バイナリCP分解モデルのビット反転学習法

† 林 克彦 [a,*]　岸本 広輝 [b,*]　新保 仁 [c]
[a] 東京大学 大学院情報理工学系研究科
[b] 大阪大学 産業科学研究所
[c] 千葉工業大学 ステアラボ

†khayashi0201@gmail.com

1 はじめに

知識グラフ埋め込み (KGE) は知識獲得，推薦，質問応答などを応用先とし，NLP や AI 分野で盛んに研究されている．KGE では一般にエンティティと関係を実数ベクトルとして表現するが，YAGO や Freebase といった大規模な知識グラフには数百万から数十億単位のエンティティが含まれるため，埋め込みモデルの軽量化が重要な課題となる．

岸本ら [5] は KGE に対する CP テンソル分解モデルのパラメータを全てバイナリ化するバイナリ CP 分解モデル (**B-CP**) を提案し，タスク性能を保ったまま，従来の KGE モデルより数十倍コンパクトな埋め込みを実現した．さらに，彼らは B-CP が知識グラフをモデル化するのに十分な表現力を持つことも理論的に証明している [4]．表 1 に B-CP モデルの概要，及び，他の KGE モデルとの比較をまとめる．

B-CP の学習は **Hinton 法 (HSTE)** [1] に基づいており，勾配学習の中で量子化関数によって強制的に実数ベクトルを離散化してバイナリベクトルを構築する．そのため，学習では数百から千数百次元の実数ベクトルを保持することになり，学習時のメモリ効率は悪い．また，実数ベクトルを強制的に離散化するため，学習収束の観点からも効率的な学習法とは言えない．

本稿では B-CP の学習に使う目的関数 (ここではロジスティック損失関数) をバイナリベクトルのまま直接最適化する新しい学習法を提案する (**ビット反転学習法**)．提案法ではバイナリベクトルのある次元パラメータ (ビット) を反転した場合の目的関数の変化を計算し，1 ビットずつ目的関数が改善するように山登り法で学習を行う．提案法は単純であり，

表 1: KGE モデルの比較: 表現力は任意の三次隣接テンソル $\mathscr{X} \in \{0,1\}^{N_e \times N_e \times N_r}$ を再現可能な次元数．

モデル	計算量	スコア関数	表現力
TransE 等	$O(D)-O(KD)$	$-\|\|p(\mathbf{a}_i)+\mathbf{b}_k-p(\mathbf{a}_j)\|\|$	✗
RESCAL	$O(D^2)$	$\mathbf{a}_i^{\mathsf{T}}\mathbf{B}\mathbf{a}_j$	N_e
DistMult	$O(D)$	$\langle \mathbf{a}_i, \mathbf{a}_j, \mathbf{b}_k \rangle$	✗
CP	$O(D)$	$\langle \mathbf{a}_i, \mathbf{b}_j, \mathbf{c}_k \rangle$	$N_e N_r$
B-CP	$O(D)$	$-h(\mathbf{a}_i, \mathrm{XNOR}(\mathbf{b}_j, \mathbf{c}_k))$	$8N_eN_r$

- 実装が容易 (著者の github に公開予定)
- ハイパーパラメータが少ない

という利点を持つ．さらに，ベンチマークデータセットを用いた実験から，良好な結果が得られたのでここに報告する．

知識グラフ以外にも，単語やネットワークの埋め込みをバイナリ化する手法は研究されているが [6]，提案法はそれらへの応用も十分に期待される．

2 準備

ベクトル，行列，テンソルをそれぞれ太字かつ小文字 (**a**)，太字かつ大文字 (**A**)，カリグラフィック体 ($\mathscr{X}$) で表す．行列 **A** の i 番目の行ベクトルを $\mathbf{a}_{i:}$，j 番目の列ベクトルを $\mathbf{a}_{:j}$ または $\mathbf{a}_j$ と表す．ベクトル，行列，テンソルの各要素を a_i, a_{ij}, x_{ijk} と表す．

記号 $\circ$ はアダマール積，$\otimes$ は外積を表す．3 つの (行または列) ベクトル $\mathbf{a},\mathbf{b},\mathbf{c} \in \mathbb{R}^D$ が与えられたとき，三重内積を $\langle \mathbf{a},\mathbf{b},\mathbf{c} \rangle = \sum_{d\in[D]} a_d b_d c_d$ として定義する．$[D]$ は $1,2,\cdots D$ の自然数の集合を表す．

三次テンソル $\mathscr{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ が 3 つのベクトルの外積 $\mathscr{X} = \mathbf{a}\otimes\mathbf{b}\otimes\mathbf{c}$ で表せるとき，これを**ランク 1 テンソル**と呼ぶ．

*これらの著者は本稿に等しく貢献した．

3 バイナリ CP 分解モデル (B-CP)

知識グラフは主語・目的語のエンティティe_i, e_j，及び，それらの間の関係r_kから成る事実(e_i, e_j, r_k)の集合である．集合に存在する事実を 1，存在しない事実を 0 として，知識グラフは三次隣接テンソル$\mathscr{X} \in \{0,1\}^{N_e \times N_e \times N_r}$で表せる．ここで$N_e$はエンティティ数，$N_r$は関係数を表す．

CP 分解はテンソルをD個のランク 1 テンソルの線形和によって分解する．知識グラフを表した三次隣接テンソル$\mathscr{X}$に対する CP 分解は

$$\mathscr{X} \approx \sum_{d \in [D]} \mathbf{a}_d \otimes \mathbf{b}_d \otimes \mathbf{c}_d$$

として表せる．B-CP では普通の CP 分解とは異なり，$\mathbf{a}_d \in \{-1,+1\}^{N_e}, \mathbf{b}_d \in \{-1,+1\}^{N_e}, \mathbf{c}_d \in \{-1,+1\}^{N_r}$ として，実数ベクトルの代わりにバイナリベクトルを用いる．行列$\mathbf{A} = [\mathbf{a}_1 \mathbf{a}_2 \cdots \mathbf{a}_D]$は因子行列とし，$\mathbf{B}, \mathbf{C}$も同様に定義する．$\mathscr{X}$の要素$x_{ijk}$は$\langle \mathbf{a}_{i:}, \mathbf{b}_{j:}, \mathbf{c}_{k:} \rangle$で近似される．$\mathbf{a}_{i:}, \mathbf{b}_{j:}, \mathbf{c}_{k:}$はそれぞれ主語，目的語，関係に対する$D$ビットのバイナリベクトルである．

本稿では知識グラフに対する B-CP の学習を二値分類問題として考え，ロジスティック回帰による定式化を行う．まず，各x_{ijk}が因子行列$\mathbf{A}, \mathbf{B}, \mathbf{C}$から独立に生起すると仮定し，

$$p(\mathscr{X} | \mathbf{A}, \mathbf{B}, \mathbf{C}) = \prod_{i \in [N_e]} \prod_{j \in [N_e]} \prod_{k \in [N_r]} p(x_{ijk} | \theta_{ijk})$$

を考える．ある事実(e_i, e_j, r_k)に対するスコアは$\theta_{ijk} = \langle \mathbf{a}_{i:}, \mathbf{b}_{j:}, \mathbf{c}_{k:} \rangle$として定義される．また，確率変数$x_{ijk}$が$x_{ijk} \sim \mathrm{Bernoulli}(\sigma(\theta_{ijk}))$と仮定すると，事後確率は以下のように定義される．

$$p(x_{ijk} | \theta_{ijk}) = \begin{cases} \sigma(\theta_{ijk}) & \text{if } x_{ijk} = 1, \\ 1 - \sigma(\theta_{ijk}) & \text{if } x_{ijk} = 0 \end{cases}.$$

シグモイド関数$\sigma(x) = 1/(1+\exp(-x))$とする．

ここでは最大事後確率の負の対数尤度

$$\underset{\mathbf{A},\mathbf{B},\mathbf{C}}{\mathrm{argmin}} \Big\{ E(\mathbf{A}, \mathbf{B}, \mathbf{C}) = \sum_{i \in [N_e]} \sum_{j \in [N_e]} \sum_{k \in [N_r]} E_{ijk} \Big\}.$$

$$E_{ijk} = x_{ijk} \log(\sigma(\theta_{ijk})) + (x_{ijk} - 1) \log(1 - \sigma(\theta_{ijk}))$$

を目的関数とし，それを最小化することで学習を行う．

Algorithm 1 Greedy Bit-flip Training Algorithm

```
Random Initialize A,B ∈ {−1,+1}^{N_e×D} and C ∈ {−1,+1}^{N_r×D}
DL = {1,...,D}
for iter ∈ {1,...,I} do
    Negative Sampling // 負例を生成
    Update(C, N_r, DL) // 関係の因子行列の更新
    Update(A, N_e, DL) // 主語の因子行列の更新
    Update(B, N_e, DL) // 目的語の因子行列の更新
    Flush Negative Samples // 負例をクリア
    Convergence Check // 収束判定
end for
```

Algorithm 2 Update(**M**, N, DL)

```
Random Shuffle DL
// N に対して非同期な並列処理が可能
for i ∈ {1,...,N} do
    for j ∈ DL do
        if Δ(m_ij) < 0 then
            m_ij = −m_ij
        end if
    end for
end for
```

4 B-CP のビット反転学習法

提案法では B-CP の因子行列$\mathbf{A}, \mathbf{B}, \mathbf{C}$における 1 つの要素 (ビット) を反転させたとき，目的関数が向上すれば，それを反転させるという作業を繰り返して学習を進める．アルゴリズム全体の流れは Algorithm 1，及び，2 に記述した通りである．

Algorithm 1 ではある因子行列を更新する際，残りの 2 つの因子行列は固定する．知識グラフにおいて，関係の数N_rはエンティティの数N_eよりかなり少ないため，1 つの関係に対するパラメータの更新が目的関数に与える影響は大きい．よって，ここでは関係の因子行列$\mathbf{C}$を先に更新することで収束を早めるように工夫している．

Algorithm 2 では因子行列の更新を実際に行う．注意として，因子行列の行要素 (関係やエンティティ) は独立に処理できるため，Algorithm 2 は行要素に対して非同期な並列処理が可能である．一方で，列要素 (次元数) に対しては更新順序が他の列要素の更新に影響を与えるため，次元の集合DLをシャッフルすることでその影響を軽減する．

Algorithm 2 の更新では，因子行列の 1 つの要素を反転させたときの目的関数の変化を計算する必要がある．例えば，因子行列$\mathbf{A}$の要素a_{ij}を$-a_{ij}$に反転したときの因子行列を$\mathbf{A}'$とし，そのときの目的関数の差分$\Delta(a_{ij}) = E(\mathbf{A}', \mathbf{B}, \mathbf{C}) - E(\mathbf{A}, \mathbf{B}, \mathbf{C})$を考える．$\Delta(a_{ij})$

は以下の等式で表される．

$$\Delta(a_{ij}) = \sum_{y\in[N_e]}\sum_{z\in[N_r]}\left(x_{iyz}\log\left(\frac{\sigma(\theta_{iyz})}{\sigma(\theta_{iyz}-2a_{ij}b_{yj}c_{zj})}\right)\right.$$
$$\left.+(1-x_{iyz})\log\left(\frac{1-\sigma(\theta_{iyz})}{1-\sigma(\theta_{iyz}-2a_{ij}b_{yj}c_{zj})}\right)\right).$$

この変化が目的関数を向上させる場合，$a_{ij} = -a_{ij}$ としてビット反転して更新を行う[1]．因子行列 $\mathbf{B},\mathbf{C}$ についても同様である．提案法では目的関数は必ず向上するように更新が行われるが，パラメータの更新順序に依存するため，局所解に陥る可能性がある．学習は目的関数の向上が見られなくなるか，事前に定めれたエポック回数が終わると終了する．

以下では，学習を高速化するための実装上の主要な工夫について説明する．

4.1 ビット演算によるスコア計算の高速化

Algorithm 2 ではスコア θ_{ijk} の計算が繰り返し行われる．学習を高速に行うには，スコア計算の高速化が必須である．対処法の 1 つとして，全ての事実のスコアをキャッシュしておく事も考えられるが，事実の数は非常に多いため，メモリ効率の観点から，ここではビット演算によるスコア計算の高速化を考える．

実際，B-CP のスコア計算はビット演算を使って計算することができる．

$$\theta_{ijk} = D - 2h(\mathbf{a}_{i:}, \mathrm{XNOR}(\mathbf{b}_{j:}, \mathbf{c}_{k:})).$$

ここで $h(\cdot,\cdot)$ はハミング距離，$\mathrm{XNOR}(\cdot,\cdot)$ は排他的論理和の否定を表す．文献 [5] でも報告されているように，D が数百次元の場合，ビット演算によるスコア計算は実数ベクトルの三重内積よりも約 50 倍高速に計算できるため，スコア計算のコストはほとんど無視できるものとなる．

4.2 負例サンプリング

目的関数の計算において，全ての負例を考慮することは現実的ではない．そこで従来法の慣習に従い，負例サンプリングによって目的関数を近似する．文献 [3] で提案されたように，B-CP の訓練では正例となる事実 (e_i, e_j, r_k) が存在するとき，逆関係の事実 (e_j, e_i, r_k^{-1}) も正例として追加する．ここで負例サンプリングは，

[1]更新に関連する事実へのアクセスは単純な Trie 構造を利用したが，より効率の良いデータ構造を考えることもできる．

表 2: 知識グラフ補完精度の評価用データセット．

	WN18RR	FB15k-237
N_e	40,559	14,505
N_r	11	237
訓練サンプル数	86,835	272,115
開発サンプル数	3,034	17,535
テストサンプル数	3,134	20,466

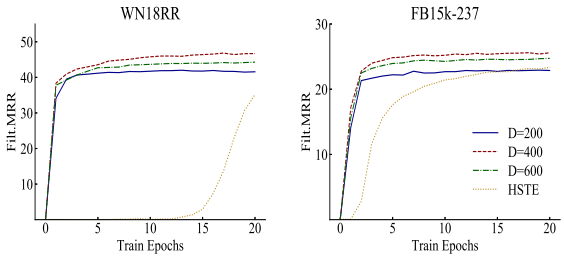

図 1: WN18RR，及び，FB15k-237 でのエポック数 vs モデル精度: MRR は開発データ上で計測した．

(e_i, e_j, r_k) に対して，ランダムサンプリングしたエンティティ e によって，負例 (e_i, e, r_k) を生成する．また，逆関係を持つ正例 (e_j, e_i, r_k^{-1}) には，ランダムサンプリングしたエンティティ e によって，負例 (e, e_i, r_k^{-1}) を生成する．1 つの正例当たり，これらを事前に定められた負例サンプル数分だけ行う．

5 実験

5.1 データセット・評価方法・実験設定

KGE モデルの精度を評価するため，標準の知識グラフ補完データセット WN18RR，及び，FB15k-237 を用いた (表 2)．これらは他のデータセットよりも補完が難しいことで知られている．

テストデータ中の事実 (e_i, e_j, r_k) における e_i (または e_j) を e_ℓ ($\ell \in [N_e]$) に置き換え，スコアで降順にソートして，元の事実の順位を平均逆順位 (MRR) と Hits@1, 3, 10 で評価した．複数の正解事実が存在する場合，対象としている事実以外を除いた (Filterd 設定)．

実験では次元数 $D \in \{200, 400, 600\}$ を調べ，MRR を開発データ上で最大化するハイパーパラメータを求めた．$\mathbf{A},\mathbf{B},\mathbf{C}$ は標準正規分布に従う bool 型の擬似乱数によって初期化し，負例サンプルは 1 つの正例当たり，WN18RR で 5 個，FB15k-237 で 20 個生成した．

表 3: WN18RR，及び，FB15k-237 での知識グラフ補完精度の比較: *の結果は文献 [2] から抜粋.

	WN18RR				FB15k-237			
	MRR	Hits@			MRR	Hits@		
Models		1	3	10		1	3	10
DistMult*	43.0	39.0	44.0	49.0	24.1	15.5	26.3	41.9
ComplEx*	44.0	41.0	46.0	51.0	24.7	15.8	27.5	42.8
R-GCN*	–	–	–	–	24.8	15.3	25.8	41.7
ConvE*	43.0	40.0	44.0	52.0	**32.5**	**23.7**	**35.6**	**50.1**
HSTE	46.0	44.0	47.0	52.0	29.5	21.0	32.4	48.3
B-CP (400)	47.7 ±0.2	44.7 ±0.3	48.9 ±0.2	53.3 ±0.1	27.6 ±0.0	19.5 ±0.1	30.1 ±0.1	43.7 ±0.1
B-CP (400×2)	48.8	45.7	50.0	54.5	28.7	20.5	31.4	45.1
B-CP (400×3)	49.0	46.0	50.1	54.7	29.0	20.7	32.0	45.6
B-CP (400×4)	**49.1**	**46.1**	**50.3**	54.9	29.3	21.0	32.1	45.9
B-CP (400×5)	49.1	46.0	50.2	**55.0**	29.5	21.2	32.3	46.0

最大エポック数は 20 とした．時間の計測は 2.7GHz Intel Core i7 の CPU を積んだラップトップ PC で行った．WN18RR に対する 400 次元のモデルは 8 並列処理で 20 エポックを約 7 分で終了した．

5.2 主な実験結果

図 1 には訓練のエポック毎の MRR 精度を示した．提案法は訓練の早い段階で学習が収束しているが，これはパラメータの冗長性が少ないためだと考えられる．また，両データにおいて，次元数 400 で最良の結果が得られたので，テストデータ上での評価には 400 次元のモデルを用いた．

表 3 ではテストデータ上でのモデル精度を比較した．提案法の初期値依存性を調べるため，パラメータ初期化に用いるランダムシードを変えて，モデルを 5 つ構築した．結果はそれらの精度の平均値と標準偏差を示しており，提案法は初期値に大きく依存することなく，従来法と遜色ない精度が得られた．注意として，提案法はバイナリベクトルのため，従来モデルと同次元数であれば，大まかに 1/32 のメモリ使用量となる．

提案法が軽量かつスコア計算が高速である利点を活かし，5 つのモデルをアンサンブルした．表 3 の結果より，複数のモデルを用いることでモデル性能が向上することを確認した．

5.3 クエリ検索速度

KGE の応用先の 1 つに確率データベース検索がある．例えば，次のような質問文

$$Q(x):\ \exists x.(e_{i1},x,r_{k1})\wedge(e_{i2},x,r_{k2})$$

に対して，変数 x に当てはまるエンティティをランキングしたいことはよくある．ここでは FB15k-237 のテストデータから目的語が共通する事実 2 つを選び，上の形式の質問文を 553,774 文取り出した．そして，x に対する最良の目的語を探索した際の探索時間を計算した．

結果は 800 次元のバイナリベクトルで総時間 141 秒となった (1 クエリ当たり 0.0002 秒)．実数ベクトルではそのおよそ 200 倍となった．また，目的語をそれぞれの事実単独で予測した際の Filtered なしの MRR 精度の平均は 81.1 であったのに対し，同時に予測した際の精度は 84.6 で向上が見られた (表 3 よりも精度が高いのは，一般に目的語の方が当てやすいからである．)．これらの結果より B-CP は確率データベース検索でも有効に活用できる可能性があると言える．

6 まとめ

本稿では KGE に対する B-CP モデルをバイナリパラメータのまま学習する新たな手法を提案し，実験を通して，その有効性を示した．

現在，提案法の拡張を進めるべく，単語バイナリ埋め込みモデル Word2Bits [6] への拡張を検討している．

参考文献

[1] Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR*, Vol. abs/1308.3432, , 2013.

[2] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In *Proc. of the 32nd AAAI Conference on Artificial Intelligence (AAAI)*, pp. 1811–1818, 2018.

[3] Seyed Mehran Kazemi and David Poole. Simple embedding for link prediction in knowledge graphs. In *Proc. of the 32nd Annual Conference on Neural Information Processing Systems (NeurIPS)*, pp. 4289–4300, 2018.

[4] Koki Kishimoto, Katsuhiko Hayashi, Genki Akai, and Masashi Shimbo. Binarized canonical polyadic decomposition for knowledge graph completion. *arXiv preprint arXiv:1912.02686*, 2019.

[5] Koki Kishimoto, Katsuhiko Hayashi, Genki Akai, Masashi Shimbo, and Kazunori Komatani. Binarized knowledge graph embeddings. In *European Conference on Information Retrieval (ECIR)*, pp. 181–196. Springer, 2019.

[6] Maximilian Lam. Word2bits - quantized word vectors. *CoRR*, Vol. abs/1803.05651, , 2018.