

# GCN

作者源码: <https://github.com/tkipf/gcn>

相关博客:

1. [GRAPH CONVOLUTIONAL NETWORKS](#)
2. [GCN代码分析](#)
3. [GCN+AE代码逐行学习](#)
4. [图卷积网络GCN代码分析](#)
5. [如何理解 Graph Convolutional Network \(GCN\)](#)
6. [GCN和GCN在文本分类中应用](#)

```
└── data      // 图数据
└── inits     // 初始化的一些公用函数
└── layers    // GCN层的定义
└── metrics   // 评测指标的计算
└── models    // 模型结构定义
└── train     // 训练
└── utils     // 工具函数的定义
```

1. def parse\_index\_file(filename) # 处理index文件并返回index矩阵
2. def sample\_mask(idx, l) # 创建 mask 并返回mask矩阵
3. def load\_data(dataset\_str) # 读取数据

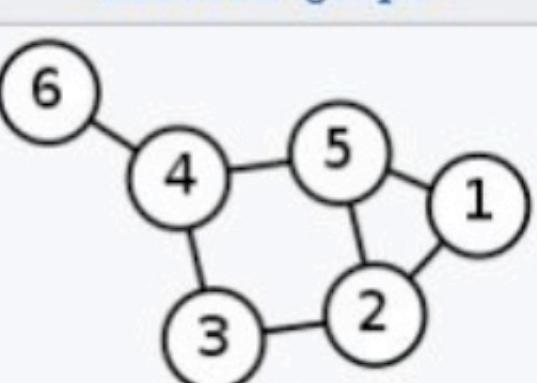
从gcn/data文件夹下读取数据，文件包括有：

- ind.dataset\_str.x => 训练实例的特征向量，如scipy.sparse.csr.csr\_matrix类的实例
- ind.dataset\_str.tx => 测试实例的特征向量，如scipy.sparse.csr.csr\_matrix类的实例
- ind.dataset\_str.allx => 有标签的+无无标签训练实例的特征向量，是ind.dataset\_str.x的超集
- ind.dataset\_str.y => 训练实例的标签，独热编码，numpy.ndarray类的实例
- ind.dataset\_str.ty => 测试实例的标签，独热编码，numpy.ndarray类的实例
- ind.dataset\_str.ally => 有标签的+无无标签训练实例的标签，独热编码，numpy.ndarray类的实例
- ind.dataset\_str.graph => 图数据，collections.defaultdict类的实例，格式为 {index: [index\_of\_neighbor\_nodes]}
- ind.dataset\_str.test.index => 测试实例的id

## 1. 统计邻接矩阵

Graph Fourier Transformation及Graph Convolution的定义都用到图的拉普拉斯矩阵，那么首先来介绍一下拉普拉斯矩阵。

对于图  $G = (V, E)$ ，其Laplacian 矩阵的定义为  $L = D - A$ ，其中  $L$  是Laplacian 矩阵， $D$  是顶点的度矩阵（对角矩阵），对角线上元素依次为各个顶点的度， $A$  是图的邻接矩阵。看图5的示例，就能很快知道Laplacian 矩阵的计算方法。

Labeled graph	Degree matrix	Adjacency matrix	Laplacian matrix
	$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 2 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & -1 & 0 & -1 & 3 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix}$

The screenshot shows a browser window with multiple tabs open, including CSDN, blog.csdn.net, and several YouTube and video player tabs. The main content area displays a Python code snippet for converting index lists into a sparse matrix using NumPy and SciPy. The code uses arrays for row, column indices, and data values, and then creates a CSR matrix. A terminal window below the browser shows the execution of this code, resulting in a sparse matrix with non-zero elements at specific indices.

```
>>> import numpy as np
>>> from scipy.sparse import csr_matrix
>>> csr_matrix((3, 4), dtype=np.int8).toarray()
array([[0, 0, 0, 0],
       [0, 0, 0, 0],
       [0, 0, 0, 0]], dtype=int8)

>>> row = np.array([0, 0, 1, 2, 2])
>>> col = np.array([0, 2, 2, 0, 1, 2])
>>> data = np.array([1, 2, 3, 4, 5, 6])
>>> csr_matrix((data, (row, col)), shape=(3, 3)).toarray()
array([[1, 0, 4],
       [0, 0, 5],
       [2, 3, 6]])
```

## 邻接矩阵统计

index list 转 matrix

## 稀疏矩阵压缩

```
>>> import numpy as np
>>> from scipy.sparse import csr_matrix
>>> csr_matrix((3, 4), dtype=np.int8).toarray()
array([[0, 0, 0, 0],
       [0, 0, 0, 0],
       [0, 0, 0, 0]], dtype=int8)

>>> row = np.array([0, 0, 1, 2, 2])
>>> col = np.array([0, 2, 2, 0, 1, 2])
>>> data = np.array([1, 2, 3, 4, 5, 6])
>>> csr_matrix((data, (row, col)), shape=(3, 3)).toarray()
array([[1, 0, 2],
       [0, 0, 3],
       [4, 5, 6]])
```

## 模型选择

```
97     # Some preprocessing 实际是做归一化
98     features = preprocess_features(features)
99     if FLAGS.model == 'gcn':
100         support = [preprocess_adj(adj)]
101         num_supports = 1
102         model_func = GCN
103     elif FLAGS.model == 'gcn_cheby':
104         support = chebyshev_polynomials(adj, FLAGS.max_degree)
105         num_supports = 1 + FLAGS.max_degree
106         model_func = GCN
107     elif FLAGS.model == 'dense':
108         support = [preprocess_adj(adj)] # Not used
109         num_supports = 1
110         model_func = MLP
111     else:
112         raise ValueError('Invalid argument for model: ' + str(FLAGS.model))
```

GCN定义在model.py文件中，model.py 定义了一个model基类，以及两个继承自model类的MLP、GCN类。

```
1      from layers import *
2      from metrics import *
3      import tensorflow as tf
4
5      flags = tf.app.flags
6      FLAGS = flags.FLAGS
7
8
9      class Model(object):
10
11
12      class MLP(Model):
13
14      class GCN(Model):
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
```

## GCN类的定义

```

148     def _loss(self):
149         # Weight decay loss
150         for var in self.layers[0].vars.values():
151             self.loss += FLAGS.weight_decay * tf.nn.l2_loss(var)
152
153         # Cross entropy error
154         self.loss += masked_softmax_cross_entropy(self.outputs, self.placeholders['labels'],
155                                                 self.placeholders['labels_mask'],
156                                                 self.multi_label)
157
158     def _accuracy(self):
159         self.accuracy = masked_accuracy(self.outputs, self.placeholders['labels'],
160                                         self.placeholders['labels_mask'],
161                                         self.multi_label)
162
163         if not self.multi_label:
164             self.pred = tf.argmax(self.outputs, 1)
165             self.labels = tf.argmax(self.placeholders['labels'], 1)
166         else:
167             self.pred = tf.where(tf.nn.sigmoid(self.outputs) >= 0.5, tf.ones(tf.shape(self.outputs)), tf.zeros(tf.shape(self.outputs)))
168             self.pred = tf.cast(self.pred, tf.int32)
169             self.labels = self.placeholders['labels']
170
171     def _build(self):
172         # 两个卷积层
173         self.layers.append(GraphConvolution(input_dim=self.input_dim,
174                                              output_dim=FLAGS.hidden1,
175                                              placeholders=self.placeholders,
176                                              act=tf.nn.relu,
177                                              dropout=True,
178                                              featureless=True,
179                                              sparse_inputs=True,
180                                              logging=self.logging))
181
182         self.layers.append(GraphConvolution(input_dim=FLAGS.hidden1,
183                                              output_dim=self.output_dim,
184                                              placeholders=self.placeholders,
185                                              act=lambda x: x,#,
186                                              dropout=True,
187                                              logging=self.logging))
188
189     def predict(self):
190         if not self.multi_label:
191             return tf.nn.softmax(self.outputs)
192         else:
193             return tf.nn.sigmoid(self.outputs)

```

## 卷积操作

$$Z = \tilde{D}^{-1/2} \tilde{A}^{-1/2} X \theta$$

```

136     # 从 Layer 继承下来得到图卷积网络
137     # 与denseNet的唯一差别是_call函数和__init__函数 (self.support = placeholders['support'] 的初始化)
138     class GraphConvolution(Layer):
139         """Graph convolution layer."""
140
141         def __init__(self, input_dim, output_dim, placeholders, dropout=0.,
142                      sparse_inputs=False, act=tf.nn.relu, bias=False,
143                      featureless=False, **kwargs):
144
145             self.input_dim = input_dim
146             self.output_dim = output_dim
147             self.placeholders = placeholders
148             self.dropout = dropout
149             self.sparse_inputs = sparse_inputs
150             self.act = act
151             self.bias = bias
152             self.featureless = featureless
153             self.kwargs = kwargs
154
155             self.vars = {}
156             self.supports = []
157             self.embedding = None
158
159             if not self.featureless:
160                 self.vars['weights_0'] = glorot_uniform_variable("weights_0",
161                                                               [input_dim, output_dim])
162             else:
163                 self.vars['weights_0'] = glorot_uniform_variable("weights_0",
164                                                               [1, output_dim])
165
166             if self.bias:
167                 self.vars['bias'] = variable("bias", [output_dim])
168
169             self._call(inputs)
170
171     def _call(self, inputs):
172         x = inputs
173
174         # dropout
175         if self.sparse_inputs:
176             x = sparse_dropout(x, 1 - self.dropout, self.num_features_nonzero)
177         else:
178             x = tf.nn.dropout(x, 1 - self.dropout)
179
180         # convolve
181         # convolve 卷积的实现。主要是根据论文中公式  $Z = \tilde{D}^{-1/2} \tilde{A}^{-1/2} X \theta$  实现
182         # convolve
183         supports = list()
184         for i in range(len(self.support)):
185             if not self.featureless:
186                 pre_sup = dot(x, self.vars['weights_' + str(i)],
187                               sparse=self.sparse_inputs)
188             else:
189                 pre_sup = self.vars['weights_' + str(i)]
190                 support = dot(self.support[i], pre_sup, sparse=True)
191                 supports.append(support)
192         output = tf.add_n(supports)
193
194         # bias
195         if self.bias:
196             output += self.vars['bias']
197         self.embedding = output # output
198         return self.act(output)

```

## LIL (Row-Based Linked List Format) - 基于行的链表格式

稀疏矩阵转化成两个链表data和rows:

列表.data: data[k]是行k中的非零元素的列表。如果该行中的所有元素都为0，则它包含一个空列表。

列表.rows: 是在位置k包含了在行k中的非零元素列索引列表。

```

import numpy as np
import scipy.sparse as sp

A=np.array([[1,0,2,0],[0,0,0,0],[3,0,0,0],[1,0,0,4]])
AS=sp.lil_matrix(A)

print(AS.data)
# [list([1, 2]) list([]) list([3]) list([1, 4])]
print(AS.rows)
# [list([0, 2]) list([]) list([0]) list([0, 3])]

```

## 输入数据

载入数据的维度 (以Cora数据集为例)

- adj(邻接矩阵): 由于比较稀疏, 邻接矩阵格式是LIL的, 并且shape为(2708, 2708)
- features (特征矩阵) : 每个节点的特征向量也是稀疏的, 也用LIL格式存储, features.shape: (2708, 1433)
- labels: ally, ty数据集叠加构成, labels.shape:(2708, 7)
- train\_mask, val\_mask, test\_mask: shape都为(2708, )的向量, 但是train\_mask中的[0,140)范围的是True, 其余是False; val\_mask中范围为(140, 640]范围为True, 其余的是False; test\_mask中范围为[1708,2707]范围是True, 其余的是False
- y\_train, y\_val, y\_test: shape都是(2708, 7)。y\_train的值为对应与labels中train\_mask为True的行, 其余全是0; y\_val的值为对应与labels中val\_mask为True的行, 其余全是0; y\_test的值为对应与labels中test\_mask为True的行, 其余全是0
- 特征矩阵进行归一化并返回一个格式为(coords, values, shape)的元组

- 将邻接矩阵加上自环以后，对称归一化，并存储为COO模式，最后返回格式为(coords, values, shape)的元组

## 由于是非对称的,所以需要一个矩阵的操作

```

[0, 0, 0, 0]], dtype=int32)
>>>
>>> row = np.array([0, 1, 2])
>>> col = np.array([0, 1, 2, 3, 4, 5])
>>> data = np.array([1, 2, 3, 4, 5, 6])
>>> adj = csc_matrix((data, (row, col)), shape=(3, 3))
<3x3 sparse matrix of type '<class 'numpy.int32'>'>
with 6 stored elements in Compressed Sparse Column format>
>>> a = adj + adj.T.multiply(adj.T > adj) - adj.multiply(adj.T > adj)
>>> a.toarray()
array([[1, 0, 4],
       [0, 0, 5],
       [2, 3, 6]])
>>> adj.toarray()
array([[1, 0, 4],
       [0, 0, 5],
       [2, 3, 6]])
>>>

```

data是存储的数据，矩阵向量Vector的角度来看，[1,2,3,5]，在运算时都要

row, col data这里的行，0列存储了2；2行

```

>>> indptr = np.array([0, 1, 2, 3, 4, 5, 6])
>>> indices = np.array([0, 1, 2, 3, 4, 5, 6])

```

需要加自连接 单位矩阵 ,对角线加一 ,在归一化里面进行

```
from scipy.sparse import csr_matrix
```

接下来所有操作都是为了算这个PMI的TF IDF值

十分钟后理解Scip x | 庚澄廉 Harlem x | 知 GCN的概念与应 x | New Tab x | GCN和GCN在 x | 开课吧管理后台 x | Launch 15-2G x | +

cnblogs.com/Kalafinaian/p/11440732.html

Apps Google 常用 Practice Algorithm PT\_Job 吉他谱 PHD JOB 考试 LeetCode

数,  $|voc|$ 表示词汇总量, 对于特征矩阵 $X$ , 我们采用单位矩阵 $I$ 表示, 即每个节点的向量都是one-hot形式表示, 下面我们将介绍如何定义邻接矩阵 $A$ , 其公式如所示, 对于文档节点和词汇节点的权重, 我们采用TF-IDF表示, 对于词汇节点之间的权重, 我们采用互信息表示(PMI, point-wise mutual information), 在实验中, PMI表现好于两个词汇的共现词汇数, 其公式如所示:

$$A_{ij} = \begin{cases} \text{PMI}(i, j) & i \text{ 和 } j \text{ 是词语而且 } \text{PMI}(i, j) > 0 \\ \text{TF-IDF}(i, j) & i \text{ 是文档 } j \text{ 是词语} \\ 1 & i = j \\ 0 & \text{其他} \end{cases}$$

$$\text{PMI}(i, j) = \log \frac{p(i, j)}{p(i)p(j)}$$

$$p(i, j) = \frac{\#W(i, j)}{\#W}$$

$$p(i) = \frac{\#W(i)}{\#W}$$

其中 $\#W(i)$ 表示在固定滑动窗口下词汇*i*出现的数量,  $\#W(i, j)$ 表示在固定滑动窗口下词汇*i, j*同时出现的数量, 当 $\text{PMI}(i, j)$ 为正数表示词汇*i*和词汇*j*有较强的语义关联性, 当 $\text{PMI}(i, j)$ 为负数的时候表示词汇*i, j*语义关联性较低, 在构建完图后, 我们代入GCN中, 构建两层GCN, 如下:

10 mod 23, 而不是 $6 \times (3-7) \bmod 23$ , 博主好像少减了个数值。... --hesetone

4. Re:ECC椭圆曲线详解(有具体实例) 谢谢博主的文章, 看了好多篇, 您的最好了 --wkun123444

5. Re:ECC椭圆曲线详解(有具体实例) 博主, ECC的  $n$ , 也就是  $P$  点的阶怎么求的, 比如您说的比特币使用的, 那个  $n$  那么大是一步一步算的吗? 还有就是 Bob 在选择随机数  $r$  的时候, 他怎么知道有没有小于  $P$  点的阶  $n$ ? --wkun123444

8:44 2019/12/28

## 契比雪夫

十分钟后理解Scip x | 庚澄廉 Harlem x | 知 GCN的概念与应 x | New Tab x | GCN和GCN在 x | 开课吧管理后台 x | Launch 15-2G x | Mask矩阵 x | +

zhuanlan.zhihu.com/p/72546603

Apps Google 常用 Practice Algorithm PT\_Job 吉他谱 PHD JOB 考试 LeetCode

**知乎** 首发于 NLP杂货铺 关注专栏 写文章 ...

数。

已赞同 25 分享

**拉普拉斯矩阵:**

在图中, 定义 $D$ 是 $N \times N$ 的度数矩阵:

$$D(i, j) = \begin{cases} d_i, & \text{if } i = j \\ 0, & \text{otherwise} \end{cases}$$

$$d_i = \sum_j A_{ij}$$

$A$ 是 $N \times N$ 的邻接矩阵:

$$A(i, j) = \begin{cases} 1, & \text{if } x_i \sim x_j \\ 0, & \text{otherwise} \end{cases}$$

拉普拉斯矩阵定义为:

## 基于PaddleHub的疫情期间网民情绪识别

baseline: <https://aistudio.baidu.com/aistudio/projectdetail/294224>

Q1 思考900k未标签数据的使用

Q2 图片数据的使用

Q3 用户关系的使用

Q4 时间特征的使用