A Systematic Analysis of Regularisation Terms for Neural Link Prediction Models

Abstract

Regularisers are instrumental in improving the generalisation accuracy of neural link prediction models, as well as avoiding pathological solution to the loss minimisation problem. In this paper, we attempt to introduce various regularization methods to the factorization-based neural link predictors and evaluate how they impact the task of knowledge graph completion (KGC). We propose three novel regularizers, including gradient penalty, multi-task learning and manifold regularization. We conducted extensive and comprehensive experiments to analyze how the proposed methods could potentially impact KGC tasks. Based on the experiment results on benchmark datasets FB15k-237 and WN18RR, we find both gradient penalty and multi-task learning methods could bring significant generalization improvement to neural link predictors.

1 Introduction

As the result of constructing large-scale knowledge graphs (KG) such as Freebase [Bollacker *et al.*, 2008], DBpedia [Bizer *et al.*, 2009] and YAGO [Suchanek *et al.*, 2007], more entities with few or zero relations were added to the KG. These missing entries resulted in an incomplete structure of the knowledge graph. Therefore, it was crucial to investigate the implicit relationships among entities or relations to recover the missing facts and construct a complete KG for real-world applications. The research on Knowledge Graph Completion (KGC) was proposed to tackle such a problem.

During the past decade, the neural link predictor, a kind of Knowledge Graph Embedding (KGE) Model, has become more and more popular in the research of KGC tasks. This method focuses on learning low-dimensional representation (embeddings) for entities and relations based on existing triples, and then uses the learned embeddings to evaluate the plausibility of new facts through a scoring function.

However, neural link predictors are thought to have poor generalization [Trouillon *et al.*, 2016] in some particular circumstances. To get out of such a predicament, regularisation is commonly required during the training of KGE models.

In this paper, inspired by the regularisers in latent space representation models [Hoffman *et al.*, 2019; Thanh-Tung *et al.*, 2019], multi-task learning [Chen *et al.*, 2021], and matrix factorisation [Cai *et al.*, 2010], we propose 3 new regularisers for KGC tasks with the hope of improving the model generalisation. Specifically, the methods include *gradient penalty*, *multi-task learning*, and *manifold regularisation*.

2 Background and related work

A knowledge graph $\mathcal{G}$ is defined by a set of entities nodes $\mathcal{E}$ and a set of relations $\mathcal{R}$. The data stored in the knowledge graph is formed as factual triples $\langle s, r, o \rangle$, where each triple represents a connection between subject s and object o with relation type r. It is noticeable that the subjects and the objects are from the same set of entities, so the knowledge graph lies in a 3-order space, with $\mathcal{G} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$.

Knowledge Graph Completion The general Knowledge Graph Completion (KGC) tasks include complementing the missing entities, relations, or even attributes. In this paper, we are going to focus on a specific type of KGC problem called neural link prediction.

Neural link prediction intends to predict the missing entries by mining the facts in the knowledge graph and learning the representation for each entity and relation. The completion task forms its dataset as follows. The training set consists of a series of triples that are known to hold true in a knowledge graph, denoted as $\mathcal{S} = \left\{ (s_1, r_1, o_1), \ldots, \left(s_{|\mathcal{S}|}, r_{|\mathcal{S}|}, o_{|\mathcal{S}|}\right) \right\} \subseteq \mathcal{G}$. While the queries in the validation and test sets come with

the form $\langle s, r, ?\rangle$ or $\langle ?, r, o\rangle$, the model is required to find out the index of the missing entities.

We would like to answer the query $\langle s, r, ?\rangle$ (similarly to $\langle ?, r, o\rangle$) by finding the object entity o^* that has the highest conditional probability $P_\theta(o^* \mid s, r)$, where θ is the trainable parameters in the model. An intuitive way to solve this problem is by calculating $P_\theta(o' \mid s, r)$ for all $o' \in \mathcal{E}$, and finding the one with the highest probability, noted as o^*. The likelihood of $P_\theta(o \mid s, r)$ can be estimated by normalising a parametric score function $\phi_\theta(s, r, o)$:

$$P_\theta(o \mid s, r) = \frac{\exp(\phi_\theta(s, r, o))}{\sum_{o'} \exp(\phi_\theta(s, r, o'))}$$

Score function: Neural link predictors can be characterized by their scoring function ϕ_θ. Formally, we will use $\mathbf{e}_s$, $\mathbf{e}_o$ and $\mathbf{w}_r$ to denote the embedding of a subject s, object o and relation r, and in this paper we are particularly interested in the factorization-based models. For instance, DistMult [Yang *et al.*, 2015] defined the score function as $\phi_\theta(s, r, o) = \sum_k e_s^k w_r^k e_o^k := \langle \mathbf{e}_s, \mathbf{w}_r, \mathbf{e}_o\rangle$, where $\langle \cdot, \cdot, \cdot\rangle$ denotes the tensor inner product. Canonical Tensor (CP) Decomposition [Hitchcock, 1927] uses 2 distinguished representation for an entity when it is used for subject or object, and use a same tensor inner product to calculate the score function. ComplEx [Trouillon *et al.*, 2016] extends DistMult to solve the problem of symmetry and anti-symmetry by introducing complex numbers to the embeddings, which has the score function $\phi(s, r, o) = \mathrm{Re}(\langle \mathbf{e}_s, \mathbf{w}_r, \bar{\mathbf{e}}_o\rangle)$, where $\mathrm{Re}(x)$ is the real part of x. Tucker [Balazevic *et al.*, 2019] introduces a core matrix to the model, and this core tensor works as an information compression for the original tensor. It has the score function $\phi(s, r, o) = \mathcal{Z} \times_1 \mathbf{e}_s \times_2 \mathbf{w}_r \times_3 \mathbf{e}_o$.

2.1 Training objective

Neural link predictors could be trained by a large range of loss functions, e.g ranking losses, binary logistic regression or sampled multi-class log-loss. In this paper, we will follow the convention in [Lacroix *et al.*, 2018] to use multi-class log loss during the training stage. The loss function can be interpreted as the negative summation of subject and object log-likelihood,

$$\mathcal{L} = -\sum_{\langle s,r,o\rangle \in \mathcal{S}} [\log P_\theta(s \mid r, o) + \log P_\theta(o \mid s, r)] \tag{1}$$

We further introduce loss term for each training triple $\langle s, r, o\rangle$ to simplify the expression, which is $\ell_{s,r,o} = -\log P_\theta(s \mid r, o) - \log P_\theta(o \mid s, r)$.

2.2 Regularizers

Previous studies suggest that, without a regulariser, the factorisation-based models could trivially minimise the loss $\mathcal{L}$ by increasing the embeddings norm [Bordes *et al.*, 2013], which would impair the testing performance. So it is necessary to introduce regularisations to the models. Thus, the loss function with a regulariser can be written as:

$$\begin{aligned}\mathcal{L} &= \sum_{s,r,o\in S} \ell_{s,r,o} + \lambda \mathbf{R}(\mathbf{e}_s, \mathbf{e}_o, \mathbf{w}_r) \\ &= \sum_{s,r,o\in\mathcal{S}} (\ell_{s,r,o} + \sum_{\mathbf{z}\in\{\mathbf{e}_s,\mathbf{e}_o,\mathbf{w}_r\}} \lambda \mathbf{R}(\mathbf{z}))\end{aligned}$$

For simplicity, we use $\mathbf{z} \in \mathbb{C}^K$ to denote the embedding vector instead of the conventional notation $\mathbf{e}_s, \mathbf{w}_r, \mathbf{e}_o$ in the following section.

To the best of our knowledge, all prior researches on regularising neural link prediction were about norm-based methods, which aimed at preventing large values of the embedding and minimising the rank in tensor decomposition. Among them, l_1 and l_2 norm of embeddings were most frequently used. For example, the regularisers used by [Bordes *et al.*, 2013] and [Trouillon *et al.*, 2016] are simply $\mathbf{R}(\mathbf{z}) = \|\mathbf{z}\|_1$ and $\mathbf{R}(\mathbf{z}) = \|\mathbf{z}\|_2$

More recent work started to consider using tensor norm as a regulariser instead of simple embeddings norms. [Yuan and Zhang, 2016] first proposed to use tensor trace (nuclear) norm as a regulariser, and [Lacroix *et al.*, 2018] used tensor nuclear-3 norm as the regularisation. Their work suggested that the nuclear norm can work as an approximation of the tensor rank. While in the factorisation models that we consider, the nuclear-3 (n_3) norm works exactly the same as a l_3 norm of each embedding. This is a simple yet efficient method to minimising the rank in tensor factorisation and is proved to be significantly beneficial for training factorisation-based neural link predictors.

3 Regularisation term

In this paper, we will propose four novel regularisers, and investigate their impact on the neural link prediction models.

3.1 Norm-based regularisation

Inspired by the huge success of elastic net [Zou and Hastie, 2005], our first attempt is to combine l_1, l_2 and n_3 norms and define a new regulariser:

$$\mathbf{R}(\mathbf{z}) = \lambda_1 \|\mathbf{z}\|_1 + \lambda_2 \|\mathbf{z}\|_2 + \lambda_3 \|\mathbf{z}\|_3$$

This regulariser is simple a combination of different order norm-based regularisation.

3.2 Gradient Penalty

Gradient penalty has been widely applied to latent space models, e.g. Generative Adversarial Neural Networks (GANs) [Thanh-Tung *et al.*, 2019] and Variational Auto-Encoders (VAEs) [Rifai *et al.*, 2011]. Neural link predictors are also latent space

models with the encoder being embedding lookup functions and the decoder being score functions [Hamilton, 2020]. Thus, we are curious whether the gradient penalty regularisation could also work on the scheme of neural link predictors. Specifically, we consider applying gradient penalty to the decoder part (score function) since the encoder part is simply a lookup function and not differentiable.

The input vector would be the embedding vectors $\mathbf{z} \in \mathbf{e}_s, \mathbf{w}_r, \mathbf{e}_o$ and the output would the score function $\phi_\theta(s, r, o)$. Formally, denoting the model output as $y = f(\mathbf{z})$, a small perturbation applied to the input $\mathbf{z}$ can be expressed as $\mathbf{z} + \boldsymbol{\epsilon}$. According to Taylor expansion, the corresponding function output will be approximated as:

$$f(\mathbf{z} + \boldsymbol{\epsilon}) = f(\mathbf{z}) + \sum_{i=1}^{K} \epsilon_i \cdot \frac{\partial f}{\partial z_i}(\mathbf{z}) + O\left(\boldsymbol{\epsilon}^2\right)$$

If we wish to minimise the output change $f(\mathbf{z} + \boldsymbol{\epsilon}) - f(\mathbf{z})$, it is equivalent to minimising the term $\sum_{i=1}^{K} \epsilon_i \cdot \frac{\partial f}{\partial z_i}(\mathbf{z})$ by neglecting the second order infinitesimal $O\left(\epsilon^2\right)$. That is to say, the model output change caused by input perturbation $\boldsymbol{\epsilon}$ is governed by the so-called Jacobian function

$$J_i(\mathbf{z}) \equiv \frac{\partial f}{\partial z_i}(\mathbf{z}), \qquad i \in \{1, \ldots, K\}$$

.

Thus, minimising the norm of Jacobian function $\|\mathbf{J}(\mathbf{z})\|_p$ would work as a regulariser to make the model insensitive to input noise. The inputs of the model are the embeddings of the entities and the relations, $\mathbf{e}_s$, $\mathbf{w}_r$ and $\mathbf{e}_o$. By introducing the l_2 gradient penalty to our model, we can now form the new loss function as

$$\mathcal{L} = \sum_{s,r,o \in \mathcal{S}} (\ell_{s,r,o} + \sum_{\mathbf{z} \in \{\mathbf{e}_s, \mathbf{e}_o, \mathbf{w}_r\}} \lambda \|\mathbf{J}(\mathbf{z})\|_2)$$

the output function $f(\mathbf{z}) = \phi_\theta(s, r, o)$, so

$$\mathbf{J}_i(\mathbf{z}) \equiv \frac{\partial \phi_\theta(s, r, o)}{\partial \mathbf{z}} \mid_i, \qquad i \in \{1, \ldots, K\} \quad (2)$$

We calculate the Jacobian matrix w.r.t score function as in Equation (2) instead of multi-class output of the models to reduce the tensor size to be $b \times K$. In this way the computational resources are saved and we can still obtain the gradient penalty.

3.3 Multi-task Learning as a Regulariser

Graph representation learning algorithms [Hamilton, 2020], e.g. Node2Vec, Struc2Vec, use embedding to encode the graph structure. Inspired by this, we consider predicting the graph features to be helpful in the training of entity embeddings. In this part, we will manually construct graph features based on factual triples and their multi-hop relationships. And the model will be asked to predict features during training as auxiliary tasks, which can be viewed as a regularizer.

Former studies [Dobrowolska *et al.*, 2021; Galkin *et al.*, 2021] have suggested several ways to design node representation features. Based on their work, we develop three types of feature representations, respectively called in-range and in-domain (IRID) feature representation, random paths representation, and NodePiece representation.

IRID Feature Representation In-range and in-domain (IRID) feature representation utilizes the relation types to construct the features. Considering that the relation types could be highly correlated to its subject or object, we can aggregate all the relation types that are directly connected to an entity to construct a feature. Specifically, given a triple $\langle s, r, o \rangle$, we say relation r is in the range of subject s and in the domain of object o. Thus two clauses can be defined, namely in-range and in-domain:

$$\text{in-range}(e, r) = \begin{cases} 1 & \forall o, \text{if } \exists (e, r, o) \in \mathcal{S} \\ 0 & \text{otherwise} \end{cases}$$

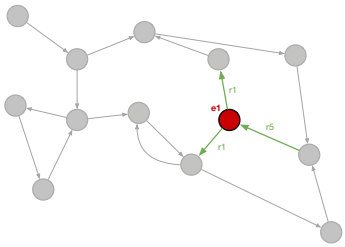
$$\text{in-domain}(e, r) = \begin{cases} 1 & \forall s, \text{if } \exists (s, r, e) \in \mathcal{S} \\ 0 & \text{otherwise} \end{cases}$$

If we use both in-range and in-domain features for all relations to represent a node entity, we can easily get a binary vector, $\vec{h} \in \mathbb{R}^{2|\mathcal{R}|}$. An example of IRID representation can be found in Figure 1a.

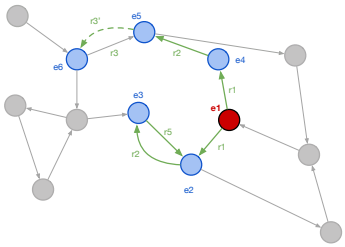
Random Paths Feature Representation The idea of Random Paths feature representation was first proposed from the work of [Das *et al.*, 2020]. The algorithm works as follows: First, for each node entity, n random paths[1] starting from this node are sampled, and each path can be expressed as a sequence of entities and relations, i.e. $p = (e_1, r_1, e_2, \ldots e_{K-1}, r_{K-1}, e_K)$, in which e_i's are the nodes this path walks through and r_i's are the relations that connect these nodes.

This path sampling method aims to build a subgraph around an entity and find out the entities and relations that are closely linked to the target node. In NodePiece representation we will consider the entities, but for now, the feature vector is constructed only by the types of relations a path travels along. The direction of the relation (forward and inverse) is considered in our work, which leads to a vector of size $|\mathcal{E}| \times 2|\mathcal{R}|$. For simplicity, we still construct a binary feature, where 1 represents the case when a relation appears in the sampled path p, and 0 otherwise. An example is in Figure 1b.

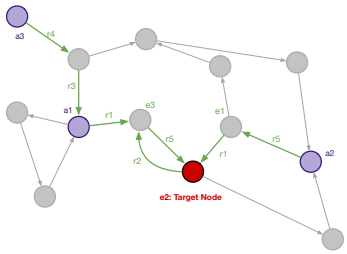
[1] n is a hyper-parameter to be tuned, in our experiment we test a range of $n \in [5, 10, 100, 1000]$.



(a) Considering the target node e_1 and assuming 5 relations types exist, $r_1, \ldots, r_5$, r_1 is in the range of e_1 and r_5 is in the domain of e_1. The feature then is $[1, 0, 0, 0, 0, 0, 0, 0, 0, 1]$.



(b) Assuming 5 relations types exist, $r_1, \ldots, r_5$ and considering the inverse relation r'_i, we sampled 2 length-3 paths from e_1, $p_1 = (e_1, r_1, e_2, r_2, e_3, r_5, e_2)$ and $p_2 = (e_1, r_1, e_4, r_2, e_5, r'_3, e_6)$. The feature is $[1, 1, 0, 0, 1, 0, 0, 1, 0, 0]$.



(c) e_2 is the target node. If 3 anchor nodes a_1, a_2, a_3 are selected, and 2-nearest anchors are a_1 and a_2, the path from a_1, a_2 to e_2 are $p_1 = (a_1, r_1, e_3, r_5, e_2)$ and $p_2 = (a_2, r_5, e_1, r_1, e_2)$. The constructured feature is $[1, 1, 0, 1, 0, 0, 0, 1]$

Figure 1: Examples of three feature construction methods

NodePiece Feature Representation Aside from the relation types, the entities that are relevant to the target node are also useful for constructing the node feature. However, since the number of entities in a knowledge graph is usually giant, it is computationally not feasible to construct a feature vector to identify all the entities. Thus, we refer to the idea in [Galkin *et al.*, 2021] and consider constructing the features based on a small subset of the graph entities, which are called anchor nodes.

Specifically, given a knowledge graph $\mathcal{G} = (\mathcal{E}, \mathcal{R})$, the task is to use fixed-size of anchor nodes and all the relation types to form a vocabulary set and represent all the entities. For instance, assuming that we successfully select $|A|$ nodes as anchors, each anchor node a_i has the shortest path p directing to the target node v, which records $|A|$ paths. We keep the k nearest anchors by measuring the distance between a_i's and v. Then the index of these anchor nodes and the relation types along the paths could be used to represent the target nodes. An example can be found in Figure 1c.

The anchor nodes can be selected either randomly or by importance measurement. For our purpose, centrality and Personalized PageRank (PPR) [Hamilton, 2020] on nodes would be combined to determine the anchor nodes. And we continue to use the convention in [Galkin *et al.*, 2021] by sampling 40% of the anchor nodes by centrality, 50% by Personalized PageRank and 10% by random sampling. Similar to the IRID feature construction, we used a binary identity function to construct the feature. For each entity node v, the feature is decomposed into 2 parts, the anchor node representation part $\vec{h}^a$, and the relation representation part. The first part could be expressed as,

$$\vec{h}_i^a = \begin{cases} 1 & \text{if } a_i \text{ is one of the } k\text{-nearest anchors} \\ 0 & \text{otherwise} \end{cases}$$

If node a_i is selected to be an anchor node and appear in the k-nearest anchors of node v, we then find k shortest paths between all selected a_i's and v, which is $p_i = (e_1, r_1, e_2, \ldots e_{K-1}, r_{K-1}, e_K)$. We again record the relation that appears in the path to form a relation feature.

$$\vec{h}_i^r = \begin{cases} 1 & \text{if relation } r \text{ appers in } p_i \\ 0 & \text{otherwise} \end{cases}$$

Then the feature vectors $\vec{h}_i^a$ and $\vec{h}_i^r$ are concatenated and finally forming the $\vec{h} \in \mathbb{R}^{|A|+|\mathcal{R}|}$. The feature matrix then is $\mathbf{H} \in \mathbb{R}^{|\mathcal{E}| \times (|A|+|\mathcal{R}|)}$. In the experiment, we test a range of the number of anchors $|A| \in \{200, 500, 1000\}$ and the number of neighborhoods $k \in \{5, 20, 50, 100\}$.

The auxiliary task design Currently, we only consider constructing features for entities, which can be denoted as $\mathbf{H} = \{\vec{h}_1, \ldots \vec{h}_i, \ldots, \vec{h}_{|\mathcal{E}|}\}^T \in \mathbb{R}^{|\mathcal{E}| \times F}$, where $|\mathcal{E}|$ is the number of entities and F is the size of the feature vectors. As all the features constructed are binary, the model design for the feature prediction tasks is very simple. We feed the entities embeddings to a dense layer $g(\cdot)$ with sigmoid activation to reconstruct the features and use binary cross-entropy loss to train the model. The overall loss now becomes:

$$\mathcal{L} = \sum_{s,r,o \in \mathcal{S}} \ell_{s,r,o} + \lambda L' \left(g \left(\mathbf{E} \right), \mathbf{H} \right)$$

3.4 Manifold Regularization

Manifold regularization was first used in matrix factorization [Cai *et al.*, 2010]. For tensor factorization models in neural link predictors, the intuition is that if two data entities e_i and e_j lies closely in the intrinsic geometry space, their embeddings $\mathbf{e}_i$ and $\mathbf{e}_j$ should also be close. If the similarity between embeddings is measured by the l_2 distance,

Dataset	Regularizers	CP			DistMult			ComplEx		
		MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10
FB15k-237	Baseline	34.83	25.77	52.88	35.84	26.53	54.78	36.67	27.28	55.78
	Norm-based	34.85	25.78	52.97	36.06	26.79	54.83	36.81	27.46	55.90
	GP	35.01	26.05	52.94	36.09	26.81	54.92	36.90	27.54	55.90
	IRID	35.00	25.97	53.03	36.14	26.86	54.91	36.79	27.36	55.80
	Path	35.01	26.03	53.01	36.00	26.71	54.90	36.75	27.27	55.74
	NodePiece	35.11	26.04	53.18	36.12	26.88	54.91	36.84	27.51	55.83
	Manifold				/	/	/	36.16	26.76	55.27
WN18RR	Baseline	11.40	7.43	19.25	45.07	41.02	53.11	**48.60**	**44.14**	**57.48**
	Norm-based	11.58	7.71	19.51	45.08	40.94	53.44	**48.70**	**44.54**	**56.77**
	GP	11.69	7.58	20.33	45.27	41.23	53.60	48.35	44.13	56.78
	IRID	11.89	7.84	20.35	**45.22**	**41.48**	53.20	48.71	44.73	57.08
	Path	11.52	7.53	19.82	45.19	41.18	53.53	48.60	44.50	57.07
	NodePiece	12.12	7.83	21.03	45.20	40.84	53.74	48.62	44.40	56.77

*IRID - In-range and in-domain feature; GP - Gradient Penalty, embedding size = 2000

Table 1: Experiments results

we can get the regularization as

$$\mathbf{R}_k = \sum_{i,j=1}^{N} \|\mathbf{e}_i - \mathbf{e}_j\|_2^2 \mathbf{W}_{ij}$$

The distance between the embeddings $\mathbf{e}_i$ and $\mathbf{e}_j$ is minimized according to the amplitude of a penalization weight $\mathbf{W}_{ij}$, which is determined by the similarity between entities e_i and e_j. By this definition we formalize the manifold regularization in neural link predictors, and the loss function now becomes

$$\mathcal{L} = \sum_{s,r,o \in S} \ell_{s,r,o} + \lambda \sum_{i=1}^{K} \mathbf{R}_k$$

Similarity Matrix Construction Manifold regularization needs to access a distance matrix $\mathbf{D}$ to retrieve the similar entities in the knowledge base. To construct a similarity matrix and calculate the weight W_{ij}, we would adopt the feature construction methods in Section 3.3, and use the feature vectors as a representation for entities. Specifically, The distance matrix is calculated by measuring the distance between feature vectors, where $D_{ij} = (\vec{h}_i - \vec{h}_j)^2$

Regularization Weights Determination The value of $\mathbf{W}_{ij}$ could be calculated according to 2 methods, respectively the k-Nearest Neighbors (KNN) and the Gaussian kernel.

The KNN weight construction only penalize the distance between a target node e_i and its neighbor nodes, where $\mathbf{W}_{ij} = 1$ if e_j is one of the k-neareest neighborhoods of e_i. The neighbors are found based on the distance matrix $\mathbf{D}$. In the Gaussian kernel method, the weight $\mathbf{W}_{ij}$ is calculated based on the distance between each entity in the feature matrix $\mathbf{H}$. $\mathbf{W}_{ij} = \exp(-(\vec{h}_i - \vec{h}_j)^2/\sigma)$.

4 Empirical Study

To verify the effectiveness of our proposed methods, the experiments are designed with the following settings:

Datasets Two benchmark datasets, FB15k-237 [Bollacker *et al.*, 2008] and WN18RR [Dettmers *et al.*, 2018] are selected in the paper.

Metrics We use Hits@k, $k \in \{1, 3, 10\}$ and filtered mean reciprocal rank (MRR)[Akrami *et al.*, 2020] as the evaluation metrics.

Models Experiments are conducted with models based on tensor factorization, including CP, DistMult and ComplEx. We used the nuclear N3 norm [Lacroix *et al.*, 2018] as a regularizer.

4.1 The impact of regularizers

The experiment results are shown in Table 1. For norm-based regularisers, we find nuclear norm dominates the performance. The best performance is observed when nuclear norm is applied solely and l_1, l_2 norms can only bring a small change.

Except for the norm-based regularisers, we compare the performance of training with and without the regularisers to answer how do the extra regularizers impact the KGC models. All the regularisers are trained together with nuclear norm. To find the best hyperparameter combinations, grid search was done with n3 $\in \{0, 0.001, 0.01, 0.05, 0.1, 0.5\}$, regulariser weight $\in \{0.1, 1, 10, 50, 100\}$.

We observe that our regularisers can only bring a marginal improvement to the models. Regularisers like multi-task learning with random paths representation and manifold regularisation cannot even outperform baseline models. The results are disappointing at the first glimpse. But as we further look into the cases when the models are trained with

Dataset	Regularizers	CP			DistMult			ComplEx		
		MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10
FB15k-237	w/o reg	33.43	24.86	50.34	34.80	25.90	52.63	34.79	25.74	52.91
	GP	**34.44**	25.53	**52.11**	**35.78**	26.54	**54.56**	**36.49**	**27.23**	**55.13**
	IRID	33.72	25.04	50.78	35.58	**26.61**	53.66	35.50	26.36	53.92
	NodePiece	34.40	**25.63**	51.57	35.29	26.37	53.37	35.37	26.30	53.77
WN18RR	w/o reg	8.39	6.06	12.98	44.32	41.30	50.29	45.51	42.47	51.08
	GP	10.10	7.31	15.00	44.34	41.35	50.77	**47.27**	**43.34**	**55.34**
	IRID	10.31	7.68	14.91	**44.48**	41.50	50.14	45.99	42.75	51.83
	NodePiece	**11.27**	**7.70**	**18.00**	44.46	**41.50**	**50.77**	45.99	42.66	52.44

* w/o reg - without regularizer; GP - Gradient Penalty; IRID - In-range and in-domain feature;

Table 2: Experiments results when the regularizers are applied individually

smaller N3 weights, they do benefit from some of our designed regularisers.

Experiments without nuclear norm To have a clear understanding of our regularisers, we further investigate in the scenarios when the regularisers are applied to the models individually without nuclear norm. The results are illustrated in Table 2. We did not further conduct the experiments on regularisers found ineffective in previous steps. The experiment results suggest that gradient penalty can bring a comparative generalisation improvement as N3 norm, and the model training also benefits from the multi-task learning regularisers with IRID features and NodePiece feature.

4.2 Data Efficiency Analysis

In the machine learning tasks, if the data points are not sufficient, the model would be easily overfitting to the data, and fewer data points usually mean more likely of overfitting. In that consideration, we thought it necessary to conduct a data efficiency experiment to test how feature prediction as an additional task influences the model training when data points are inadequate.

We test in the situations when 5%, 10%, 20% and 50% data points from the whole dataset are used for training by uniformly random sampling from the original dataset. And we construct the features and train the model respectively with the subset or with the whole dataset. Figure 2 gives an example when NodePiece feature is used.

For all the experiments with insufficient data, training with auxiliary tasks shows an improvement on the model generalisation, and using the feature constructed from the whole dataset brings even better improvement. This result proves that the auxiliary task would encode extra information while training the embeddings.

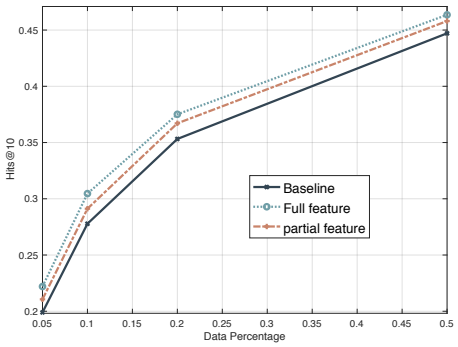


Figure 2: Data efficiency analysis for feature prediction evaluated on FB15k-237, with $k = 2000$ and n3 $= 0.1$

5 Conclusion

Our works suggests that the proposed regularizers cannot further boost the performance of the factorization-based models when nuclear-3 norm is applied. However, we find that gradient penalty regularization could bring a similar improvement to the models as nuclear norm. And multi-task learning regularizers would also benefit the training of neural link predictors, even though the boost is not as significant as other regularizers. As a future work, we do encourage researches who are interested in this topic to give an explanation about the mechanism behind this phenomenon.

References

[Akrami *et al.*, 2020] Farahnaz Akrami, Mohammed Samiul Saeef, Qingheng Zhang, Wei Hu, and Chengkai Li. Realistic re-evaluation of knowledge graph completion methods: An experimental study. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 1995–2010, 2020.

[Balazevic *et al.*, 2019] Ivana Balazevic, Carl Allen, and Timothy M. Hospedales. Tucker: Tensor factorization for knowledge graph completion. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 5184–5193. Association for Computational Linguistics, 2019.

[Bizer *et al.*, 2009] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. Dbpedia - A crystallization point for the web of data. *J. Web Semant.*, 7(3):154–165, 2009.

[Bollacker *et al.*, 2008] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250, 2008.

[Bordes *et al.*, 2013] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26, 2013.

[Cai *et al.*, 2010] Deng Cai, Xiaofei He, Jiawei Han, and Thomas S Huang. Graph regularized nonnegative matrix factorization for data representation. *IEEE transactions on pattern analysis and machine intelligence*, 33(8):1548–1560, 2010.

[Chen *et al.*, 2021] Yihong Chen, Pasquale Minervini, Sebastian Riedel, and Pontus Stenetorp. Relation prediction as an auxiliary training objective for improving multi-relational graph representations. In *3rd Conference on Automated Knowledge Base Construction*, 2021.

[Das *et al.*, 2020] Rajarshi Das, Ameya Godbole, Shehzaad Dhuliawala, Manzil Zaheer, and Andrew McCallum. A simple approach to case-based reasoning in knowledge bases. *arXiv preprint arXiv:2006.14198*, 2020.

[Dettmers *et al.*, 2018] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 1811–1818. AAAI Press, 2018.

[Dobrowolska *et al.*, 2021] Agnieszka Dobrowolska, antonio vergari, and Pasquale Minervini. Neural concept formation in knowledge graphs. In *3rd Conference on Automated Knowledge Base Construction*, 2021.

[Galkin *et al.*, 2021] Mikhail Galkin, Jiapeng Wu, Etienne Denis, and William L. Hamilton. Nodepiece: Compositional and parameter-efficient representations of large knowledge graphs. *CoRR*, abs/2106.12144, 2021.

[Hamilton, 2020] William L Hamilton. Graph representation learning. *Synthesis Lectures on Artifical Intelligence and Machine Learning*, 14(3):1–159, 2020.

[Hitchcock, 1927] Frank L. Hitchcock. The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics*, 6(1-4):164–189, 1927.

[Hoffman *et al.*, 2019] Judy Hoffman, Daniel A. Roberts, and Sho Yaida. Robust learning with jacobian regularization. *CoRR*, abs/1908.02729, 2019.

[Lacroix *et al.*, 2018] Timothée Lacroix, Nicolas Usunier, and Guillaume Obozinski. Canonical tensor decomposition for knowledge base completion. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 2869–2878. PMLR, 2018.

[Rifai *et al.*, 2011] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In Lise Getoor and Tobias Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 833–840. Omnipress, 2011.

[Suchanek *et al.*, 2007] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings*

of the 16th international conference on World Wide Web, pages 697–706, 2007.

[Thanh-Tung *et al.*, 2019] Hoang Thanh-Tung, Truyen Tran, and Svetha Venkatesh. Improving generalization and stability of generative adversarial networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[Trouillon *et al.*, 2016] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In Maria-Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 2071–2080. JMLR.org, 2016.

[Yang *et al.*, 2015] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[Yuan and Zhang, 2016] Ming Yuan and Cun-Hui Zhang. On tensor completion via nuclear norm minimization. *Found. Comput. Math.*, 16(4):1031–1068, 2016.

[Zou and Hastie, 2005] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.