A Proof of Theorem 1

Figure 1 shows the three filling modes of hypersphere in 2D. Figure 2(b) shows an example of hypersphere with compact mode in 3D, and the shaded area represents the gap. Here is the formal proof:

Proof. A hypersphere can be filled with several identical smaller hyperspheres by three filling modes: align, compact and hybrid.

We first give the proof of align mode (Figure 1(a)):

The volume of a n-dimensional hypersphere of radius R in n-dimensional Euclidean space is:

$$V_{hypersphere} = \frac{\pi^{\frac{n}{2}}}{\Gamma(\frac{n}{2}+1)} R^n, \tag{1}$$

where Γ is Leonhard Euler's gamma function. It satisfies $\Gamma(n) = (n-1)!$ if n is a positive integer and $\Gamma(n+\frac{1}{2}) = (n-\frac{1}{2}) \cdot (n-\frac{3}{2}) \cdot \ldots \cdot \frac{1}{2} \cdot \pi^{\frac{1}{2}}$ if n is a non-negative integer.

Similarly, the volume of a n-dimensional hypercube of edge length L is:

$$V_{hypercube} = L^n. \tag{2}$$

Then, we have the gap between each smaller hypersphere and its circumscribed cube:

$$\begin{aligned} V_{gap} &= V_{hypercube}(2r) - V_{hypersphere}(r) \\ &= (2r)^n - \frac{\pi^{\frac{n}{2}}}{\Gamma(\frac{n}{2}+1)} r^n \\ &= (2^n - (\pi^{\frac{1}{2}})^n) \cdot \frac{r^n}{\Gamma(\frac{n}{2}+1)} > 0, \end{aligned} \tag{3}$$

where r is the radius of each smaller hypersphere.

Let N_r be the number of smaller hyperspheres inside the given hypersphere, we have:

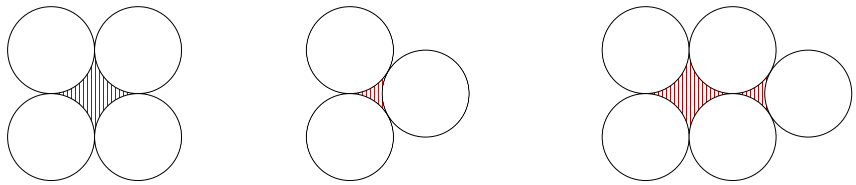
$$\begin{aligned} N_r &\geq \frac{V_{hypersphere}(R-2r)}{V_{hypercube}(2r)} \\ &= \frac{\pi^{\frac{n}{2}}}{\Gamma(\frac{n}{2}+1)} \cdot \frac{(R-2r)^n}{(2r)^n} > 0, \end{aligned} \tag{4}$$

$$V_{gaps} = V_{gap} \cdot N_r > 0. \tag{5}$$

Therefore, a hypersphere cannot be filled with several identical smaller hyperspheres without a single gap by align mode.

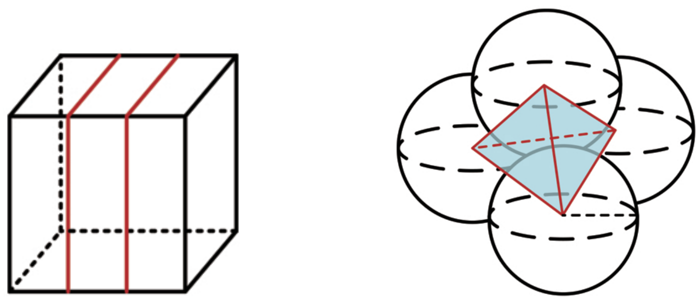
Similarly, this conclusion can be obtained in compact mode (Figure 1(b)).

Hybrid mode (Figure 1(c)) is the mixture of align mode and compact mode. As a result, hybrid mode also cannot fill in a hypersphere with several identical smaller hyperspheres without a single gap.



(a) Align. (b) Compact. (c) Hybrid.

Fig. 1. Three filling modes in 2D. Shaded areas among circles represent the gaps.



(a) Box: align mode. (b) Hypersphere: compact mode.

Fig. 2. Examples in 3D.

B Proof of Theorem 2

Figure 2(a) shows an example that a box can be exactly filled with 3 identical smaller boxes along horizontal axis. Here is the formal proof:

Proof. A box can be divided into k equal parts along any dimension by align mode, and each part is a smaller box. Given a box $b(\mathbf{cen}, \mathbf{off})$, we will recieve k identical smaller boxes by dividing b into k equal parts along the *n-th* dimension (n can be any dimension). The centers and the offsets of each smaller box are the same as b except in *n-th* dimension. The center and the offset of the *n-th* dimension are:

$$
\begin{aligned}
cen_{i,n} &= cen_n - off_n + \frac{off_n}{k} + \frac{2(i-1) \cdot off_n}{k} \\
&= cen_n + \frac{(2i-k-1) \cdot off_n}{k}, \\
off_{i,n} &= \frac{off_n}{k}.
\end{aligned}
$$

where $cen_{i,n}$ denotes the center in *n-th* dimension of the *i-th* smaller box, and $off_{i,n}$ denotes the offset in *n-th* dimension of the *i-th* smaller box.

C Proof of Theorem 3

Proof. According to the Theorem 1 and Theorem 2, the embedding space utilization of box is higher than hypersphere. From Definition 2, we know that the representation power of box (hypersphere) structure can be measured by the space utilization of the embedding space. Thus, the representation power of box is superior to hypersphere.

D Proof of Theorem 4

Relations in IBKE can be categorised as *isA* relations (i.e., `instanceOf` and `subclassOf`) and *non-isA* relations. Note that *isA* relations can only appear in **antisymmetric** and **composition** pattern. Here is the formal proof:

Proof. For *non-isA* relations, if $r(x, y)$ and $\neg r(y, x)$ hold, we have

$$\mathbf{x} + \mathbf{r} = \mathbf{y} \ \wedge \ \mathbf{y} + \mathbf{r} \neq \mathbf{x},$$

then, we have:

$$\|\mathbf{r}\| > \epsilon,$$

where ϵ is a positive number.

Therefore, IBKE can infer the **antisymmetric** pattern for *non-isA* relations.

If $r_1(y, x)$ and $r_2(x, y)$ hold, we have

$$\mathbf{y} + \mathbf{r}_1 = \mathbf{x} \ \wedge \ \mathbf{x} + \mathbf{r}_2 = \mathbf{y},$$

then, we have:

$$\mathbf{r}_1 = -\mathbf{r}_2.$$

Therefore, IBKE can infer the **inversion** pattern for *non-isA* relations.

If $r_2(x, y)$, $r_3(y, z)$ and $r_1(x, z)$ hold, we have:

$$\mathbf{x} + \mathbf{r}_2 = \mathbf{y} \ \wedge \ \mathbf{y} + \mathbf{r}_3 = \mathbf{z} \ \wedge \ \mathbf{x} + \mathbf{r}_1 = \mathbf{z},$$

then, we have:

$$\mathbf{r}_1 = \mathbf{r}_2 + \mathbf{r}_3.$$

Therefore, IBKE can infer the **composition** pattern for *non-isA* relations.

For *isA* relations, the proof for **antisymmetric** pattern is the same as *non-isA* relations. In particular, if a is an instance of b and b is a subclass of c, we can see that a is an instance of c, which is a composition pattern. If a is a subclass of b and b is a subclass of c, we can see that a is a subclass of c, which is also a composition pattern. Therefore, IBKE can infer the **composition** pattern for *isA* relations.

In conclusion, IBKE can infer the **antisymmetric**, **inversion** and **composition** pattern.

Algorithm 1 Random update strategy

Input: Instance vector $\mathbf{i}$, concept box $b(\mathbf{cen}, \mathbf{off})$ of (i, r_i, c), positive triple $\xi \in \mathcal{S}_i$, negative triple $\xi' \in \mathcal{S}_i'$, positive update threshold ϕ_{pos} and negative update threshold ϕ_{neg}.

Let $p = 0$.
for $\xi \in \mathcal{S}_i$ **do** //positive triples
    $p \leftarrow rand(0, 1)$.
    if $f_i > 0$ **then** //instance i outside the box b
        Update embeddings w.r.t. $\nabla\mathcal{L}_i$.
    else //instance i inside the box b
        while $p < \phi_{pos}$ **do**
            Update embeddings w.r.t. $\nabla\mathcal{L}_i$.
        end while
    end if
end for
for $\xi' \in \mathcal{S}_i'$ **do** //negative triples
    $p \leftarrow rand(0, 1)$.
    if $f_i < 0$ **then** //instance i inside the box b
        Update embeddings w.r.t. $\nabla\mathcal{L}_i$.
    else //instance i outside the box b
        while $p < \phi_{neg}$ **do**
            Update embeddings w.r.t. $\nabla\mathcal{L}_i$.
        end while
    end if
end for

E Negative Sampling

For each relational triple (h, r, t), we randomly replace h or t to construct a negative triple (h', r, t) or (h, r, t'). Specifically, we obtain h' or t' by randomly sampling from a *brother* set $\mathcal{M}_t = \mathcal{M}_1 \cup \mathcal{M}_2 \cup \ldots \cup \mathcal{M}_m$, where m is the number of concepts that t belongs to and $\mathcal{M}_m = \{a | a \in \mathcal{I} \wedge (a, r_i, c_m) \in \mathcal{S}_i \wedge (t, r_i, c_m) \in \mathcal{S}_i \wedge t \neq a\}$. For the other two kinds of triples, we follow the same policy to construct negative triple sets.

F Random Update Strategy

Algorithm 1 shows the details of Random update strategy.

G Countries Dataset

Countries contains 2 relations (*locatedIn*(e_1, e_2) and *neighborOf*(e_1, e_2)) and 271 entities (243 countries, 5 regions and 23 subregions). It has three sub-tasks: S1, S2 and S3. These sub-tasks increase difficulty in a step-wise fashion by increasing the minimum pathlength of solutions. Each task in Countries is to predict *locatedIn*(c, r), i.e., if a country locates in a region. In S1, the correct triples can be predicted from the following patterns:

$$\textit{locatedIn}(c, s) \wedge \textit{locatedIn}(s, r) \Rightarrow \textit{locatedIn}(c, r)$$

where s refers to the subregion of country c.

In S2, the correct triples can be predicted from the following patterns:

$$\textit{neighborOf}(c_1, c_2) \wedge \textit{locatedIn}(c_2, r) \Rightarrow \textit{locatedIn}(c_1, r)$$

In S3, the correct triples can be predicted from the following patterns:

$$\begin{aligned} &\textit{neighborOf}(c_1, c_2) \wedge \textit{locatedIn}(c_2, s) \wedge \\ &\quad \textit{locatedIn}(s, r) \Rightarrow \textit{locatedIn}(c_1, r) \end{aligned}$$

Note that S3 is the most difficult task, which requires a path of length 3.

The statistics of Countries are listed in Table 1.

Table 1. Details of Countries.

	Countries				
	#Train	#Test	#Valid	#Entity	#Relation
S1	1111	24	24	271	2
S2	1063	24	24	271	2
S3	985	24	24	271	2

Table 2. Configurations on YAGO39K and Countries.

	k	γ_r	γ_i	γ_c	λ	ϕ_{pos}	ϕ_{neg}
YAGO39K	100/200	1	0.1	1	0.001	0.2	0.7
Countries	500	0.1	0.1	0.1	0.001	0.2	0.7

H YAGO39K Dataset

The statistics of YAGO39K are listed in Table 3.

I Optimal Configurations

The optimal configurations are listed in Table 2. For fairness, we train IBKE for 1000 epochs with $k = 100, 200$. IBKE on Countries is trained with $k = 500$ for 2000 epochs to get the best results.

Table 3. Details of YAGO39K. #Ins, #Con and #Rel denote the number of instances, concepts and relations, respectively.

				#InstanceOf			#SubclassOf			#Relational		
DataSet	#Ins	#Con	#Rel	Train	Test	Valid	Train	Test	Valid	Train	Test	Valid
YAGO39K	39374	46110	39	442,836	5,000	5,000	30,181	1,000	1,000	354,997	9,364	9,341

Table 4. Computational complexity of several embedding models. n and m denote the quantity of entities and relations, respectively.

Model	Space Complexity	Time Complexity
TransE	$\mathcal{O}(nk + mk)$	$\mathcal{O}(k)$
TransH	$\mathcal{O}(nk + mk)$	$\mathcal{O}(k)$
TransR	$\mathcal{O}(nk_e + mk_ek_r)$	$\mathcal{O}(k_ek_r)$
TransD	$\mathcal{O}(nk_e + mk_r)$	$\mathcal{O}(\max(k_e, k_r))$
HolE	$\mathcal{O}(nk + mk)$	$\mathcal{O}(k \log k)$
DistMult	$\mathcal{O}(nk + mk)$	$\mathcal{O}(k)$
ComplEx	$\mathcal{O}(nk + mk)$	$\mathcal{O}(k)$
SimplE	$\mathcal{O}(nk + mk)$	$\mathcal{O}(k)$
TorusE	$\mathcal{O}(nk + mk)$	$\mathcal{O}(k)$
RotatE	$\mathcal{O}(nk + mk)$	$\mathcal{O}(k)$
QuatE	$\mathcal{O}(nk + mk)$	$\mathcal{O}(k)$
TransC	$\mathcal{O}(nk + mk)$	$\mathcal{O}(k)$
IBKE	$\mathcal{O}(nk + mk)$	$\mathcal{O}(k)$

J Computational Complexity

We summarize the space complexity and time complexity of several representitive methods in Table 4, in which we can see that the using box structure instead of hypersphere does not significantly increase the computational complexity.