2020年12月21日 9:14

# 图卷积网络的半监督分类——代码和思想分析

Semi-Supervised Classification with Graph Convolutional Networks arXiv:1609.02907 代码链接: https://github.com/tkipf/gcn

分析以Cora数据集为例,cora、citeseer以及 pubmed数据集的预处理见 https://github.com/kimiyoung/planetoid

(文章直接使用的经讨预处理的数据集)

· 基于案例Case Based

- 遗传算法Genetic\_Algorithms神经网络Neural\_Networks

- 概率方法Probabilistic\_Methods强化学习Reinforcement\_Learning
- · 规则学习Rule\_Learning

Cora数据集: Cora数据集包含2708个科学出版物,分为七个类别。 引文网络由5429个链接组成。数据集中的每个出版物都用0/1值的词向量描述,该词向量指示字典中是否存在相应的词。该词典包含1433个独特的单词。含cora.content和cora.cites两个文件。

语料库共**2708**条数据(论文): 每篇论文引用 至少一篇论文或者被至少其他一篇论文引用。

数据集

★ Cora.content文件(论文描述):〈paper\_id〉〈word\_attributes〉〈class\_label〉

2708 24043 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Neural Networks

以最后一条数据为例,

Al 本表 Navar/2077, paper\_id=24043,是该篇论文的唯一字符串标识 word\_attributes,共1433个字段(二进制**0/1)**,表明该单词是否在论文中存在 class\_label=Neural\_Networks,论文的类别标签(七个之一)

★ Cora.cites文件(引用图): 〈ID of cited paper〉 〈ID of citing paper〉

<被引文编号> <引文编号> 1 35 1033 2 35 103482 103515 35 35 1050679

8 35 1112911

1103960 以<35 1033>为例,即"论文1033->论文35", 35 1103985 表明idx为1033号论文引用了idx为35号论文。 1109199

第一个条目是被引用论文的标识,第二个标识代表包含引用的论文。链接的方向 是从右向左。由此可通过论文之间的索引关系建立邻接矩阵adj

\_init\_\_ data // 图数据 - inits — layers metrics

// 初始化的一些公用函数 // GCN层的定义 // 评估指标的计算

// 模型结构定义 models // 训练 — train // 工具函数的定义 - utils

图数据:分别是对citeseer、cora、pubmed三个数据集处理后的训练数据、验证数据、测试数据。

初始化公用函数:

- uniform(shape,scale=0.05,name=None) # 产生一个维度为shape的张
- 量、值分布在(-0.005-0.005)之间,且为均匀分布 · glorot(shape, name=None) # glorot方法保证前向传播(激活值)和反向传播(梯度值)的方差一致,是一个通过该层的输入和输出参数个数得到的 分布范围内的均匀分布。

训练数据: 140, 范围 [0.140]

验证数据: 500, 范围 [140,640] 测试数据: 1000, 范围 [1708, 2707]

节点组成的邻接矩阵、特征矩阵

输出数据: 预测标签

layers

代码结构

图卷积层的定义

基类Layer: 为所有层对象定义了最基本的API, 如

- \_\_init\_\_() 初始化部分参数name定义了层的变量范围
- logging用于切换开启/关闭Tensorflow直方图记录
- \_call\_\_()调用\_call()函数,让layer层的实例可调用

**继承类Dense**: 定义了dense层基本的参数和操作,如

- 激活函数、稀疏数据、特征矩阵、偏置项等
- 执行inits里的glorot函数
- · 重写\_call()函数,如果输入数据是稀疏的,进行稀疏dropout

**继承类GraphConvolution**: 定义了图卷积层的参数和操作,和dense不同之处在于:

• 增加了support (邻接矩阵经过处理的形式),减小了张量的op操作和存储空间

models

模型结构的定义

基类Model: 根据Layer来建立的基类Model模型

- \_\_init\_\_() 初始化必要的变量,如layers,activations,
- loss、accuracy、optimizer等 Build()用于建立序列层模型,activations用于存储输入和输出

**继承类MLP**: 多层感知机模型

- 采取了动量梯度下降方法的优化器等
- 对于损失值计算了I2权重衰减(防止过拟合)和交叉熵误差
- 采用的是双层Dense全连接网络
- 对最终输出进行softmax归一化

继承类GCN: 图卷积模型,和MLP不同之处在于:

• 神经网络层采用的是Layers中定义的GraphConvolution图卷积层

train

对模型进行训练

- flags=tf.app.flags # 定义了支持接受命令行传递参数的全局变量,包含数据 模型、学习率、训练时期数、dropout率、权重衰减、早期停止时间等。
- 输入维度input\_dim=features[2][1](1433),也就是每个节点特征的维度
- 训练模型
  - o 对(features, support, y\_train, train\_mask, placeholders)进行训练
  - 对(features, support, y\_val, val\_mask, placeholders)进行验证
  - 对(features, support, y\_test, test\_mask, placeholders)进行测试
     计算损失值loss和精确度accuracy, 打印测试结果

metrics

定义了两个评估指标函数

 masked\_softmax\_cross\_entropy(preds,labels,mask) # mask相当于帶标签的索引向量,和loss相乘后只对帶有标签的样本计算损失,得到的是一维张 量。最后返回平均损失值。

代码

• masked accuracy(preds,labels,mask) # 同上, 计算得到的是平均精度

utils 定义了一些功能函数

读取数据: load\_data(dataset\_str):

- 载入数据的维度(共2708个节点);
  - o 邻接矩阵(2708, 2708)
  - 特征矩阵(2708, 1433)
  - labels: ally, ty数据集叠加构成, labels.shape:(2708, 7)

  - train\_mask, val\_mask, test\_mask: (2708, )train\_mask中的[0,140)范围的是True, 其余是False: val\_masks是[140, 640); test\_mask是[1708,2707]

  - y\_train, y\_val, y\_test: (2708, 7)
     y\_train的値为对应与labels中train\_mask为True的行, 其余全是0: 另外两个也类似。

特征矩阵预处理: preprocess\_features(features):

· 特征矩阵进行·行归一化·并返回一个格式为(coords, values, shape)的元组

邻接矩阵预处理: preprocess\_adj(adj):

将邻接矩阵加上自环以后,对称归一化,并存储为C00模式,最后返回元组格式

构建输入字典: construct\_feed\_dict(features, support, labels, labels\_mask, placeholders):

- 传入具体的值并返回,49126是特征矩阵存储为coo模式后非零元素的个数(2078\*1433里只有49126个非零,稀疏度达1.3%)
- 由于邻接矩阵是稀疏的,并且用LIL格式表示,因此定义为一个tf.sparse\_placeholder(tf.float32),可以节省内存 计算切比雪夫近似矩阵: chebyshev\_polynomials(adj, k):
- 返回一个稀疏矩阵列表 (元组表示)

读取数据

V-1/2/201	
.x	训练实例的特征向量, x.shape:(140, 1433)
.tx	测试实例的特征向量, y. shape: (140, 7)
.allx	带标签和未带标签的训练实例, allx.shape:(1708, 1433)
.y	带标签的训练实例的独热标签, y. shape: (140, 7)
.ty	测试实例的独热标签, ty.shape:(1708, 1433)
.ally	将ind.dataset_str.allx中的实例标签作为numpy.ndarray对象,ally.shape:(1708,7)
.graph	格式为{index: [index_of_neighbor_nodes]}的dicts作为collections.defaultdict对象
.test.index	图中测试实例的索引,用于归纳设置为列表对象。

以上所有对象必须使用python pickle模块保存 返回:加载所有数据输入文件(以及训练/测试数据)。

核心思想是什么

通过标记过的节点之间的引用关系图,来预 测未被标记的节点分类。

总结分析

输入了什么 经过了什么变换 输出了什么

输入了什么

经过了什么变换

输出了什么

训练阶段: 经过处理后的节点特征矩阵、邻接矩阵、经过mask后带标签的训练实例、训练实例对应的mask张量(都是经过 placeholder的数据,可减小运算操作和存储空间,下同) 验证阶段:

经过处理后的节点特征矩阵、邻接矩阵、经过mask后带标签的验证实例、验证实例对应的mask张量

经过处理后的节点特征矩阵、邻接矩阵、经过mask后带标 签的测试实例、测试实例对应的mask张量

两层图卷积 卷积操作被定义为:

 $Z = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} X \Theta$ 

卷积的最终输出是对节点的预测标签 程序的最终输出是预测的损失值和精度

## 问题总结

- > 训练集是在每个类别上挑选20个样本得到的。本文使用拥有500个样本的验证集来优化参数,比如dropout比率、L2正则项权重和学习率。训练好模型后,对于每个数
- 新级,本文在1000个随机,接选的测试样本上评估预测的正确率。

  ▶ 作者进行了1-10层的图卷积实验对比发现: 2~3层图卷积的效果最好,这是因为over smoothing,随着层数的增加,远处的节点和近处的节点相似而更难以区分,效果 反而降低。
- 及问时以。 》为什么都有标签叫做半监督呢?半监督体现在文中的哪里?查询了一些资料发现,作者在计算损失函数时只考虑有标签的样本,先用有标签的计算损失求参数,然后 再正向传播一次得到无标签节点的标签。半监督是因为用了数据集的所有特征加上了训练集的所有标签来预测的测试集的损失值。