algorithm/KGCN/model.py

from typing import List

import tensorflow as tf

from Recommender\_System.algorithm.KGCN.layer import SumAggregator, ConcatAggregator, NeighborAggregator

from Recommender\_System.utility.decorator import logger

@logger('初始化KGCN模型：', ('n\_user', 'n\_entity', 'n\_relation', 'neighbor\_size', 'iter\_size', 'dim', 'l2', 'aggregator'))

def KGCN\_model(n\_user: int, n\_entity: int, n\_relation: int, adj\_entity: List[List[int]], adj\_relation: List[List[int]],

neighbor\_size: int, iter\_size=2, dim=16, l2=1e-7, aggregator='sum') -> tf.keras.Model:

assert neighbor\_size == len(adj\_entity[0]) == len(adj\_relation[0])

l2 = tf.keras.regularizers.l2(l2)

user\_id = tf.keras.Input(shape=(), name='user\_id', dtype=tf.int32)

item\_id = tf.keras.Input(shape=(), name='item\_id', dtype=tf.int32)

user\_embedding = tf.keras.layers.Embedding(n\_user, dim, embeddings\_initializer='glorot\_uniform', embeddings\_regularizer=l2)

entity\_embedding = tf.keras.layers.Embedding(n\_entity, dim, embeddings\_initializer='glorot\_uniform', embeddings\_regularizer=l2)

relation\_embedding = tf.keras.layers.Embedding(n\_relation, dim, embeddings\_initializer='glorot\_uniform', embeddings\_regularizer=l2)

u = user\_embedding(user\_id)

flatten = tf.keras.layers.Flatten()

entities = [tf.expand\_dims(item\_id, axis=1)] # [(batch, 1), (batch, n\_neighbor), (batch, n\_neighbor^2), ..., (batch, n\_neighbor^n\_iter)]

relations = [] # [(batch, n\_neighbor), (batch, n\_neighbor^2), ..., (batch, n\_neighbor^n\_iter)]

for \_ in range(iter\_size):

neighbor\_entities = flatten(tf.gather(adj\_entity, entities[-1]))

neighbor\_relations = flatten(tf.gather(adj\_relation, entities[-1]))

entities.append(neighbor\_entities)

relations.append(neighbor\_relations)

if aggregator == 'sum':

aggregator\_class = SumAggregator

elif aggregator == 'concat':

aggregator\_class = ConcatAggregator

elif aggregator == 'neighbor':

aggregator\_class = NeighborAggregator

else:

raise Exception("Unknown aggregator: " + aggregator)

entity\_vectors = [entity\_embedding(entity) for entity in entities] # [(batch, 1, dim), (batch, n\_neighbor, dim), (batch, n\_neighbor^2, dim), ..., (batch, n\_neighbor^n\_iter, dim)]

relation\_vectors = [relation\_embedding(relation) for relation in relations] # [(batch, n\_neighbor, dim), (batch, n\_neighbor^2, dim), ..., (batch, n\_neighbor^n\_iter, dim)]

for it in range(iter\_size):

aggregator = aggregator\_class(activation='relu' if it < iter\_size - 1 else 'tanh', kernel\_regularizer=l2)

entities\_next = []

for hop in range(iter\_size - it):

inputs = (entity\_vectors[hop], entity\_vectors[hop + 1], relation\_vectors[hop], u)

vector = aggregator(inputs, neighbor\_size=neighbor\_size)

entities\_next.append(vector)

entity\_vectors = entities\_next

assert len(entity\_vectors) == 1

i = tf.reshape(entity\_vectors[0], shape=(-1, dim)) # batch, dim

score = tf.sigmoid(tf.reduce\_sum(u \* i, axis=1))

return tf.keras.Model(inputs=[user\_id, item\_id], outputs=score)

if \_\_name\_\_ == '\_\_main\_\_':

adj = [[1, 2], [0, 2], [0, 1]]

model = KGCN\_model(3, 3, 3, adj, adj)