

به نام خدا



دانشکده مهندسی صنایع و سیستم ها
گروه فناوری اطلاعات

نام و نام خانوادگی: محمدرضا صبی پور

شماره دانشجویی: ۴۰۳۶۶۲۴۱۰۰۳

تمرین ۲ - پیش بینی پیوند

درس:

تحلیل شبکه های اطلاعات

استاد:

دکتر بابک تیمورپور

۱۴۰۳/۱۲/۱۶

فهرست

۳.....	شرح تمرین.....
۳.....	مقدمه.....
۴.....	محاسبه معیار پیش بینی پیوند.....
۵.....	درخت تصمیم.....
۶.....	مقدار خطا
۷.....	تحلیل پیوندهای اشتباه
۸.....	تاثیر نرمال سازی

شرح تمرین

هدف از این تمرین ابتدا محاسبه معیارهای پیش‌بینی پیوند بر روی گراف داده کلپ زاخاری می‌باشد. سپس با استفاده از یک الگوریتم دسته‌بندی (ترجیحا درخت تصمیم) و با توجه به معیار محاسبه شده عمل پیش‌بینی پیوند را انجام می‌دهیم. در آخر با بررسی دقت مدل به بررسی و تحلیل خروجی می‌پردازیم و بررسی می‌کنیم که کدام پیوندها اشتباه دسته‌بندی شده و علت آن چیست.

مقدمه

دیتای کلپ کاراته زاخاری شامل دو بخش می‌باشد: مربی و مدیریت. به دلیل وجود اختلاف نظر بین این دو نفر، باشگاه به دو بخش تقسیم شده است. اعضای جدید که عضو این باشگاه می‌شوند باید یکی از این دو نفر را انتخاب کنند این داده شامل ۳۵ راس و ۷۶ یال می‌باشد. همانطور که معلوم است تمام یال‌های ممکن در گراف موجود نمی‌باشند. برای استفاده از این دیتاست از کتابخانه igraph استفاده شده است. قطعه کد زیر این دیتاست را از مجموعه داده‌های کتابخانه igraph می‌خواند.

```
import igraph as ig
import math
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.preprocessing import StandardScaler, MinMaxScaler

g = ig.Graph.Famous('Zachary')
e = g.get_edgelist()
v_count = g.vcount()
```

تعداد گره‌های گراف و لیست یال‌ها به ترتیب در متغیرهای v و e ذخیره شده‌اند.

محاسبه معیار پیش‌بینی پیوند

برای این تمرین سه معیار پیش‌بینی پیوند را در نظر گرفته‌ایم.

- Jaccard-coefficient
- Adamic-adar
- Preferential-attachment

هر یک از این معیارها را به صورت دستی در پایتون و به کمک توابع `igraph` محاسبه شده‌اند. نحوه محاسبه این مقادیر با توجه به اسلایدهای درس انجام شده است و سپس با توابع داخلی `igraph` شباهت داده شده‌اند. نتیجه نهایی محاسبات دستی با محاسبات `igraph` تطابق کامل داشته است. قطعه کد زیر محاسبات بالا را انجام می‌دهد:

```
jaccard = np.zeros([v_count, v_count])
preferential_attachment = np.zeros([v_count, v_count])
adamic_adar = np.zeros([v_count, v_count])

for i in range(v_count):
    v = set(g.neighbors(i, mode='all'))
    for j in range(i+1, v_count):
        u = set(g.neighbors(j, mode='all'))
        jaccard[i,j] = len(u.intersection(v))/len(u.union(v))
        preferential_attachment[i,j] = len(u) * len(v)
        common_neighbors = v.intersection(u)
        score = sum(1/np.log(g.degree(w)) for w in common_neighbors if g.degree(w) > 1)
        adamic_adar[i,j] = score
```

در حین انجام محاسبات متوجه تفاوت بین مقادیر بدست آمده برای معیار `jaccard` و مقادیر محاسبه شده توسط توابع `igraph` شدیم. با بررسی مراحل محاسبه متوجه شدیم که تابع داخلی `igraph` که مقدار `jaccard` را محاسبه می‌کند از فرمولی متفاوت نسبت به فرمول موجود در اسلایدهای درس استفاده می‌کند. در `igraph` گره مورد بررسی نیز، خود به عنوان همسایه خود در نظر گرفته می‌شود. این در حالی است که در اسلایدهای درس گره مورد بررسی به عنوان همسایه خود در نظر گرفته نمی‌شود و فقط

گره های متصل به آن به عنوان همسایه لحاظ می شوند. به همین علت مقدار محاسبه شده برای معیار jaccard با مقداری که خود igraph بدست می آورد تفاوت جزئی دارد که ناشی از علت فوق الذکر است. ما در این تمرین گره های همسایه را فقط آن گره هایی که توسط یال به گره مذکور متصل هستند لحاظ کرده ایم.

درخت تصمیم

در این مرحله با استفاده از کتابخانه sklearn یک مدل درخت تصمیم ساخته و آن را با استفاده از ۸۰ درصد داده موجود تمرین داده و سپس بر روی باقی داده تست می کنیم. در ابتدا می بایست دیتای تمرین و تست را از هم جدا کنیم. قطعه کد زیر عمل جداسازی دیتا را انجام می دهد.

```
X = np.column_stack([
    jaccard[np.triu_indices(v_count, k=1)],
    preferential_attachment[np.triu_indices(v_count, k=1)],
    adamic_adar[np.triu_indices(v_count, k=1)]
])

Y = np.array([(1 if (i, j) in e or (j, i) in e else 0) for i in range(v_count) for j in range(i+1,
v_count)])

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, random_state=4, test_size=0.2)
```

در کد بالا متغیر X مقادیر محاسبه شده برای پیش بینی پیوند را در خود نگه می دارد که به آن بردار ویژگی نیز گویند. متغیر Y هم شامل برچسب یال ها می باشد. در صورتی که یال در داده موجود باشد آن اندیس با عدد ۱ پر شده و در غیر این صورت عدد صفر ذخیره می شود.

در ادامه یک مدل درخت تصمیم ساخته و با استفاده از دیتا آن را تمرین می‌دهیم. سپس آن را بر روی دیتای تست که تا الان توسط مدل دیده نشده است آزمایش می‌کنیم و مقدار دقت آن را محاسبه می‌کنیم.

```
clf = DecisionTreeClassifier(random_state=42)
clf.fit(X_train, Y_train)
```

```
Y_pred = clf.predict(X_test)
```





مقدار خطا

پس از تمرین دادن مدل و انجام آزمایش، مقدار دقت آن را محاسبه می‌کنیم. در قطعه کد زیر ابتدا مقدار دقت محاسبه شده و سپس یک جدول برای خطاها تشکیل می‌دهیم.

```
clf = DecisionTreeClassifier(random_state=42)
clf.fit(X_train, Y_train)
```

```
Y_pred = clf.predict(X_test)
```

دقت مدل بر روی داده کلپ زاخاری ۰/۸۹ بوده است که با توجه به کوچک بودن گراف و کم حجم بودن دیتای تمرین کاملاً عادی می‌باشد.

	Predicted: 0 (No Edge)	Predicted: 1 (Edge Exists)
Actual: 0 (No Edge)	91  Correct	7  Wrong
Actual: 1 (Edge Exists)	5  Wrong	10  Correct

همانطور که در جدول بالا مشاهده می‌شود، ۵ یال به اشتباه غایب تشخیص داده شده و در بخش یال‌های موجود نیز ۷ یال به اشتباه موجود در نظر گرفته شده است. خطای مدل با توجه به کم بودن بیش از حد داده کاملاً معقولانه می‌باشد. می‌توان گفت عملکرد کلی آن با توجه به جدول بالا تقریباً خوب بوده است.

تحلیل پیوندهای اشتباه

جدول زیر مقادیر معیارهای پیش‌بینی پیوند برای داده‌هایی که اشتباه پیش‌بینی شده‌اند را نمایش می‌دهد.

Jaccard_coefficient	Preferential_attachment	Adamic_adar
0.15384615	50	0.76310336
0	40	0
0.28571429	20	0.75538573
5.88235294e-02	7.20000000e+01	4.34294482e-01
0.52631579	204	10.45695074
0.125	20	0.35295612
0.05882353	32	0.43429448
0	51	0
0.25	24	1.26319535
0.05555556	48	0.45511961
1.05263158e-01	1.08000000e+02	1.15564200e+00
1.30434783e-01	1.53000000e+02	2.25292168e+00

با بررسی جدول بالا نتایج زیر حاصل می‌شود:

۱. مشکل در وابستگی بیش از حد به Preferential Attachment

- در مواردی که مقدار Jaccard نزدیک به صفر است، اما PA بالاست، احتمالاً مدل را گمراه کرده است.
- مثال (0.0, 51, 0.0) و (0.0, 40, 0.0)

۲. شاخص (Adamic-adar) در برخی موارد تأثیر زیادی گذاشته است

- مواردی که مقدار AA خیلی بالا است، حتی اگر PA متوسط باشد، مدل را به گمراه کرده است.

- مثال (0.25, 24, 1.2632) و (0.1053, 108, 1.1556)

۳. در مواردی که گره ها همسایه مشترک ندارند هم یال را تشخیص داده است

- برخی یال ها مقدار $Jaccard = 0$ دارند که یعنی هیچ همسایه مشترکی ندارند اما مدل آن ها را به عنوان پیوند انتخاب کرده است.

- مثال. (0.0, 51, 0.0), (0.0, 40, 0.0).

تأثیر نرمال سازی

با توجه به اینکه معیارهای محاسبه شده مقادیر متفاوتی دارند و برخی بیش از حد بزرگ هستند به نظر می آید که این مقادیر باید نرمال سازی بشوند. مدل را بر روی دو روش نرمال سازی تست کردیم.

روش اول: نرمال سازی به روش Min_Max

دقت حاصل شده پس از انجام نرمال سازی به روش Min-Max بر روی X_{train} و X_{test} مقدار ۰/۸۶ بود.

روش دوم: نرمال سازی به روش StandardScaler

دقت حاصل شده پس از انجام این مدل نرمال سازی عدد ۰/۸۲ بود.

با توجه به موارد بالا و اطلاعات موجود در مورد درخت های تصمیم به نظر می آید نرمال سازی در درخت تصمیم تقریباً غیر ضروری بوده و دقت مدل را کاهش می دهد.


```

import igraph as ig
import math
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.preprocessing import StandardScaler, MinMaxScaler

g = ig.Graph.Famous('Zachary')
e = g.get_edgelist()
v_count = g.vcount()

jaccard = np.zeros([v_count, v_count])
preferential_attachment = np.zeros([v_count, v_count])
adamic_adar = np.zeros([v_count, v_count])

for i in range(v_count):
    v = set(g.neighbors(i, mode='all'))
    for j in range(i+1, v_count):
        u = set(g.neighbors(j, mode='all'))
        jaccard[i,j] = len(u.intersection(v))/len(u.union(v))
        preferential_attachment[i,j] = len(u) * len(v)
        common_neighbors = v.intersection(u)
        score = sum(1/np.log(g.degree(w)) for w in common_neighbors if g.degree(w) > 1 )
        adamic_adar[i,j] = score

X = np.column_stack([
    jaccard[np.triu_indices(v_count, k=1)],
    preferential_attachment[np.triu_indices(v_count, k=1)],
    adamic_adar[np.triu_indices(v_count, k=1)]
])

Y = np.array([(1 if (i, j) in e or (j, i) in e else 0) for i in range(v_count) for j in range(i+1, v_count)])
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, random_state=4, test_size=0.2)

clf = DecisionTreeClassifier(random_state=42)
clf.fit(X_train, Y_train)
Y_pred = clf.predict(X_test)

accuracy = accuracy_score(Y_test, Y_pred)
conf_matrix = confusion_matrix(Y_test, Y_pred)
for i in range(len(Y_pred)):
    if Y_pred[i] != Y_test[i]:
        print(X_test[i])

print("Accuracy:", accuracy)
print("Confusion Matrix:\n", conf_matrix)

```