Лабораториска вежба 3
Граф невронски мрежи

StellarGraph Python библиотека

StellarGraph е Python библиотека за машинско учење на граф структурирани податоци. Библиотеката StellarGraph имплементира неколку најсовремени алгоритми за примена на методи на машинско учење за откривање на шаблони и давање на одговори на прашања користејќи граф структурирани податоци. Библиотеката е достапна на следната GitHub страна.

Инсталација на библиотеката:

```
pip install stellargraph
```

Во оваа библиотека се имплементирани многу најсовремени техники за граф невронски мрежи. Во оваа лабораториска вежба потребни се моделите Graph Convolutional Network и GraphSAGE.

Креирање на модел GCN во Keras

За работа со рамката може да се креира StellarGraph објект од постоечки NetworkX граф и дадените каратеристики и класи на јазлите. Графот е потребно да го имаме во NetworkX формат, а карактеристиките на јазлите и класите во Pandas Dataframe. Овие два објекта може да се искористат за да се креира објектот StellarGraph кој е соодветен за употреба во алгоритми на машинско учење за графови кои е имплементирани во библиотеката. За да се креира класа инстанца од оваа класа се користи следниот код:

```
from stellargraph.core.graph import StellarGraph, StellarDiGraph

G = StellarGraph(g, node_features=nodes)
print(G.info())
```

За да се пренесат податоците од графот во Keras модел потребно е да се креира генератор. Затоа што GCN е full-batch модел, може да се користи класата FullBatchNodeGenerator за да се предадат карактеристиките на јазлите и нормализираната матрица на графот во моделот. Доколку се специфицира method='gcn' како аргумент на FullBatchNodeGenerator ќе ја претпроцесира матрицата на соседство и на моделот ќе му ја предаде нормализираната Лапласова

матрица од графот. За тренирање ги мапираме само јазлите за тренирање заедно со целните вредности (класите).

```
from stellargraph.mapper.full_batch_generators import FullBatchNodeGenerator

generator = FullBatchNodeGenerator(G, method="gcn", sparse=False)
train_gen = generator.flow(train_data.index, train_targets)
```

Сега може со поставените параметри да се креира модел на машинско учење, за кое што ни се потребни неколку параметри:

- layer_sizes – листа на големини на скриените слоеви во моделот. Во примерот се користат два GCN слоеви со 16-димензионални скриени карактеристики на јазлите во секој слој.
- activations – листа на активациски функции кои се применуваат на излезот од секој слој.
- dropout=0.5 – означува 50% dropout на секој слој.

GCN модел може да се креира како следниот пример:

```
from stellargraph.layer import GCN


gcn = GCN(
    layer_sizes=[16, 16],
    activations=['relu', 'relu'],
    generator=generator,
    dropout=0.5
)
```

За да се креира модел на Keras може од GCN моделот да се извлечат влезните и излезните слоеви преку методот GCN.node_model

```
x_inp, x_out = gcn.node_model()
```

Креирање на модел GraphSAGE во Keras

Од даден NetworkX граф и карактеристиките и класите на јазлите може да се креира модел StellarGraph објект. За да се пренесат податоците од даден граф во Keras модел потребен е генератор на податоци кој ги пренесува податоците од графот во keras моделот. Генераторите се специјализирани за моделот и задачата на учење, па соодветно, за употреба на GraphSage за класификација на јазли, треба да се користи генераторот GraphSAGENodeGenerator. Потребни ни се два параметри, batch_size кој се користи при тренирање и бројот на јазли што се семплираат на секое ниво на моделот. Во примерот избираме модел со две нивоа со 10 јазли семплирани во првиот слој, а 5 во вториот.

```
from stellargraph.mapper.sampled_node_generators import GraphSAGENodeGenerator
```

```
batch_size = 50
num_samples = [10, 5]
generator = GraphSAGENodeGenerator(G, batch_size, num_samples)
train_gen = generator.flow(train_data.index, train_targets, shuffle=True)
```

Користејќи го методот generator.flow(), можеме да создадеме итератори на јазли што треба да се користат за тренирање, валидација или оценување на моделот. За тренирање ги користиме само јазлите за и целните вредности. Аргументот shuffle=True е даден на методот за подобрување на тренирањето.

За да можеме да го креираме моделот на машинско учење, потребни ни се уште неколку параметри:

- layer_sizes – листа на големина на скриените слоеви, во примерот се користат 32-димензионални скриени карактеристики на јазлите во секој слој.
- bias и dropout се внатрешни параметри на моделот.

```
from stellargraph.layer import GraphSAGE

graphsage_model = GraphSAGE(
    layer_sizes=[32, 32],
    generator=generator,
    bias=True,
    dropout=0.5,
)
```

Податочно множество – Cora

Податочното множество Cora се состои од 2 708 научни публикации класифицирани во една од седум класи. Мрежата се состои од 5 429 врски. Секој јазол претставува научен труд, а рабовите помеѓу јазлите претставуваат цитирана врска помеѓу двата документи. Секоја публикација во множеството е опишана преку вектор со 0/1 вредности кој покажува отсутност/присутност на соодветниот збор од речникот во насловот на публикацијата. Речникот се состои од 1 433 уникатни зборови. Податочното множество може да се преземе од следниот линк. Датотеката README обезбедува повеќе детали.

Задачи

Задача 1 – Класификација на јазли со GCN (5 поени)

Во оваа задача треба да се имплементира класификација на јазлите преку модел на длабоко учење кој ќе користи GCN слоеви. Првиот дел на моделот се 2 или 3 GCN слоеви по што се надоврзуваат 1, 2 или 3 скриени FC слоеви за класификација на јазлите. Најпрво потребно е графот да се вчита, да се одреди тренирачко и тестирачко множество на јазли и да се енкодираат класите во соодветни векторски репрезентации. Кодот за овој дел е истиот од претходната лабораториска вежба. Евалуацијата на моделите е истата како и претходната лабораториска вежба и може да ја искористите за пресметување на вредностите на accuracy, F1-micro и F1-macro. Направете споредба на дадените вредности на креираните модели со различен број на слоеви и различни големини на слоевите, а потоа визуализирајте ги node embeddings во дво-димензионален простор со користење на методот TSNE и различна боја за секоја класа на јазлите:

```
from sklearn.manifold import TSNE

trans = TSNE(n_components=2)
trans.fit_transform(X)
```

Напомена: При повик на функциите за тренирање fit_generator и за предвидување predict_generator не заборавајте да ги поставите вредностите на атрибутите steps_per_epoch, validation_steps и steps.

Задача 2 – Класификација на јазли со GraphSAGE (5 поени)

Направете и втор модел за класификација на јазли кој го имплементира GraphSAGE. Првиот дел на моделот се 2 или 3 слоеви во GraphSAGE, по што се надоврзуваат 1, 2 или 3 скриени FC слоеви за класификација на јазлите. Направете класификација на јазлите преку споредба на различните големини и комбинации на слоеви на моделите. Дали се добиваат резултати кои се споредливи со GCN моделот? Интерпретирајте ги резултатите.

Потребно е да ги визуелизирате креираните node embeddings во 2Д простор со користење на TSNE методот и различна боја за секоја класа. Какви се изгенерираните node embeddings во споредба со node embeddings на GCN моделот?