DeepRobust

[Deep Robust]

- Graph Package

Graph Dataset

1. Clean (Unattacked) Graphs for Node Classification

deeprobust.graph.data.Dataset

- data.adj : adjacency matrix ; [num_nodes, num_nodes]
- data.features : Node feature matrix ; [num_nodes, num_node_features]
- data.labels : target to train against; [num_nodes, *]
- data.train_idx : training node 인덱스
- data.val_idx : validation node 인덱스
- data.test_idx : test node 인덱스

디폴트로는 그래프의 가장 큰 부분을 선택하게끔 되어있지만 유저가 이를 조정해 줄 수 있음.

사용되는 데이터셋으로는 : Cora, Cora-ML, Citeseer, Pubmed, Polblogs, ACM, BlogCatalog, Flicker, UAI 가 있다.

디폴트로 data는 deeprobust.graph.utils.get_train_val_test 로 data split이 이루어지게 된다. 기본적으로는 train:val:test = 1:1:8 의 비율로 되어있다. deeprobust.graph.utils.get_train_val_test 혹은 deeprobust.graph.utils.get_train_val_test_gcn 으로 따로 split을 만들어줄 수도 있다.

deeprobust.graph.utils.get_train_val_test_gcn 에서는 setting이라는 parameter가 있어서 선택해줄 수 있는데 {"nettack", "gcn", "prognn"} 중에 선택이 가능하다.

setting = "nettack" : datasplit 비율 1:1:8. Graph의 가장 큰 connection 을 선택

setting = "gcn" : 전체 graph를 사용하게 됨. Data split : training에서는 class 당 20 node 씩 할당, validation에서는 500 node씩 할당, test에서는 1000 nodes씩 할당. (random choose)

setting = "prognn" : graph의 가장 큰 connection을 선택하고 PROGNN에서 제공되는 datasplit을 사용(1:1:8)

How to load DeepRobust dataset

```
from deeprobust.graph.data import Dataset # loading cora dataset data
= Dataset(root='/tmp/', name='cora', seed=15) adj, features, labels =
data.adj, data.features, data.labels idx_train, idx_val, idx_test =
data.idx_train, data.idx_val, data.idx_test # you can also split the
data by yourself idx_train, idx_val, idx_test =
get_train_val_test(adj.shape[0], val_size=0.1, test_size=0.8) #
loading acm dataset data = Dataset(root='/tmp/', name='acm', seed=15)
```

DeepRobust는 Pytorch에서 사용되는 Amazon과 Coauthor dataset도 제공한다.

Geometric : Amazon-Computers, Amazon-Photo, Coauthor-CS, Coauthor-Physics

2. Attacked Graphs for Node Classification

metaattack과 nettack을 통해 perturbed 된 graph를 제공한다. 저자들의 Tensorflow implementation을 통해 attack을 받은 graph들이다. random split seed는 15를 사용했다.

다양한 GNN들의 성능은 ProGNN 논문에서 찾아볼 수 있다. 그리고 그것들은 deeprobust.graph.data.PrePtbDataset 에서 찾아볼 수 있다. attribute는 adj 하나만 가지고 있다.

그로 인해 deeprobust.graph.dataPrePtbDataset은 deeprobust.graph.data.Dataset과 같이 사용되어 node feature과 label을 얻을 수 있다.

metattack을 위해 Cora, Citeseer, Polblogs, Pubmed를 위한 attacked graph를 제공하고 perturbation rate는 [0.05, 0.1, 0.15, 0.2, 0.25] 중에 선택이 가능하다.

```
from deeprobust.graph.data import Dataset, PrePtbDataset # You can
either use setting='prognn' or seed=15 to get the prognn splits data =
Dataset(root='/tmp/', name='cora', setting='prognn') data =
Dataset(root='/tmp/', name='cora', seed=15) # since the attacked graph
are generated under seed 15 adj, features, labels = data.adj,
```

```
data.features, data.labels idx_train, idx_val, idx_test =
data.idx_train, data.idx_val, data.idx_test # Load meta attacked data
perturbed_data = PrePtbDataset(root='/tmp/', name='cora',
attack_method='meta', ptb_rate=0.05) perturbed_adj =
perturbed_data.adj
```

netattack 에서는 Cora, Citeseer, Polblogs, Pubmed를 위한 attacked graph들을 제공하고 있다. ptb_rate는 각 노드에 대해 들어간 perturbations의 수를 의미한다. [1.0, 2.0, 3.0 , 4.0, 5.0] 중에 선택되어질 수 있다.

```
from deeprobust.graph.data import Dataset, PrePtbDataset # data =
Dataset(root='/tmp/', name='cora', seed=15) # since the attacked graph
are generated under seed 15 data = Dataset(root='/tmp/', name='cora',
setting='prognn') adj, features, labels = data.adj, data.features,
data.labels idx_train, idx_val, idx_test = data.idx_train,
data.idx_val, data.idx_test # Load nettack attacked data
perturbed_data = PrePtbDataset(root='/tmp/', name='cora',
attack_method='nettack', ptb_rate=3.0) # here ptb_rate means number of
perturbation per nodes perturbed_adj = perturbed_data.adj idx_test =
perturbed_data.target_nodes
```

3. Converting Graph Data between DeepRobust and PyTorch Geometric

DeepRobust data를 PyTorch Geometric으로 변경하는 tool들을 제공하고 있다.

deeprobust.graph.data.Dpr2Pyg : DeepRobust data → PyTorch Geometric

deeprobust.graph.data.Pyg2Dpr : Pytorch Geometric → DeepRobust

```
from deeprobust.graph.data import Dataset, Dpr2Pyg, Pyg2Dpr data =
Dataset(root='/tmp/', name='cora') # load clean graph pyg_data =
Dpr2Pyg(data) # convert dpr to pyg print(pyg_data) print(pyg_data[0])
dpr_data = Pyg2Dpr(pyg_data) # convert pyg to dpr print(dpr_data.adj)
```

4. Load OGB Datasets

OGB(Open Graph Benchmark)

DeepRobust는 OGB dataset format(Pyg data format)을 DeepRobust format으로 바꿔주는 interface를 제공한다.

```
from ogb.nodeproppred import PygNodePropPredDataset from
deeprobust.graph.data import Pyg2Dpr pyg_data =
PygNodePropPredDataset(name = 'ogbn-arxiv') dpr_data =
Pyg2Dpr(pyg_data) # convert pyg to dpr
```

5. Load Pytorch Geometric Amazon and Coauthor Datasets

DeepRobust는 Amazon dataset과 Coauthor dataset을 활용할 수 있게 해준다.

deeprobust.graph.data.AmazonPyg : Amazon dataset

deeprobust.graph.data.coauthorPyg : Coauthor dataset

```
from deeprobust.graph.data import AmazonPyg computers =
AmazonPyg(root='/tmp', name='computers') print(computers)
print(computers[0]) photo = AmazonPyg(root='/tmp', name='photo')
print(photo) print(photo[0])
```

```
from deeprobust.graph.data import CoauthorPyg cs =
CoauthorPyg(root='/tmp', name='cs') print(cs) print(cs[0]) physics =
CoauthorPyg(root='/tmp', name='physics') print(physics) print(physics[0])
```

Introduction to Graph Attack with Examples

2가지 타입의 graph attack algorithm을 제공한다.

(1) : targeted attack : deeprobust.graph.targeted_attack

(2) : global attack : deeprobust.graph.global_attack

Global (Untargeted) Attack for Node Classification

Aims to fool GNNs into giving wrong predictions on all given nodes.
(전체 노드에 대해 wrong prediction을 줘서 fool)

- deeprobust.graph.global_attack.Metattack
- deeprobust.graph.global_attack.MetaApprox
- deeprobust.graph.global_attack.DICE
- deeprobust.graph.global_attack.MinMax
- deeprobust.graph.global_attack.PGDAttack
- deeprobust.graph.global_attack.NIPA
- deeprobust.graph.global_attack.Random
- deeprobust.graph.global_attack.NodeEmbeddingAttack
- deeprobust.graph.global_attack.OtherNodeEmbeddingAttack

NodeEmbeddingAttack과 OtherNodeEmbeddingAttack을 제외하고는 모드
adjacency matrix, node feature matrix, label을 input으로 받는다.

Adjacency matrix의 경우 scipy.sparse.csr_matrix 를 이용하고

feature matrix의 경우 scipy.sparse.csr_matrix 혹은 numpy.array를 이용한다.

attack algorithm은 torch.tensor를 통해 전달된다. torch.tensor를 input으로 주게되어도 알고리즘이 자동으로 이를 다루게 된다.

```
import numpy as np from deeprobust.graph.data import Dataset from
deeprobust.graph.defense import GCN from deeprobust.graph.global_attack
import Metattack data = Dataset(root='/tmp/', name='cora') adj, features,
labels = data.adj, data.features, data.labels idx_train, idx_val,
idx_test = data.idx_train, data.idx_val, data.idx_test idx_unlabeled =
np.union1d(idx_val, idx_test) idx_unlabeled = np.union1d(idx_val,
idx_test) # Setup Surrogate model surrogate =
GCN(nfeat=features.shape[1], nclass=labels.max().item()+1, nhid=16,
dropout=0, with_relu=False, with_bias=False, device='cpu').to('cpu')
surrogate.fit(features, adj, labels, idx_train, idx_val, patience=30) #
Setup Attack Model model = Metattack(surrogate, nnodes=adj.shape[0],
feature_shape=features.shape, attack_structure=True,
attack_features=False, device='cpu', lambda_=0).to('cpu') # Attack
model.attack(features, adj, labels, idx_train, idx_unlabeled,
n_perturbations=10, ll_constraint=False) modified_adj =
model.modified_adj # modified_adj is a torch.tensor
```

Targeted Attack for Node classification

Aims to fool GNNs into give wrong predictions on a subset of nodes.
(일부 노드에 대해서만 wrong prediction을 줘서 fool)

- deeprobust.graph.targeted_attack.Nettack
- deeprobust.graph.targeted_attack.RLS2V
- deeprobust.graph.targeted_attack.FGA
- deeprobust.graph.targeted_attack.RND
- deeprobust.graph.targeted_attack.IGAttack

위에 써져있는 애들은 전부 adjacency matrix, node feature matrix, label을 input으로 받는다.

Adjacency matrix와 Feature matrix의 format 과 관련된 부분은 앞에서 본 것과 같다.

```
from deeprobust.graph.data import Dataset from deeprobust.graph.defense
import GCN from deeprobust.graph.targeted_attack import Nettack data =
Dataset(root='/tmp/', name='cora') adj, features, labels = data.adj,
data.features, data.labels idx_train, idx_val, idx_test = data.idx_train,
data.idx_val, data.idx_test # Setup Surrogate model surrogate =
GCN(nfeat=features.shape[1], nclass=labels.max().item()+1, nhid=16,
dropout=0, with_relu=False, with_bias=False, device='cpu').to('cpu')
surrogate.fit(features, adj, labels, idx_train, idx_val, patience=30) #
Setup Attack Model target_node = 0 model = Nettack(surrogate,
nnodes=adj.shape[0], attack_structure=True, attack_features=True,
device='cpu').to('cpu') # Attack model.attack(features, adj, labels,
target_node, n_perturbations=5) modified_adj = model.modified_adj # scipy
sparse matrix modified_features = model.modified_features # scipy sparse
matrix
```

Introduction to Graph Defense with Examples

DeepRobust에서 제공되는 graph attack algorithm을 소개할 예정.

Test your model's robustness on poisoned graph

Deeprobust에서는 GNNs의 robustness를 강화하기 위한 defense method들을 제공한다.

Victim Models:

- deeprobust.graph.defense.GCN
- deeprobust.graph.defense.GAT
- deeprobust.graph.defense.ChebNet
- deeprobust.graph.defense.SGC

Node Embedding Victim Models

- deeprobust.graph.defense.DeepWalk
- deeprobus.graph.defense.Node2Vec

Defense Methods

- deeprobust.graph.defense.GCNJaccard
- deeprobust.graph.defense.GCNSVD
- deeprobust.graph.defense.ProGNN
- deeprobust.graph.defense.RGCN
- deeprobust.graph.defense.SimPGCN
- deeprobust.graph.defense.AdvTraining

1. Load pre-attacked graph data

```
from deeprobust.graph.data import Dataset, PrePtbDataset # load the
prognn splits by using setting='prognn' # because the attacked graphs are
generated under prognn splits data = Dataset(root='/tmp/', name='cora',
setting='prognn') adj, features, labels = data.adj, data.features,
data.labels idx_train, idx_val, idx_test = data.idx_train, data.idx_val,
data.idx_test # Load meta attacked data perturbed_data =
PrePtbDataset(root='/tmp/', name='cora', attack_method='meta',
ptb_rate=0.05) perturbed_adj = perturbed_data.adj
```

2. You can also choose to load graphs attacked by nettack

```
# Load nettack attacked data perturbed_data = PrePtbDataset(root='/tmp/',
name='cora', attack_method='nettack', ptb_rate=3.0) # here ptb_rate means
number of perturbation per nodes perturbed_adj = perturbed_data.adj
idx_test = perturbed_data.target_nodes
```

3. Train a victim model (GCN) on clean/poisoned graph

```
from deeprobust.graph.defense import GCN gcn =
GCN(nfeat=features.shape[1], nhid=16, nclass=labels.max().item() + 1,
dropout=0.5, device='cpu') gcn = gcn.to('cpu') gcn.fit(features, adj,
labels, idx_train, idx_val) # train on clean graph with earlystopping
gcn.test(idx_test) gcn.fit(features, perturbed_adj, labels, idx_train,
idx_val) # train on poisoned graph gcn.test(idx_test)
```

4. Train defense models (GCN-Jaccard, RGCN, ProGNN) poisoned graph

```
from deeprobust.graph.defense import GCNJaccard model =
GCNJaccard(nfeat=features.shape[1], nhid=16, nclass=labels.max().item() +
1, dropout=0.5, device='cpu').to('cpu') model.fit(features,
perturbed_adj, labels, idx_train, idx_val, threshold=0.03)
model.test(idx_test)
```

```
from deeprobust.graph.defense import GCNJaccard model =
RGCN(nnodes=perturbed_adj.shape[0], nfeat=features.shape[1],
nclass=labels.max()+1, nhid=32, device='cpu') model.fit(features,
perturbed_adj, labels, idx_train, idx_val, train_iters=200, verbose=True)
model.test(idx_test)
```

Using PyTorch Geometric in DeepRobust

DeepRobust는 PyTorch Geometric과 DeepRobust 사이의 data를 convert 하는 기능을 제공한다.

Converting Graph Data between DeepRobust and PyTorch Geometric

- deeprobust.graph.data.Dpr2Pyg : DeepRobust data to PyTorch Geometric
- deeprobust.graph.data.PygDpr : PyTorch Geometric data to DeepRobust

아래 코드 예시에서는 data instance를 먼저 생성한 후 pytorch geometric format으로 바꿔주고 있다.

```
from deeprobust.graph.data import Dataset, Dpr2Pyg, Pyg2Dpr data =
Dataset(root='/tmp/', name='cora') # load clean graph pyg_data =
Dpr2Pyg(data) # convert dpr to pyg print(pyg_data) print(pyg_data[0])
dpr_data = Pyg2Dpr(pyg_data) # convert pyg to dpr print(dpr_data.adj)
```

pyg_data는 PyTorch Geometric format의 perturbed data가 된다. 이를 다양한 PyTorch Geometric 모델들에 적용시켜 사용할 수 있다!

Load OGB Datasets

Open Graph Benchmark (OGB)

DeepRobust에서는 OGB dataset format (Pyg data format)을 DeepRobust format으로 바꿔주는 tool을 제공합니다.

```
from ogb.nodeproppred import PygNodePropPredDataset from
deeprobust.graph.data import Pyg2Dpr pyg_data =
PygNodePropPredDataset(name = 'ogbn-arxiv') dpr_data = Pyg2Dpr(pyg_data)
# convert pyg to dpr
```

Load Pytorch Geometric Amazon and Coauthor Datasets

DeepRobust에서는 Amazon dataset과 Coauthor dataset을 사용할 수 있도록 해줍니다.

(Amazon-Computers, Amazon-Photo, Coauthor-CS, Coauthor-Physics (from PyTorch Geometric))

deeprobust.graph.data.AmazonPyg 와 deeprobust.graph.data.CoauthorPyg 를 통해 사용 가능.

AmazonPyg

```
from deeprobust.graph.data import AmazonPyg computers =
AmazonPyg(root='/tmp', name='computers') print(computers)
print(computers[0]) photo = AmazonPyg(root='/tmp', name='photo')
print(photo) print(photo[0])
```

CoauthorPyg

```
from deeprobust.graph.data import CoauthorPyg cs =
CoauthorPyg(root='/tmp', name='cs') print(cs) print(cs[0]) physics =
CoauthorPyg(root='/tmp', name='physics') print(physics) print(physics[0])
```

Working on PyTorch Geometric Models

PyTorch Geometric 기반으로 GNNs를 사용하는 방법.

이 코드에서는 GAT를 deeprobust.graph.defense.GAT 로, ChebNet을 deeprobust.graph.defense.ChebNet 으로 사용할 예정.

기본적으로 우선 DeepRobust data를 PyTorch Geometric data로 변경한 후 Pyg 모델들을 훈련시켜야 한다.

```
from deeprobust.graph.data import Dataset, Dpr2Pyg, PrePtbDataset from
deeprobust.graph.defense import GAT data = Dataset(root='/tmp/',
name='cora', seed=15) adj, features, labels = data.adj, data.features,
data.labels idx_train, idx_val, idx_test = data.idx_train, data.idx_val,
data.idx_test gat = GAT(nfeat=features.shape[1], nhid=8, heads=8,
nclass=labels.max().item() + 1, dropout=0.5, device='cpu') gat =
gat.to('cpu') pyg_data = Dpr2Pyg(data) # convert deeprobust dataset to
pyg dataset gat.fit(pyg_data, patience=100, verbose=True) # train with
earlystopping gat.test() # test performance on clean graph # load
perturbed graph perturbed_data = PrePtbDataset(root='/tmp/', name='cora',
attack_method='meta', ptb_rate=0.05) perturbed_adj = perturbed_data.adj
pyg_data.update_edge_index(perturbed_adj) # inplace operation
gat.fit(pyg_data, patience=100, verbose=True) # train with earlystopping
gat.test() # test performance on perturbed graph
```

```
from deeprobust.graph.data import Dataset, Dpr2Pyg from
deeprobust.graph.defense import ChebNet data = Dataset(root='/tmp/',
name='cora') adj, features, labels = data.adj, data.features, data.labels
idx_train, idx_val, idx_test = data.idx_train, data.idx_val,
data.idx_test cheby = ChebNet(nfeat=features.shape[1], nhid=16,
num_hops=3, nclass=labels.max().item() + 1, dropout=0.5, device='cpu')
cheby = cheby.to('cpu') pyg_data = Dpr2Pyg(data) # convert deeprobust
dataset to pyg dataset cheby.fit(pyg_data, patience=10, verbose=True) #
train with earlystopping cheby.test()
```

Node Embedding Attack and Defense

Node embedding attack은 bad-quality 임베딩을 produce하여 node embedding 모델들을 fool 하는 것을 목표로 한다.

- deeprobust.graph.global_attack.NodeEmbeddingAttack
- deeprobust.graph.global_attack.OtherNodeEmbeddingAttack

얘네들은 adjacency matrix만 input으로 받는다. (scipy.sparse.csr_matrix form으로 들어감)

attack_type을 add edges 혹은 remove edges 둘 중 하나로 특정 지을 수 있다.

```
from deeprobust.graph.data import Dataset from
deeprobust.graph.global_attack import NodeEmbeddingAttack data =
Dataset(root='/tmp/', name='cora_ml', seed=15) adj, features, labels =
data.adj, data.features, data.labels model = NodeEmbeddingAttack()
model.attack(adj, attack_type="remove") modified_adj = model.modified_adj
model.attack(adj, attack_type="remove", min_span_tree=True) modified_adj
= model.modified_adj model.attack(adj, attack_type="add",
n_candidates=10000) modified_adj = model.modified_adj model.attack(adj,
attack_type="add_by_remove", n_candidates=10000) modified_adj =
model.modified_adj
```

OtherNodeEmbeddingAttack

ICML 2019에 발표된 Adversarial Attacks on Node Embeddings via Graph Poisoning 을 baseline으로 두고 있다. (degree, eigencentrality, random) 중에 공격 type을 선택할 수 있다.

```
from deeprobust.graph.data import Dataset from
deeprobust.graph.global_attack import OtherNodeEmbeddingAttack data =
Dataset(root='/tmp/', name='cora_ml', seed=15) adj, features, labels =
data.adj, data.features, data.labels model =
OtherNodeEmbeddingAttack(type='degree') model.attack(adj,
attack_type="remove") modified_adj = model.modified_adj # model =
OtherNodeEmbeddingAttack(type='eigencentrality') model.attack(adj,
attack_type="remove") modified_adj = model.modified_adj # model =
OtherNodeEmbeddingAttack(type='random') model.attack(adj,
attack_type="add", n_candidates=10000) modified_adj = model.modified_adj
```

Node Embedding Victim Models

DeepWalk와 Node2Vec 이라는 2개의 node embedding victim model들을 제공한다.

- deeprobust.graph.defense.Deepwalk
- deeprobust.graph.defense.Node2Vec

Three major functions in the two classes

- fit() : node embedding model들을 훈련시키고 embedding을 self.embedding에 저장
- evaluate_node_classification()
- evaluate_link_prediction()

```
from deeprobust.graph.data import Dataset from deeprobust.graph.defense
import DeepWalk from deeprobust.graph.global_attack import
NodeEmbeddingAttack import numpy as np dataset_str = 'cora_ml' data =
Dataset(root='/tmp/', name=dataset_str, seed=15) adj, features, labels =
data.adj, data.features, data.labels idx_train, idx_val, idx_test =
data.idx_train, data.idx_val, data.idx_test print("Test DeepWalk on clean
graph") model = DeepWalk(type="skipgram") model.fit(adj)
print(model.embedding)
```

모델 훈련이 끝나고 난 후에는 node classification과 link prediction에 대한 성능 test를 할 수 있다.

```
print("Test DeepWalk on node classification...") #
model.evaluate_node_classification(labels, idx_train, idx_test,
lr_params={"max_iter": 1000}) model.evaluate_node_classification(labels,
idx_train, idx_test) print("Test DeepWalk on link prediciton...")
model.evaluate_link_prediction(adj, np.array(adj.nonzero()).T)
```

이후 attacked graph에 대해 성능 test를 할 수 있다.

```
# set up the attack model attacker = NodeEmbeddingAttack()
attacker.attack(adj, attack_type="remove", n_perturbations=1000)
modified_adj = attacker.modified_adj print("Test DeepWalk on attacked
graph") model.fit(modified_adj)
model.evaluate_node_classification(labels, idx_train, idx_test)
```

Image Package

Image Attack and Defense

Image package에서 attack 과 defense API의 사용 예시를 보여줄 것임.

Attack Example

```
from deeprobust.image.attack.pgd import PGD from deeprobust.image.config
import attack_params from deeprobust.image.utils import download_model
import torch import deeprobust.image.netmodels.resnet as resnet URL =
"https://github.com/I-am-
Bot/deeprobust_model/raw/master/CIFAR10_ResNet18_epoch_50.pt"
download_model(URL, "$MODEL_PATH$") model = resnet.ResNet18().to('cuda')
model.load_state_dict(torch.load("$MODEL_PATH$")) model.eval()
transform_val = transforms.Compose([transforms.ToTensor()]) test_loader =
torch.utils.data.DataLoader( datasets.CIFAR10('deeprobust/image/data',
train = False, download=True, transform = transform_val), batch_size =
10, shuffle=True) x, y = next(iter(test_loader)) x = x.to('cuda').float()
adversary = PGD(model, device) Adv_img = adversary.generate(x, y,
**attack_params['PGD_CIFAR10'])
```

Defense Example

```
model = Net() train_loader = torch.utils.data.DataLoader(
datasets.MNIST('deeprobust/image/defense/data', train=True,
download=True, transform=transforms.Compose([transforms.ToTensor()])),
batch_size=100, shuffle=True) test_loader = torch.utils.data.DataLoader(
datasets.MNIST('deeprobust/image/defense/data', train=False,
transform=transforms.Compose([transforms.ToTensor()])),
batch_size=1000,shuffle=True) defense = PGDtraining(model, 'cuda')
defense.generate(train_loader, test_loader,
**defense_params["PGDtraining_MNIST"])
```