{\gamma}}$ is usually implemented as a simple embedding layer $\mathbf{h}^{\boldsymbol{\gamma}}(a) \triangleq [\boldsymbol{\gamma}]_a$, where $\boldsymbol{\gamma}$ is an embedding matrix [Nickel et al., 2016]. For pre-trained embeddings, the embedding matrix is fixed. Note that other encoding mechanisms are conceivable, such as recurrent or convolutional neural networks.

2.3 Scoring Layer

DistMult DISTMULT [Yang et al., 2015] represents each relation r using a parameter vector $\boldsymbol{\theta}_r \in \mathbb{R}^k$, and scores a link of type r between $(\mathbf{h}_1, \mathbf{h}_2)$ using the following scoring function:

$$\phi_r^{\boldsymbol{\theta}}(\mathbf{h}_1, \mathbf{h}_2) \triangleq \langle \boldsymbol{\theta}_r, \mathbf{h}_1, \mathbf{h}_2 \rangle \triangleq \sum_{i=1}^{k} \boldsymbol{\theta}_{r,i} \mathbf{h}_{1,i} \mathbf{h}_{2,i},$$

where $\langle \cdot, \cdot, \cdot \rangle$ denotes the tri-linear dot product.

ComplEx COMPLEX [Trouillon et al., 2016] is an extension of DISTMULT using complex-valued embeddings while retaining the mathematical definition of the dot product. In this model, the scoring function is defined as follows:

$$\phi_r^{\boldsymbol{\theta}}(\mathbf{h}_1, \mathbf{h}_2) \triangleq \mathrm{Re}\left(\langle \boldsymbol{\theta}_r, \mathbf{h}_1, \overline{\mathbf{h}_2} \rangle\right),$$

where $\boldsymbol{\theta}_r, \mathbf{h}_1, \mathbf{h}_2 \in \mathbb{C}^k$ are complex vectors, $\overline{\mathbf{x}}$ denotes the complex conjugate of $\mathbf{x}$, and $\mathrm{Re}\,(\mathbf{x}) \in \mathbb{R}^k$ denotes the real part of $\mathbf{x}$.

2.4 Training

Training neural link predictors amounts to minimising a loss function defined over a target graph $\mathcal{G}$ of triples $\{(r, a_1, a_2)\}$. Since such graphs usually only contain positive examples (true facts), a common strategy is to generate negative examples by *corrupting* the triples in the graph [Rendle et al., 2009, Bordes et al., 2013, Yang et al., 2015, Nickel et al., 2016].

Formally, given a triple (r, a_1, a_2), negative examples are generated by a corruption process δ defined by:

$$\begin{aligned} &\delta(r, a_1, a_2) \triangleq \\ &\quad \{(r, \tilde{a_1}, a_2) \mid \tilde{a_1} \in \mathcal{E}\} \cup \{(r, a_1, \tilde{a_2}) \mid \tilde{a_2} \in \mathcal{E}\}. \end{aligned}$$

The main motivation for this negative sampling strategy is the Local Closed World Assumption (LCWA) [Dong et al., 2014]. According to the LCWA, if a triple exists in the graph, other triples obtained by corrupting either the subject or the object of the triples not appearing in the graph can be considered as negative examples.

Similarly to Bordes et al. [2013], we learn the model parameters $\boldsymbol{\theta}$ and $\boldsymbol{\gamma}$ by minimising a *hinge loss* $\mathcal{J}_{\mathcal{F}}$, referred to as *fact loss*, defined over positive and negative examples:

$$\mathcal{J}_{\mathcal{F}}(\mathcal{G}; \boldsymbol{\theta}, \boldsymbol{\gamma}) \triangleq \sum_{(r,a_1,a_2)\in\Omega} \left[1 - y_r(a_1, a_2) \cdot \phi_r^{\boldsymbol{\theta}}(\mathbf{h}^{\boldsymbol{\gamma}}(a_1), \mathbf{h}^{\boldsymbol{\gamma}}(a_2))\right]_+, \quad (1)$$

where $\mathcal{N} \triangleq \{\tilde{\mathbf{x}} \mid \mathbf{x} \in \mathcal{G} \wedge \tilde{\mathbf{x}} \in \delta(\mathbf{x})\}$ is the set of negative examples (triples) generated by corrupting the triples in $\mathcal{G}$, $\Omega \triangleq \mathcal{G} \cup \mathcal{N}$ is a set containing both positive and negative examples, and $y_r(a_1, a_2) = \pm 1$ is an indicator function with value 1 if $(r, a_1, a_2) \in \mathcal{G}$, and -1 otherwise.

For several neural link prediction models, the fact scores can be trivially increased by increasing the magnitude of entity embeddings. A common solution is to either regularise the entity representations [Yang et al., 2015, Trouillon et al., 2016], or require them to live in subspaces such as the unit cube [Demeester et al., 2016] $\{\mathbf{h} \mid \mathbf{h} \in [0, 1]^k\}$, or in the unit sphere [Bordes et al., 2013] $\{\mathbf{h} \mid \|\mathbf{h}\|_2^2 = 1\}$.

3 ADVERSARIAL SETS

Even though the training data may be consistent with various assumptions we can make about the graph, on sets of unseen triples the local nature of the classifiers may still lead to inconsistencies. Taking the IS-A example, we may see a high score for (IS-A, CAT, FELINE) and (IS-A, FELINE, ANIMAL), but a low score for (IS-A, CAT, ANIMAL), violating the transitivity property of the IS-A hypernymy relation.

To address this problem, we generate *adversarial input sets*, and encourage the model to fix its inconsistencies with respect to these inputs. More specifically, we find a set of adversarial entity embeddings as inputs to the scoring layer of the model, instead of a set of actual entity pairs. This has two core benefits. First, it allows us to solve a continuous optimisation problem (over embeddings) as opposed to a combinatorial one (over actual entities). The former can even have closed form solutions, as shown in Sect. 3.3. Second, it forces the model to learn general correlations between relations, as opposed to knowledge about specific facts and entities through the encoder.

For clarity, we now consider a single assumption $\mathcal{A}$, such as the transitivity of relation r. Generalising to multiple assumptions only requires instantiating one adversary and one inconsistency loss for each of the assumptions. In this work, we use Horn Clauses, a subset of FOL formulae, to express our assumptions. For example, transitivity of the hypernym relation can be expressed by:

$$\text{IS-A}(X_1, X_2) \wedge \text{IS-A}(X_2, X_3) \Rightarrow \text{IS-A}(X_1, X_3), \quad (2)$$

where the atom on the right-hand side of the implication is referred to as the *head* of the clause, the conjunction of atoms on the left-hand side is referred to as the *body* of the clause, and all variables are universally quantified.

Let an *adversarial input set* $\mathcal{S}$ define a mapping from the free variables in $\mathcal{V}$ in $\mathcal{A}$ to k-dimensional embeddings – *i.e.* $\mathcal{S} : \mathcal{V} \mapsto \mathbb{R}^k$. We call $\mathcal{S}$ a *set* because it implicitly specifies a set of adversarial inputs to the scoring layers of the neural link predictors associated with the atoms in $\mathcal{A}$. For example, in the case of the transitivity clause in Eq. (2), a mapping $\mathcal{S}$ with $\mathcal{S}(X_1) = \mathbf{h}_1$, $\mathcal{S}(X_2) = \mathbf{h}_2$ and $\mathcal{S}(X_3) = \mathbf{h}_3$ will define the set of inputs $(\mathbf{h}_1, \mathbf{h}_2)$, $(\mathbf{h}_2, \mathbf{h}_3)$ and $(\mathbf{h}_1, \mathbf{h}_3)$ to the scoring layer $\phi_{\text{IS-A}}$.

Given the adversarial input set $\mathcal{S}$, the *inconsistency loss* $\mathcal{J}_{\mathcal{I}}(\mathcal{A}; \boldsymbol{\theta}, \mathcal{S})$ measures the degree to which assumption $\mathcal{A}$ is violated on $\mathcal{S}$ with model parameters $\boldsymbol{\theta}$. It is computed as a function of the neural link prediction scores on the adversarial inputs in $\mathcal{S}$. For the transitivity clause in Eq. (2), the inconsistency loss is computed as a function of $\phi_{\text{IS-A}}(\mathbf{h}_1, \mathbf{h}_2)$, $\phi_{\text{IS-A}}(\mathbf{h}_2, \mathbf{h}_3)$ and $\phi_{\text{IS-A}}(\mathbf{h}_1, \mathbf{h}_3)$.

Our loss function is then a linear combination of the fact loss function in Eq. (1) and the inconsistency loss $\mathcal{J}_{\mathcal{I}}$:

$$\mathcal{J}(\mathcal{G}, \mathcal{A}; \boldsymbol{\theta}, \boldsymbol{\gamma}, \mathcal{S}) \triangleq \mathcal{J}_{\mathcal{F}}(\mathcal{G}; \boldsymbol{\theta}, \boldsymbol{\gamma}) + \alpha \mathcal{J}_{\mathcal{I}}(\mathcal{A}; \boldsymbol{\theta}, \mathcal{S}),$$

where $\alpha \in \mathbb{R}$ controls the extent to which the assumption $\mathcal{A}$ should be enforced.

Our adversarial training algorithm attempts to find input sets $\mathcal{S}$ with maximal inconsistency, and model parameters $\boldsymbol{\theta}, \boldsymbol{\gamma}$ that minimise such an inconsistency. This can be formalised by the following minimax problem:

$$\min_{\boldsymbol{\theta},\boldsymbol{\gamma}} \max_{\mathcal{S}} \mathcal{J}(\mathcal{G}, \mathcal{A}; \boldsymbol{\theta}, \boldsymbol{\gamma}, \mathcal{S}). \quad (3)$$

Note that the search over possible $\mathcal{S}$ needs to take into account the unit sphere or cube constraints mentioned earlier: for any variable $X_i \in \mathcal{V}$, the corresponding k-dimensional embedding $\mathcal{S}(X_i)$ should live on the same subspace (*e.g.* unit sphere or unit cube) as the entity embeddings.

To instantiate this framework we need to be able to map an assumption $\mathcal{A}$ to an inconsistency loss $\mathcal{J}_{\mathcal{I}}(\mathcal{A}; \boldsymbol{\theta}, \mathcal{S})$, and to solve the optimisation problem in Eq. (3).

3.1 Inconsistency Losses

Given an assumption $\mathcal{A}$ expressed as a FOL clause $\text{BODY} \Rightarrow \text{HEAD}$, as in Eq. (2), our goal is defining a loss term $\mathcal{J}_{\mathcal{I}}(\mathcal{S})$ that assesses the degree to which $\mathcal{A}$ is violated on a set of adversarial inputs $\mathcal{S}$.

Recall that we represent $\mathcal{S} : \mathcal{V} \mapsto \mathbb{R}^k$ as a binding of the free variables $\mathcal{V}$ in $\mathcal{A}$ to adversarially-trained embeddings in $\mathbb{R}^k$. This means that in practice we have to search over variable-to-embedding bindings.

We construct the inconsistency loss $\mathcal{J}_{\mathcal{I}}$ compositionally, by first calculating $\phi(\text{HEAD})$ and $\phi(\text{BODY})$, respectively representing the scores for the head and body of the clause, based on the binding of variables to embeddings defined by $\mathcal{S}$. Subsequently, we test whether the head score is lower than the body score, *i.e.*, $\phi(\text{HEAD}) < \phi(\text{BODY})$ [Demeester et al., 2016]. If so, we assign a penalty proportional to the margin between body score and the head score. This yields the following inconsistency loss:

$$\mathcal{J}_{\mathcal{I}}(\text{BODY} \Rightarrow \text{HEAD}) \triangleq \left[\phi(\text{BODY}) - \phi(\text{HEAD})\right]_+ .$$

The motivation for this loss is that implications can be understood as "whenever the body is true, the head has to be true as well". In terms of neural link prediction models, this translates into "the score of the head should at least be as large as the score of the body".

For calculating the inconsistency loss, we need to specify how to calculate the scores of the head and body. To score a single atom, we simply map the free variables with the corresponding embeddings contained in the adversarial set $\mathcal{S}$ and apply the neural link predictor scoring function:

$$\phi\left(r\left(X_i, X_j\right)\right) = \phi_r\left(\mathcal{S}(X_i), \mathcal{S}(X_j)\right).$$

This gives us the score of the head atom, and the scores of the atoms within the body. Similarly to the product t-norm used in Rocktäschel et al. [2015], we use the Gödel t-norm, a continuous generalisation of the conjunction operator in logic [Gupta and Qi, 1991] to score the body of a clause, *i.e.*, a conjunction of several atoms:

$$\phi(A \wedge B) \triangleq \min\{\phi(A), \phi(B)\},$$

where A and B are clause atoms. This allows us to back-propagate through a conjunction of atoms. For disjunction one can use $\phi(A \vee B) \triangleq \max\{\phi(A), \phi(B)\}$, and for negation $\phi(\neg A) \triangleq -\phi(A)$, which allows the use of arbitrary function-free FOL clauses as in Rocktäschel et al. [2015]. However, in our experiments we only use Horn clauses, and leave the investigation of more complex clauses for future work.

Algorithm 1 Solving the minimax problem in Eq. (3) via Projected Stochastic Gradient Descent

Require: No. of epochs $\{\tau_a, \tau_d, \tau\}$, learning rates $\{\boldsymbol{\eta}^a, \boldsymbol{\eta}\}$
1: **main** $\text{AdversarialSetTraining}(\mathcal{A})$
2: Randomly initialise $\{\boldsymbol{\theta}_0, \boldsymbol{\gamma}_0\}$
3: **for** $i \in \{1, \ldots, \tau\}$ **do**
4: $\mathcal{S}_i \leftarrow \text{FindAdversarialSet}(\mathcal{A}, \boldsymbol{\theta}_{i-1})$
5: $(\boldsymbol{\theta}_i, \boldsymbol{\gamma}_i) \leftarrow \text{TrainDiscriminator}(\mathcal{G}, \boldsymbol{\theta}_{i-1}, \boldsymbol{\gamma}_{i-1}, \mathcal{S}_i)$
6: **end for**
7: **return** $\boldsymbol{\theta}_\tau, \boldsymbol{\gamma}_\tau$
8: **end main**
9: **function** $\text{FindAdversarialSet}(\mathcal{A}, \boldsymbol{\theta})$
10: Randomly initialise $\mathcal{S}_0$
11: **for** $i = 1, \ldots, \tau_a$ **do**
12: $\mathbf{h}_j \leftarrow \text{proj}(\mathbf{h}_j),\ \forall \mathbf{h}_j \in \mathcal{S}$
13: $g_i \leftarrow \nabla_{\mathcal{S}} \mathcal{J}_{\mathcal{I}}(\mathcal{A}; \boldsymbol{\theta}, \mathcal{S}_{i-1})$
14: $\mathcal{S}_i \leftarrow \mathcal{S}_{i-1} + \eta_i^a g_i$
15: **end for**
16: **return** $\mathcal{S}_{\tau_a}$
17: **end function**
18: **function** $\text{TrainDiscriminator}(\mathcal{G}, \boldsymbol{\theta}, \boldsymbol{\gamma}, \mathcal{S})$
19: $\boldsymbol{\theta}_0 \leftarrow \boldsymbol{\theta}, \boldsymbol{\gamma}_0 \leftarrow \boldsymbol{\gamma}$
20: **for** $i = 1, \ldots, \tau_d$ **do**
21: $\mathbf{h}_j \leftarrow \text{proj}(\mathbf{h}_j),\ \forall j \in \{1, \ldots, |\mathcal{E}|\}$
22: $g_i \leftarrow \nabla_{\langle \boldsymbol{\theta}, \boldsymbol{\gamma} \rangle} \mathcal{J}(\mathcal{G}, \mathcal{A}; \boldsymbol{\theta}_{i-1}, \boldsymbol{\gamma}_{i-1}, \mathcal{S})$
23: $(\boldsymbol{\theta}_i, \boldsymbol{\gamma}_i) \leftarrow (\boldsymbol{\theta}_{i-1}, \boldsymbol{\gamma}_{i-1}) - \eta_i g_i$
24: **end for**
25: **return** $\boldsymbol{\theta}_{\tau_d}, \boldsymbol{\gamma}_{\tau_d}$
26: **end function**

3.2 Optimisation

To optimise the minimax objective in Eq. (3), we alternate between two optimisation processes, as shown in Alg. 1. On line 4, the algorithm finds an adversarial set $\mathcal{S}$ by maximising the inconsistency loss using τ_a-many Gradient Ascent iterations. On line 5, the link prediction model is trained by jointly minimising the fact loss and the inconsistency loss on the adversarial input set $\mathcal{S}$ via τ_d-many Stochastic Gradient Descent iterations.

In our implementation of Alg. 1, we enforce all entity and adversarial embeddings to live either on the unit cube, *i.e.*, $\text{proj}(\mathbf{h}) \triangleq \min(\max(\mathbf{h}, \mathbf{0}), \mathbf{1})$, or on the unit sphere, *i.e.*, $\text{proj}(\mathbf{h}) \triangleq \mathbf{h} / \|\mathbf{h}\|_2$. At the beginning of the training process, we initialise the neural link predictor parameters $\{\boldsymbol{\theta}, \boldsymbol{\gamma}\}$ using uniform Xavier initialisation [Glorot and Bengio, 2010]. When searching for the adversarial set $\mathcal{S}$ that maximises the inconsistency loss, we initialise $\mathcal{S}$ using randomly sampled entity embeddings. Note that this algorithm is independent of the specific neural link prediction model used, and applicable to any function-free FOL clause.

Table 1: Closed form expressions $\mathcal{J}_{\mathcal{I}}^{\max}$ for DISTMULT and COMPLEX on three types of clauses. For DISTMULT $\boldsymbol{\delta} = \boldsymbol{\theta}_b - \boldsymbol{\theta}_r \in \mathbb{R}^k$, and for COMPLEX, $\boldsymbol{\delta} = \boldsymbol{\theta}_b - \boldsymbol{\theta}_r = \overline{\boldsymbol{\zeta}} \in \mathbb{C}^k$, for entity embeddings restricted to the unit sphere (left) and unit cube (right) subspaces.

Clause	Model	Unit Sphere	Unit Cube
$r(X_1, X_2)$	DISTMULT	0	0
$\Rightarrow r(X_2, X_1)$	COMPLEX	$\max_i \{2\lvert\boldsymbol{\theta}^{\mathrm{I}}_{r,i}\rvert\}$	$2\sum_i \lvert\boldsymbol{\theta}^{\mathrm{I}}_{r,i}\rvert$
$b(X_1, X_2)$	DISTMULT	$\max_i \{\lvert\delta_i\rvert\}$	$\sum_i \max\{0, \delta_i\}$
$\Rightarrow r(X_1, X_2)$	COMPLEX	$\max_i \left\{\sqrt{(\delta^{\mathrm{R}}_i)^2 + (\delta^{\mathrm{I}}_i)^2}\right\}$	$\sum_i \max(0, \delta^{\mathrm{R}}_i) + \max(\delta^{\mathrm{R}}_i, \lvert\delta^{\mathrm{I}}_i\rvert)$
$b(X_1, X_2)$	DISTMULT	$\max_i \{\lvert\delta_i\rvert\}$	$\sum_i \max\{0, \delta_i\}$
$\Rightarrow r(X_2, X_1)$	COMPLEX	$\max_j \left\{\sqrt{(\zeta^{\mathrm{R}}_i)^2 + (\zeta^{\mathrm{I}}_i)^2}\right\}$	$\sum_i \max(0, \zeta^{\mathrm{R}}_i) + \max(\zeta^{\mathrm{R}}_i, \lvert\zeta^{\mathrm{I}}_i\rvert)$

3.3 Closed Form Solutions

While the algorithm above is much more efficient than grounding out the clauses over the entity space (as in Rocktäschel et al. [2015]), it still requires the inner loop of maximising the inconsistency loss. Compared to closed-form analytical solutions, the inner optimisation loop can be computationally less efficient, require more hyperparameters (learning rate, number of iterations etc.) and offer fewer guarantees on whether the global optimum is found.

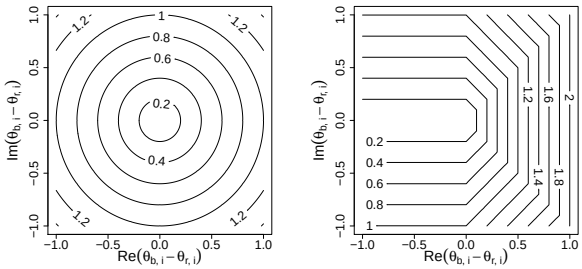
In some cases, it is possible to analytically calculate the solution $\mathcal{J}_{\mathcal{I}}^{\max}(\mathcal{A}; \boldsymbol{\theta})$ to the inner optimisation problem of maximising the inconsistency loss $\mathcal{J}_{\mathcal{I}}(\mathcal{A}; \boldsymbol{\theta}, \mathcal{S})$ with respect to the adversarial set $\mathcal{S}$:

$$\mathcal{J}_{\mathcal{I}}^{\max}(\mathcal{A}; \boldsymbol{\theta}) \triangleq \max_{\mathcal{S}} \mathcal{J}_{\mathcal{I}}(\mathcal{A}; \boldsymbol{\theta}, \mathcal{S}).$$

When $\mathcal{J}_{\mathcal{I}}^{\max}$ is known up front, the inner training loop disappears. We will call this approach the *Closed-Form* Adversarial Set Regularisation (*cASR*) method, as opposed to the more general iterative method in Alg. 1. We derive $\mathcal{J}_{\mathcal{I}}^{\max}$ for several types of clauses, both for DISTMULT and COMPLEX, as shown in Tab. 1. Full derivations are provided in the supplementary material.

Let's consider the closed-form inconsistency loss $\mathcal{J}_{\mathcal{I}}^{\max}$ for simple clauses of the form $b(X_1, X_2) \Rightarrow r(X_1, X_2)$, and let $\boldsymbol{\theta}_b, \boldsymbol{\theta}_r$ denote the predicate embeddings of predicates b and r, respectively. From Tab. 1 we can see that the expressions in the unit sphere case are independent of the sign of $\boldsymbol{\theta}_{b,i} - \boldsymbol{\theta}_{r,i}$, indicating that the non-symmetric implications cannot be explicitly modelled with unit sphere constraints. Both our theoretical and experimental results indicate that unit cube constraints used by Demeester et al. [2016] are indeed better suited for rule injection. Also note that the lifted clause injection method by Demeester et al. [2016] can be seen as a special case of this formulation, which is however limited to a single neural link prediction model and a small subset of clauses.

Figure 2: $\mathcal{J}_{\mathcal{I}}^{\max}$ contours for simple implications with COMPLEX, for varying $\boldsymbol{\theta}_{b,i} - \boldsymbol{\theta}_{r,i}$, for unit sphere (left) and unit cube (right) constraints.



(a) Unit Sphere (b) Unit Cube

Fig. 2 shows contours of the contribution to $\mathcal{J}_{\mathcal{I}}^{\max}$ from component $\boldsymbol{\theta}_{b,i} - \boldsymbol{\theta}_{r,i}$, for simple implications with COMPLEX. We observe the rotation-symmetric behaviour for the unit sphere constraints (Fig. 2a). In contrast, for unit cube constraints (Fig. 2b) we see the quite different impact of the real part (on its own a scaled version of the expression for DISTMULT) and that the imaginary part contributes as soon as its absolute value exceeds the real part.

Minimising $\mathcal{J}_{\mathcal{I}}^{\max}$ for unit sphere constraints in COMPLEX boils down to encouraging $\boldsymbol{\theta}_{r,i} \approx \boldsymbol{\theta}_{b,i}$, for both real and imaginary parts, whereas for unit cube constraints it encourages an ordering relation of the real parts $\mathrm{Re}\,(\boldsymbol{\theta}_{r,i}) \geq \mathrm{Re}\,(\boldsymbol{\theta}_{b,i})$, and the equality of the imaginary parts $\mathrm{Im}\,(\boldsymbol{\theta}_{r,i}) \approx \mathrm{Im}\,(\boldsymbol{\theta}_{b,i})$.

4 RELATED WORK

Rocktäschel et al. [2014, 2015] provide a framework for jointly maximising the probability of observed facts and propositionalised First-Order Logic clauses. Wang et al. [2015] show how different types of clauses can be included after training the model by using Integer Linear Programming. Recently, Wang and Cohen [2016] propose a method for embedding facts and clauses using matrix factorisation. However, all these approaches ground the clauses in the training data. This limits their scalability to comparably small knowledge bases that contain only few entities. As mentioned in Demeester et al. [2016], such problems provide an important motivation for *lifted* clause injection methods that do not rely on grounding of clauses, but rather regularise relation representations directly and ensure that the assumptions hold on the whole entity embedding space, improving generalisation. Dong et al. [2014], Nickel et al. [2014] and Wang et al. [2015] propose combining observable patterns in the form of clauses and latent features for link prediction tasks. However, in these models, clauses are

not used for learning better distributed representations of entities and relation types.

Our model is related to MODEL FSL proposed by Demeester et al. [2016], where they use very simple clauses for defining a partial ordering among relation embeddings. Moreover, their approach is limited to the simple matrix factorisation MODEL F by Riedel et al. [2013a], while ASR extends and improves over MODEL FSL. First, it can be used for injecting arbitrarily complex first-order logic clauses, such as the transitivity clause in Eq. (2). Second, it can be used jointly with any knowledge graph embedding model. Furthermore, it also improves over the general clause-injection methods by Rocktäschel et al. [2015] and Guo et al. [2016] since it avoids generating possible groundings of a clause. Gaifman models [Niepert, 2016] and related works train latent representations using features derived from subregions of a knowledge base: in such models, one can define relationships between subregions.

5 EXPERIMENTS

We now present our experimental results, starting by describing the experimental setup and hyper-parameters, both on synthetic and real-world data-sets. All models were implemented in TensorFlow, and source code is available on-line.[1]

Experimental Setup for Synthetic Data In order to investigate the effect of ASR on particular types of clauses, we created the following small synthetic knowledge base. We randomly sampled subject-object pairs from a low-dimensional entity space (with probability 0.1, and for $|\mathcal{E}| = 30$). With each of these pairs, we created facts by combining them with 15 relations in $\mathcal{R}$ with probability 0.1, resulting in 135 training facts. Besides those, we created clauses by randomly combining relations into the appropriate clause template, *i.e.* symmetry constraints, implications, and transitivity. For each experiment on a particular type of clause, 10 different clauses of that type were applied together. The training facts contain no evidence of these clauses. Each new fact that could be inferred based on the clauses was added as a positive item to the test data, together with a negative example obtained by corrupting its subject-object pair.

Experimental Setup for Real Data We also evaluate the proposed method on WORDNET (WN18) and FREEBASE (FB122) jointly with the set of rules released by Guo et al. [2016]. WORDNET [Miller, 1995] is a lexical knowledge base for the English language, where entities correspond to word senses, and relationships define lexical relations between them. The WN18 dataset consists of a subset of WORDNET, containing 40,943 entities, 18 relation types, and 151,442 triples. FREEBASE [Bollacker et al., 2007] is a large knowledge graph that stores general facts about the world. The FB122 dataset is a subset of FREEBASE regarding the topics of *people*, *location* and *sports*, and contains 9,738 entities, 122 relation types, and 112,476 triples. For both data sets, we used the fixed training, validation, test sets and rules provided by Guo et al. [2016]; a subset of the rules is shown in Tab. 2. Note that a subset of the test triples can be inferred by deductive logic inference. For such a reason, following Guo et al. [2016], we also partition the test set in two subsets, namely Test-I and Test-II: Test-I contains triples that *cannot* be inferred by deductive logic inference, while Test-II contains all remaining test triples. Statistics for the data sets are shown in Tab. 3.

Hyper-parameters We ran a grid search for the embedding dimension $k \in \{20, 50, 100, 150, 200\}$, the margin $\gamma \in \{1, 2, 5, 10\}$, the weight of the adversarial loss $\alpha \in \{1, 10, 10^2, 10^3, 10^4\}$, the number of iterations for the discriminator and the adversary $\tau_d, \tau_a \in \{1, 10\}$, fixed the number of epochs to $\tau = 100$, and decided the optimal subspace $\mathcal{U}$ among either unit cube or unit sphere. We use AdaGrad [Duchi et al., 2011] for automatically selecting the optimal learning rate, with an initial value of 0.1. For the synthetic dataset experiments, we use the same settings, fixing the hyper-parameters to $k = 20$, $\gamma = 1$, $\alpha = 1$, $\tau_d = 10$ and $\tau_a = 1$.

Evaluation Metrics For evaluating each model, we measure the quality of the ranking of test triples in terms of *Mean Reciprocal Rank* (MRR) and *HITS@k* [Bordes et al., 2013, Nickel et al., 2016]. MRR and Hits@K are two standard evaluation measures on these data sets and are used in two settings: *raw* and *filtered* [Bordes et al., 2013]. Our results are reported in the *filtered* setting, where metrics are computed after removing all the other known triples appearing in the training, validation or test sets from the ranking. This is motivated by observing that ranking a test triple lower than another true triple should not be penalised. For the experiments on synthetic data, we calculate the Area Under the Precision-Recall Curve (AUC-PR), based on the test data consisting of true facts and a single corruption of each true fact. Each experiment is repeated 10 times with different randomly generated train facts, test facts, and clauses. The reported AUC-PR values are averaged over these runs.

[1] https://github.com/uclmr/inferbeddings

Table 2: Examples of the clauses used for FREEBASE (FB122) and WORDNET (WN18).

/PEOPLE/PERSON/NATIONALITY(X_1, X_2) ∧ /LOCATION/COUNTRY/OFFICIAL_LANGUAGE(X_2, X_3) ⇒ /PEOPLE/PERSON/LANGUAGES(X_1, X_3)	
/COUNTRY/ADMINISTRATIVE_DIVISIONS(X_1, X_2) ∧ /ADMINISTRATIVE_DIVISION/CAPITAL(X_2, X_3) ⇒ /LOCATION/CONTAINS(X_1, X_3)	
/LOCATION/COUNTRY/CAPITAL(X_1, X_2) ⇒ /LOCATION/LOCATION/CONTAINS(X_1, X_2)	
_HYPERNYM(X_1, X_2) ⇒ _HYPONYM(X_2, X_1)	_HYPONYM(X_1, X_2) ⇒ _HYPERNYM(X_2, X_1)
_HAS_PART(X_1, X_2) ⇒ _PART_OF(X_2, X_1)	_PART_OF(X_1, X_2) ⇒ _HAS_PART(X_2, X_1)

Table 3: Statistics for the data sets.

Dataset	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{R}\rvert$	#Train	#Valid	#Test-I	#Test-II
FB122	9,738	122	91,638	9,595	5,057	6,186
WN18	40,943	18	141,442	5,000	1,394	3,606

6 RESULTS AND DISCUSSION

In this section, we describe our findings on the synthetic, WORDNET (WN18) and FREEBASE (FB122) datasets.

6.1 Synthetic Data

ASR We tested the effectiveness of adversarial training for five different types of clauses in the synthetic data setup described in Section 5. Table 6 shows the resulting AUC-PR, where entity embeddings lie on the unit cube or on the unit sphere, and with vs. without iterative adversarial training ($\alpha = 0$ and $\alpha = 1$, respectively). DISTMULT and COMPLEX are able to encode the various types of clauses into the relation and argument representations. In line with arguments given above, the results for unit cube entities are generally better than their counterparts on the unit sphere, both on the standard models and with ASR. The most complex type of clauses (*i.e.* the transitivity over three different relations) yields lower absolute AUC-PR values than simpler clauses. Nonetheless, we observe a significant increase in scores due to the clauses. A noticeable case where the standard models cannot be improved is for DISTMULT on the clause that expresses symmetry (the first clause in Table 1). This is unsurprising since symmetry is satisfied by construction in DISTMULT. This experiment confirms that ASR is able to encode different types of clauses (unlike Demeester et al. [2016]), and for different models (unlike Rocktäschel et al. [2015]).

Closed-Form Solutions Using the closed-form expressions for $\mathcal{J}_{\mathcal{I}}^{\max}$ given in Table 1, we performed the synthetic data set experiments again. The results, shown in Table 7, indicate that the optimal adversarial training is also able to encode clauses into the trained embeddings. As with the iterative method, unit cube constraints perform better than on the unit sphere. For the synthetic data experiments, no hyper-parameter optimisation was performed, and we used the same number of discriminator training cycles as for the results presented in Table 6. Compared to the iterative method, closed form results on the unit sphere are consistently weaker. This is in line with our earlier observations that their symmetric character is not suited for modelling asymmetry in clauses. In any case, we observed a strong reduction in training time: the runs based on closed-form expressions took around a fifth the time of the corresponding iterative runs.

6.2 Link Prediction in Freebase and Wordnet

We compare the proposed Adversarial Set Regularisation (ASR) method with KALE, a recently proposed neural link prediction model that also leverages rules during the training process, the Translating Embeddings model (TRANSE) [Bordes et al., 2013], DISTMULT and COMPLEX. In particular, we compare ASR with two strong KALE variants: KALE-PRE, which augments training triples by means of logic inference, and KALE-JOINT, which jointly considers training triples and rules during training [Guo et al., 2016]. Results for TRANSE and KALE are reported from Guo et al. [2016].

In experiments, we use ASR for regularising the learning process in DISTMULT and COMPLEX – we denote the resulting models by *ASR*-DISTMULT and *ASR*-COMPLEX: we retain the original formulations of the scoring functions ϕ_r, but train the models by solving the minimax optimisation problem in Eq. (3) using Alg. 1.

Results for FREEBASE are reported in Tab. 4. We can see that *ASR*-COMPLEX yields better results both in comparison with KALE and with the un-regularised model COMPLEX. When comparing DISTMULT with its ASR-regularised extension, we note an improvement from 0.628 to 0.675 for MRR, and from 72.9% to 75.2% for Hits@10. Similarly, when comparing COMPLEX with its extension *ASR*-COMPLEX, we note an improvement from 0.641 to 0.698 for MRR, and from 71.9% to 75.7% for Hits@10. Also note that ASR, when used jointly with COMPLEX and DISTMULT, yields larger relative improvements in comparison with KALE (a model inspired by TRANSE) and TRANSE.

Improvements are even more evident if we consider the FB122 results in Test-II, where test facts are directly related to the logic clauses. We can see that, in terms of MRR, the proposed regularisation method improves

Table 4: Link prediction results on the Test-I, Test-II and and Test-ALL on FB122, filtered setting.

		Test-I				Test-II				Test-ALL			
		Hits@N (%)			MRR	Hits@N (%)			MRR	Hits@N (%)			MRR
		3	5	10		3	5	10		3	5	10	
FB122	TRANSE [Bordes et al., 2013]	36.0	41.5	48.1	0.296	77.5	82.8	88.4	0.630	58.9	64.2	70.2	0.480
	KALE-PRE [Guo et al., 2016]	35.8	41.9	49.8	0.291	82.9	86.1	89.9	0.713	61.7	66.2	71.8	0.523
	KALE-JOINT [Guo et al., 2016]	**38.4**	**44.7**	**52.2**	0.325	79.7	84.1	89.6	0.684	61.2	66.4	72.8	0.523
	DISTMULT [Yang et al., 2015]	36.0	40.3	45.3	0.313	92.3	93.8	94.7	0.874	67.4	70.2	72.9	0.628
	ASR-DISTMULT	36.3	40.3	44.9	0.330	98.0	99.0	99.2	0.948	70.7	73.1	75.2	0.675
	cASR-DISTMULT	37.0	40.4	45.1	0.337	96.7	98.6	99.3	0.933	70.1	72.7	75.1	0.669
	COMPLEX [Trouillon et al., 2016]	37.0	41.3	46.2	0.329	91.4	91.9	92.4	0.887	67.3	69.5	71.9	0.641
	ASR-COMPLEX	37.3	41.0	45.9	0.338	**99.2**	**99.3**	**99.4**	**0.984**	**71.7**	**73.6**	**75.7**	**0.698**
	cASR-COMPLEX	37.9	41.7	46.2	**0.339**	97.7	**99.3**	**99.4**	0.954	71.1	**73.6**	75.6	0.680

Table 5: Results on WN18 with limited training data – *i.e.* 20%, 30%, 40% and 50% of the training set.

	Training data		20%	30%	40%	50%
WN18	Hits@10	COMPLEX	38.9	46.6	54.1	60.5
		ASR-COMPLEX	**39.9**	**47.1**	**54.8**	**61.1**
	MRR	COMPLEX	0.356	0.418	0.480	0.538
		ASR-COMPLEX	**0.366**	**0.423**	**0.484**	**0.540**

Table 6: AUC-PR results for *ASR*-DISTMULT and *ASR*-COMPLEX on synthetic datasets with various types of clauses (with $r \neq s \neq t$). Comparison of standard models without clauses ($\alpha = 0$) and iterative adversarial training with clauses ($\alpha = 1$), with unit cube or unit sphere constraints on the entity embeddings.

Clauses	Model	$\alpha = 0$ Unit Cube	$\alpha = 0$ Unit Sphere	$\alpha = 1$ Unit Cube	$\alpha = 1$ Unit Sphere
$r(X_1, X_2)$	DISTMULT	**96.0**	95.7	95.9	95.6
$\Rightarrow r(X_2, X_1)$	COMPLEX	45.0	42.4	90.6	**91.2**
$r(X_1, X_2)$	DISTMULT	59.2	61.3	**86.3**	79.2
$\Rightarrow s(X_1, X_2)$	COMPLEX	60.8	61.2	**85.8**	79.7
$r(X_1, X_2)$	DISTMULT	59.6	59.1	**76.6**	73.1
$\Rightarrow s(X_2, X_1)$	COMPLEX	54.7	51.4	**87.5**	81.0
$r(X_1, X_2) \wedge r(X_2, X_3)$	DISTMULT	45.0	37.5	**65.9**	45.8
$\Rightarrow r(X_1, X_3)$	COMPLEX	40.9	36.3	**54.8**	42.5
$r(X_1, X_2) \wedge s(X_2, X_3)$	DISTMULT	41.9	39.5	44.9	**49.8**
$\Rightarrow t(X_1, X_3)$	COMPLEX	39.1	37.8	40.3	**45.4**

Table 7: AUC-PR results on synthetic datasets for adversarial training with closed form expressions.

Clause	Model	$\alpha = 1$ Unit Cube	$\alpha = 1$ Unit Sphere
$r(X_1, X_2)$	DISTMULT	**97.3**	95.2
$\Rightarrow r(X_2, X_1)$	COMPLEX	**91.7**	90.1
$r(X_1, X_2)$	DISTMULT	**85.0**	69.5
$\Rightarrow s(X_1, X_2)$	COMPLEX	**83.6**	75.5
$r(X_1, X_2)$	DISTMULT	**76.9**	67.8
$\Rightarrow s(X_2, X_1)$	COMPLEX	**80.2**	73.9

the MRR from 0.874 to 0.948 for DISTMULT, and from 0.887 to 0.984 for COMPLEX.

We ran the same experiments on WN18 (see the supplemental material for more exhaustive results) but did not notice any significant improvements. One reason can be that the neural link prediction model has enough training data to *learn* the WN18 rules by itself. For this reason, we also evaluate ASR in settings where the availability of training data is limited. More specifically, at training time, we use a limited sample of the training triples (*i.e.* 20%, 30%, ...) whose predicate appears in the head of a clause. Results are available in Tab. 5: we note that, in a limited data regime, ASR yields some marginal improvements on WN18. In Tab. 4 we also report results for the closed-form solutions described in Sect. 3.3 and derived in the Appendix, denoted by *cASR*-COMPLEX and *cASR*-DISTMULT. In both cases, the closed form solutions for the inner adversarial training loop yield similar results to their iterative counterparts, improving on all baselines.

7 CONCLUSIONS

In this paper, we introduced Adversarial Set Regularisation (ASR), a general and scalable method for regularising neural link prediction models. In the proposed method, an assumption is used for deriving an *inconsistency loss* measuring the degree to which the model violates the assumption on an adversarially-trained set of input representations. The training objective is defined as a zero-sum game, where an adversary finds the most offending set of input representations by maximising the inconsistency loss, and the model uses the inconsistency loss on such a set for regularising its training process. Our results demonstrate that incorporating prior assumptions in the form of FOL clauses gives a steady improvement over neural link prediction models, especially when the availability of training data is limited. Furthermore, the proposed method yields consistent improvements over the recently-proposed KALE method.

Acknowledgements

We are immensely grateful to Jeff Mitchell, Johannes Welbl, and Jason Naradowsky for useful discussions. This work has been supported by the Research Foundation - Flanders (FWO), Ghent University - iMinds, a Google PhD Fellowship in Natural Language Processing, an Allen Distinguished Investigator Award, and a Marie Curie Career Integration Award.

References

K. D. Bollacker, R. P. Cook, and P. Tufts. Freebase: A Shared Database of Structured General Human Knowledge. In *AAAI*, pages 1962–1963, 2007.

A. Bordes, N. Usunier, A. García-Durán, J. Weston, and O. Yakhnenko. Translating Embeddings for Modeling Multi-relational Data. In *NIPS*, pages 2787–2795, 2013.

K.-W. Chang, W.-t. Yih, B. Yang, and C. Meek. Typed Tensor Decomposition of Knowledge Bases for Relation Extraction. In *EMNLP*, 2014.

N. N. Dalvi, P. M. Domingos, Mausam, S. K. Sanghai, and D. Verma. Adversarial Classification. In *KDD*, pages 99–108. ACM, 2004.

T. Demeester, T. Rocktäschel, and S. Riedel. Lifted rule injection for relation embeddings. In *EMNLP*, pages 1389–1399, 2016.

X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang. Knowledge Vault: A Web-Scale Approach to Probabilistic Knowledge Fusion. In *KDD*, pages 601–610, 2014.

A. Dosovitskiy, J. Tobias Springenberg, and T. Brox. Learning to generate chairs with convolutional neural networks. In *ICVPR*, pages 1538–1546, 2015.

J. C. Duchi, E. Hazan, and Y. Singer. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *JMLR*, 12:2121–2159, 2011.

X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, volume 9 of *JMLR Proceedings*, pages 249–256. JMLR.org, 2010.

I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio. Generative Adversarial Nets. In *NIPS*, pages 2672–2680, 2014.

S. Guo, Q. Wang, L. Wang, B. Wang, and L. Guo. Jointly Embedding Knowledge Graphs and Logical Rules. In *EMNLP*, pages 192–202, 2016.

M. M. Gupta and J. Qi. Theory of t-norms and fuzzy inference methods. *Fuzzy Sets Syst.*, 40(3):431–450, Apr. 1991.

G. A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, 1995.

A. Neelakantan, B. Roth, and A. McCallum. Compositional Vector Space Models for Knowledge Base Completion. *arXiv preprint arXiv:1504.06662*, 2015.

M. Nickel, V. Tresp, and H.-P. Kriegel. Factorizing YAGO: Scalable Machine Learning for Linked Data. In *WWW*, pages 271–280, 2012.

M. Nickel, X. Jiang, and V. Tresp. Reducing the Rank in Relational Factorization Models by Including Observable Patterns. In *NIPS*, pages 1179–1187, 2014.

M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33, 2016.

M. Niepert. Discriminative gaifman models. In *NIPS*, pages 3405–3413, 2016.

S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *UAI*, pages 452–461. AUAI Press, 2009.

S. Riedel, L. Yao, A. McCallum, and B. M. Marlin. Relation Extraction with Matrix Factorization and Universal Schemas. In *NAACL-HLT*, pages 74–84, 2013a.

S. Riedel, L. Yao, A. McCallum, and B. M. Marlin. Relation Extraction with Matrix Factorization and Universal Schemas. In *NAACL-HLT*, pages 74–84, 2013b.

T. Rocktäschel, M. Bosnjak, S. Singh, and S. Riedel. Low-Dimensional Embeddings of Logic. In *ACL Workshop on Semantic Parsing (SP'14)*, 2014.

T. Rocktäschel, S. Singh, and S. Riedel. Injecting Logical Background Knowledge into Embeddings for Relation Extraction. In *NAACL-HLT*, pages 1119–1129, 2015.

R. Socher, D. Chen, C. D. Manning, and A. Y. Ng. Reasoning With Neural Tensor Networks for Knowledge Base Completion. In *NIPS*, pages 926–934, 2013.

C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *ICLR*, 2014.

K. Toutanova, D. Chen, P. Pantel, P. Choudhury, and M. Gamon. Representing Text for Joint Embedding of Text and Knowledge Bases. *ACL*, 2015.

T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard. Complex Embeddings for Simple Link Prediction. In *ICML*, pages 2071–2080, 2016.

E. Vylomova, L. Rimell, T. Cohn, and T. Baldwin. Take and Took, Gaggle and Goose, Book and Read: Evaluating the Utility of Vector Differences for Lexical Relation Learning. In *ACL*, 2016.

Q. Wang, B. Wang, and L. Guo. Knowledge Base Completion Using Embeddings and Rules. In *IJCAI*, pages 1859–1866, 2015.

W. Y. Wang and W. W. Cohen. Learning First-Order Logic Embeddings via Matrix Factorization. In *IJCAI*, pages 2132–2138, 2016.

B. Yang, W. Yih, X. He, J. Gao, and L. Deng. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In *ICLR*, 2015.