Faiss

向量索引

　Faiss是Facebook AI团队开源的针对聚类和相似性搜索库，为稠密向量提供高效相似度搜索和聚类，支持十亿级别向量的搜索，是目前最为成熟的近似近邻搜索库。它包含多种搜索任意大小向量集（备注：向量集大小由RAM内存决定）的算法，以及用于算法评估和参数调整的支持代码。

下面是一个实例，有一个坑在命名文件的时候不要以faiss命名。因为有import faiss会造成混乱。可以按照这个实例根据自己的需要进行修改

import numpy as np

import faiss

import time

def IndexIVFFlat():

d = 768 # dimension

nb = 4000 # database size

#np.random.seed(1234) # make reproducible

training\_vectors= np.random.random((nb, d)).astype('float32') #构建4000个维度为768的向量

faiss.normalize\_L2(training\_vectors)

nlist = 1000 # 聚类中心的个数

k = 50 #邻居个数

quantizer = faiss.IndexFlatIP(d) # the other index，需要以其他index作为基础

time1 = time.time()

index = faiss.IndexIVFFlat(quantizer, d, nlist, faiss.METRIC\_INNER\_PRODUCT) #精准的L2搜索

time2 = time.time()

# by default it performs inner-product search

assert not index.is\_trained

index.train(training\_vectors)

assert index.is\_trained

index.nprobe = 300 # default nprobe is 1, try a few more

time3 = time.time()

index.add(training\_vectors) # add may be a bit slower as well

time4 = time.time()

print(time2-time1)

print(time3-time2)

print(time4-time3)

for i in range(1000):

vectors= np.random.random((1, d)).astype('float32')

t1=time.time()

D, I = index.search(vectors, 100) # actual search # 从向量training\_vectors中查找与vectors最相近的前100个向量

t2 = time.time()

print('faiss kmeans result times {}'.format(t2-t1))

print(t2-t1)

print(D[:5]) # neighbors of the 5 first queries

print(I)

排序loss

def ranking\_loss(labels, logits):

lm = tf.constant(1.0)

m\_plus = tf.constant(2.5)

m\_minus = tf.constant(1.0)

L = tf.constant(0.0)

i = tf.constant(0)

cond = lambda i, l: tf.less(i, config.batch\_size)

def d\_loss(logits):

\_, cminus\_indices = tf.nn.top\_k(logits, k=1)

sminus = logits[cminus\_indices[0]]

l = tf.log((1.0+tf.exp((lm\*(m\_minus+sminus)))))

return l

def u\_loss(logits, cplus):

cplus = tf.cond(tf.equal(cplus, 1), lambda: 0, lambda: 1)

sminus = tf.cond(tf.equal(cplus, 0),

lambda: logits[1], lambda: logits[0])

splus = logits[cplus]

l = tf.log((1.0+tf.exp((lm\*(m\_plus-splus))))) + \

tf.log((1.0+tf.exp((lm\*(m\_minus+sminus)))))

return l

def loop\_body(i, L):

cplus = labels[i]

l = tf.cond(tf.equal(cplus, 0), lambda:d\_loss(logits[i]), lambda:u\_loss(logits[i], cplus))

return [tf.add(i, 1), tf.add(L, l)]

\_, L = tf.while\_loop(cond, loop\_body, loop\_vars=[i, L])

nbatch = tf.to\_float(config.batch\_size)

L = L/nbatch

return L

参考

<http://yongyuan.name/blog/vector-ann-search.html>

<https://www.jianshu.com/p/944be496905f>

<https://blog.csdn.net/hihei_set/article/details/89679061>