# API

import Computational\_advertising

## Macthing

# deepwalk, node2vec

from Computational\_advertising import DeepWalk

from Computational\_advertising import Node2Vec

### deepwalk

import Computational\_advertising

from Computational\_advertising import DeepWalk

import networkx as nx

# read data

G = nx.read\_edgelist('/Users/zhangkai/Desktop/learning/Computational\_advertising/Computational\_advertising/matching/data/wiki/Wiki\_edgelist.txt',create\_using=nx.DiGraph(),nodetype=None,data=[('weight',int)])

# model train

model = DeepWalk(G,walk\_length=10,num\_walks=80,is\_directed=1, workers=1)

model.train(window\_size=5,iter=3)

embeddings = model.get\_embeddings()

### node2vec

import Computational\_advertising

from Computational\_advertising import Node2Vec

import networkx as nx

# read data

G = nx.read\_edgelist('/Users/zhangkai/Desktop/learning/Computational\_advertising/Computational\_advertising/matching/data/wiki/Wiki\_edgelist.txt',create\_using=nx.DiGraph(),nodetype=None,data=[('weight',int)])

# model train

model = Node2Vec(G,walk\_length=10,num\_walks=80,is\_directed=1, workers=1)

model.train(window\_size=5,iter=3)

embeddings = model.get\_embeddings()

### line

import Computational\_advertising

from Computational\_advertising import LINE

import networkx as nx

# read data

G = nx.read\_edgelist('/Users/zhangkai/Desktop/learning/Computational\_advertising/Computational\_advertising/matching/data/wiki/Wiki\_edgelist.txt',create\_using=nx.DiGraph(),nodetype=None,data=[('weight',int)])

# model train

model = LINE(G,embedding\_size=8,negative\_ratio=5,learning\_rate = 0.025, num\_batches=100, order="second-order")

model.train(batch\_size=1024)

embeddings = model.get\_embeddings()

## Ranking

### fm

Input embedings, batch\_size \* filed\_size \* embedding\_dim

### deepfm

def deepfm(embeddings, field\_size, embedding\_size, deep\_layers=[128, 64, 32], dropout\_keep\_deep=[0.5, 0.5, 0.5, 0.5], deep\_layers\_activation=tf.nn.relu, use\_fm=1, use\_deep=1):

"""

embeddings: embeddings should be batch \* filed \* embedding\_dim

field\_size:

embedding\_size:

deep\_layers: how many layers has in deep part

dropout\_keep\_deep: dropout in every layers, this dim just one large than deep\_layers, for input dropout

deep\_layer\_activation

use\_deep

use\_fm

"""

### dcn

def dcn(embeddings, cross\_layer\_num, field\_size):

# input: embeddings should be batch \* filed \* embedding\_dim, cross\_layer\_num is number of corss times

# output: cross\_network\_out

# conside if we can put every output of dcn to dnn, and dnn to dcn

## Strategy

# Joint

### ESMM

Input: ctr\_logits, , cvr\_logits, ctr\_label, cvr\_label

Return: loss

def ESMM(ctr\_logits, cvr\_logits, ctr\_label, cvr\_label):  
 """  
 #param ctr\_logits:you can self define ctr model to get ctr\_logits  
 #param cvr\_logits:you can self define cvr model to get cvr\_logits  
 #param ctr\_label:  
 #param cvr\_label:  
 #param params:  
 """