DEEP GRAPH INFOMAX

Petar Veličković, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, R Devon Hjelm

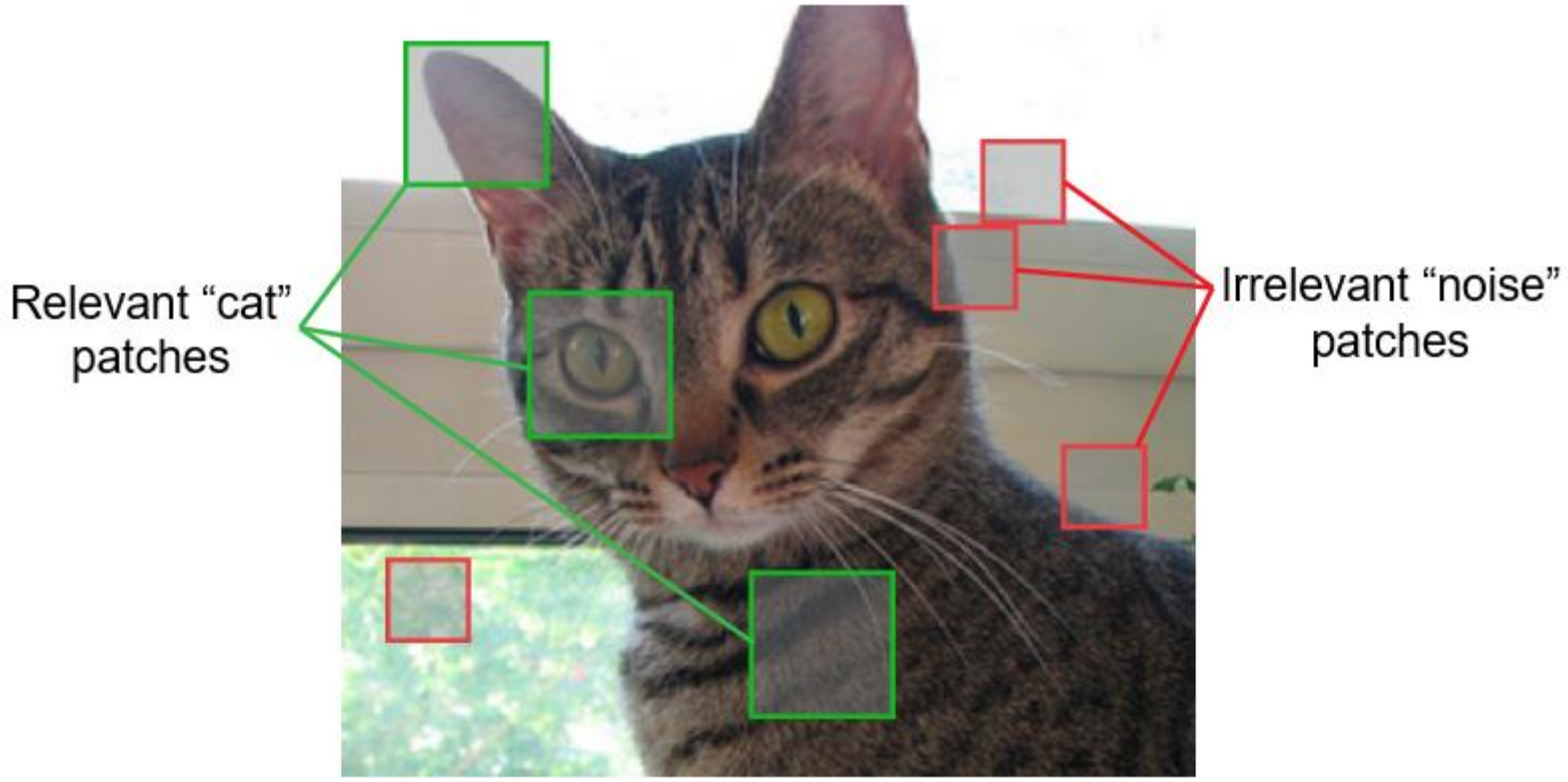
제어자동화특강 기말발표

IL Jae Kwon

Problem Setting

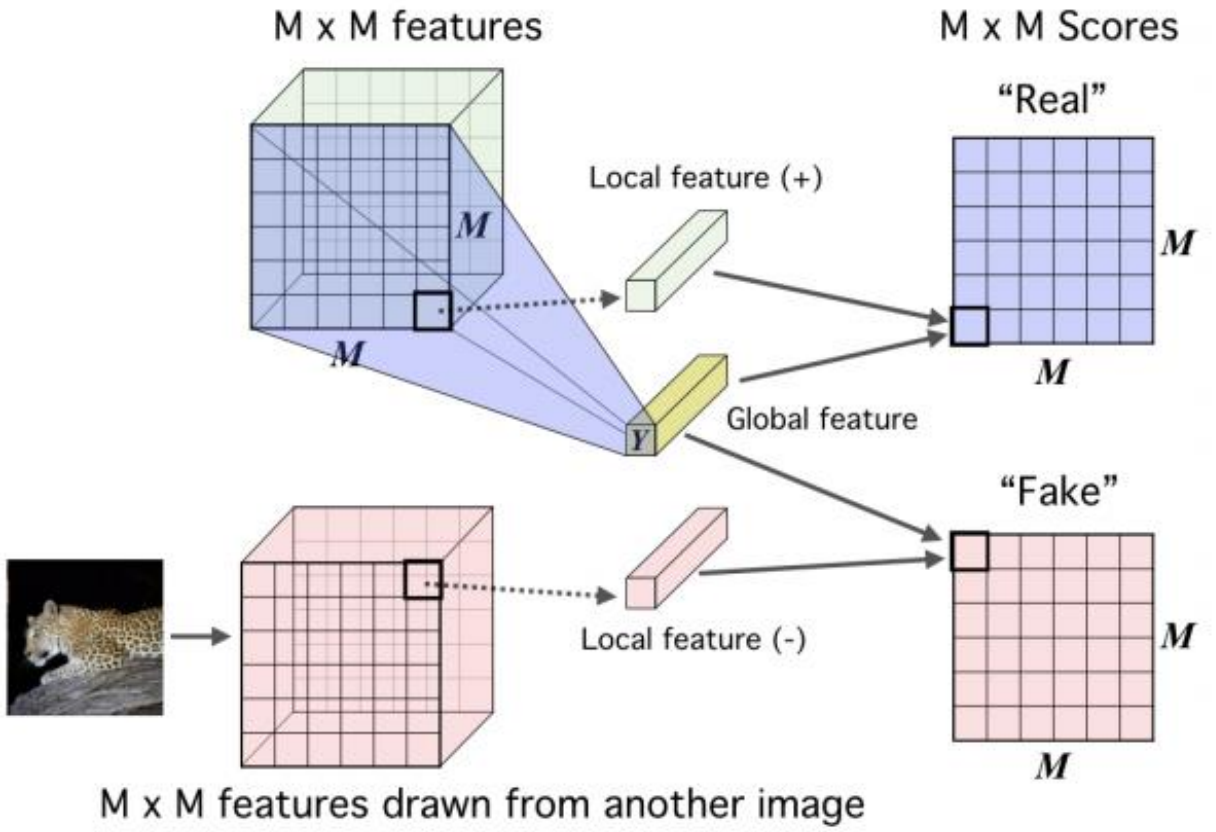
- Previous problem in Unsupervised representation learning in graph
  - Traditional metrics such as personalized PageRank use random walk for unsupervised representation learning
  - However, random walk objective is known to over-emphasize proximity information at the expense of structural information
  - Also, performance depends highly on hyperparameters such as temperature in heat kernel
- What about Mutual Information Maximization?
  - Deep InfoMax (DIM) trains encoder to maximize the mutual information between high-level global representation and local part of the input
  - In Deep Graph Infomax (DGI) adapts idea from DIM to graph-structured domain

Deep InfoMax (DIM)



Learning deep representations by mutual information estimation and maximization
R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, Yoshua Bengio

Deep InfoMax (DIM)



$$(\hat{\omega}, \hat{\psi})_L = \arg\max_{\omega,\psi} \frac{1}{M^2} \sum_{i=1}^{M^2} \widehat{\mathcal{I}}_{\omega,\psi}(C_\psi^{(i)}(X); E_\psi(X)).$$

Learning deep representations by mutual information estimation and maximization
R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, Yoshua Bengio

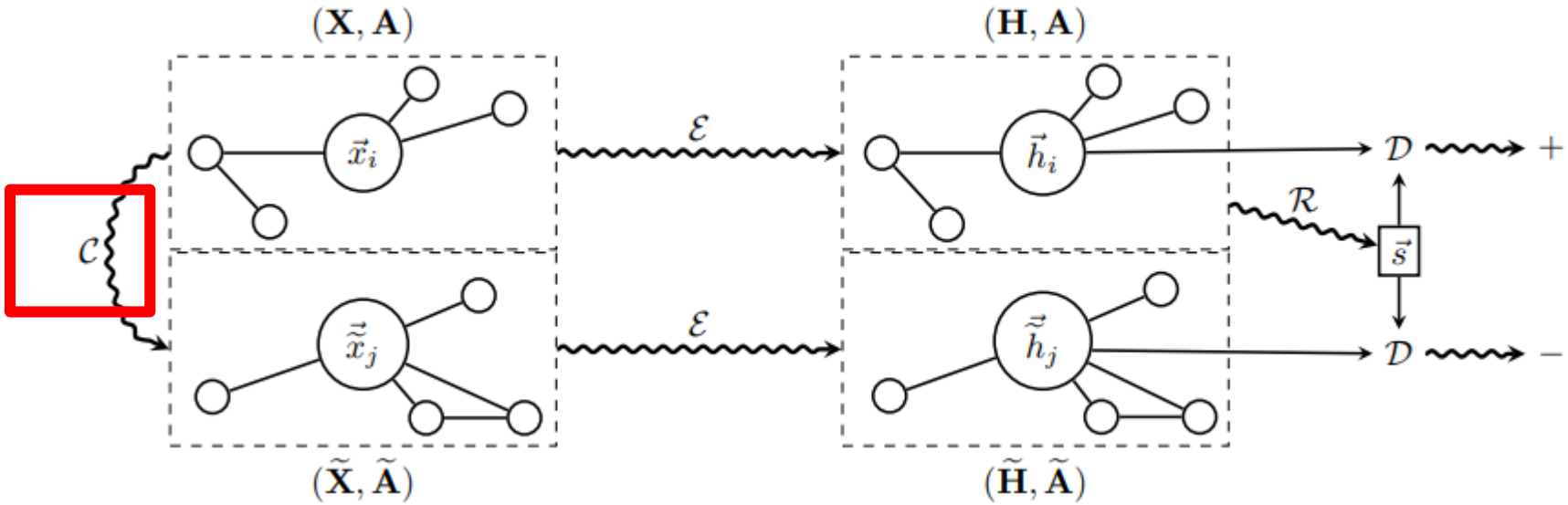
Deep Graph Infomax (DGI)

- Contributions
  - Learn node representation in an unsupervised manner
    - Maximize mutual information between node representation and global representation
  - First approach to use mutual information maximization in graph structured data
  - Outperform both supervised and unsupervised baselines

Method

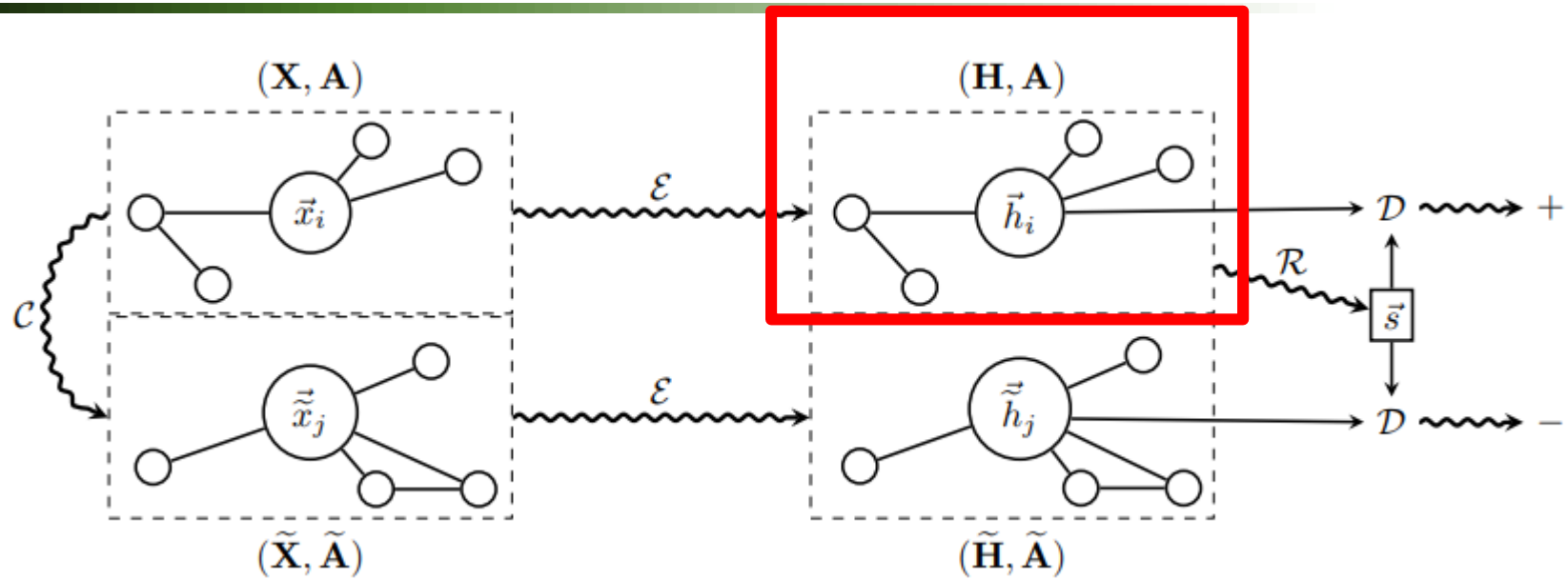
- Local – Global Mutual Information Maximization
  - Maximize MI between node representation, *h* and summary representation, *s*
    - Summary representation of the graph is obtained by readout function
  - For implement, DGI use discriminator, *D* which represents scores that measures the closeness between *h* and *s*
  - Negative samples required for the discriminator is provided by corruption function *C*
    - In inductive setting, choice of corruption function may govern the property of encoder
  - Use noise-contrastive type objective with standard binary cross-entropy loss

Method



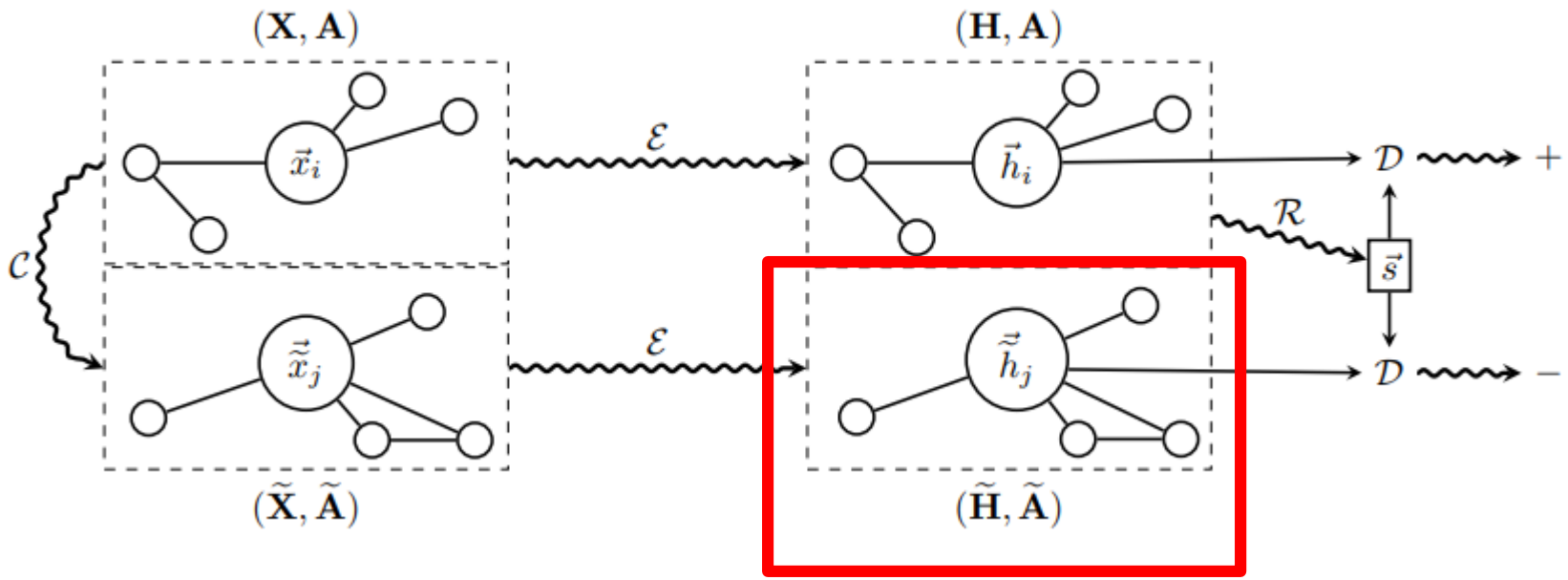
1. Sample a negative example by using corruption function C
2. Get patch representation $\vec{h}_i$ by encoder: $H = \varepsilon(X, A) = \{\vec{h}_1, \vec{h}_2, \cdots, \vec{h}_N\}$
3. Get patch representation $\tilde{\vec{h}}_i$ by encoder: $H = \varepsilon(\tilde{X}, \tilde{A}) = \{\tilde{\vec{h}}_1, \tilde{\vec{h}}_2, \cdots, \tilde{\vec{h}}_N\}$ from negative sample graph
4. Summarize input graph by readout function: $\vec{s} = R(H)$
5. Update ε, R, D maximizing mutual information

Method



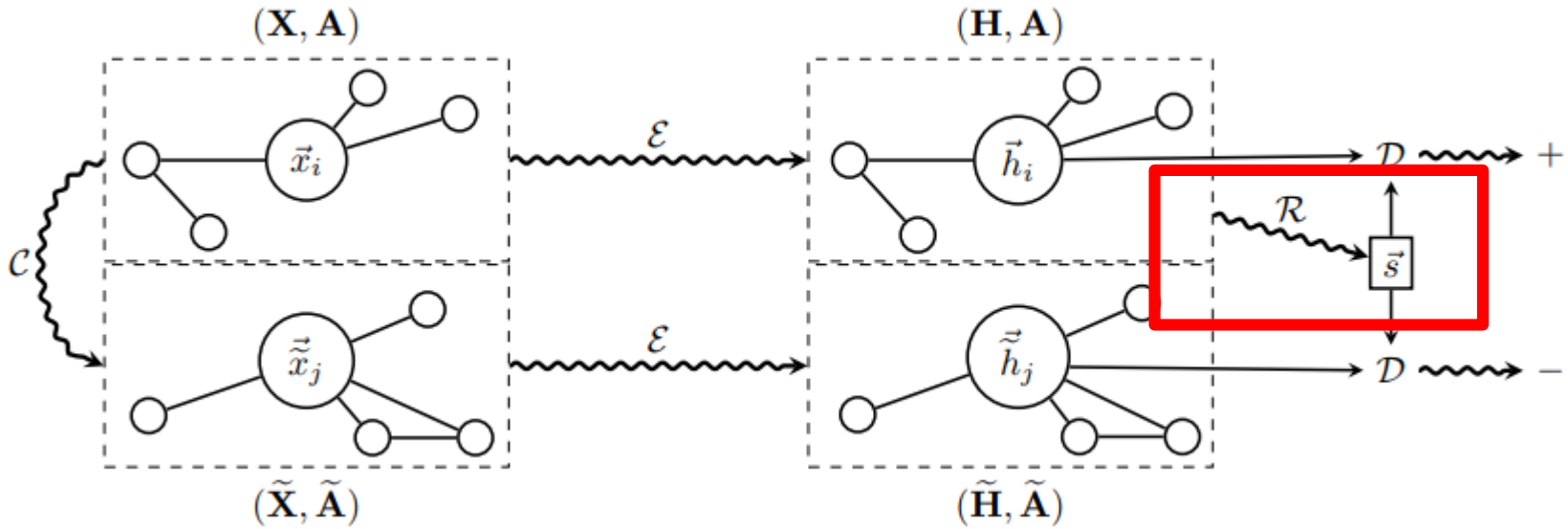
1. Sample a negative example by using corruption function C
2. Get patch representation $\vec{h}_i$ by encoder: $H = \varepsilon(X, A) = \{\vec{h}_1, \vec{h}_2, \cdots, \vec{h}_N\}$
3. Get patch representation $\tilde{\vec{h}}_i$ by encoder: $H = \varepsilon(\tilde{X}, \tilde{A}) = \{\tilde{\vec{h}}_1, \tilde{\vec{h}}_2, \cdots, \tilde{\vec{h}}_N\}$ from negative sample graph
4. Summarize input graph by readout function: $\vec{s} = R(H)$
5. Update ε, R, D maximizing mutual information

Method



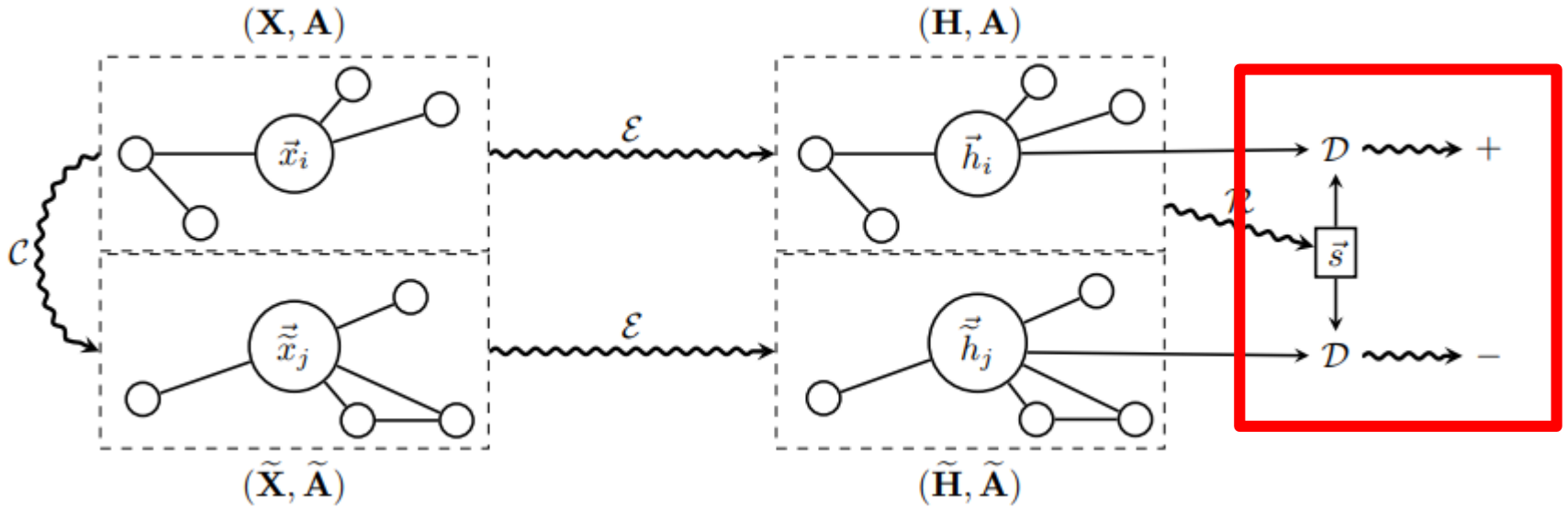
1. Sample a negative example by using corruption function C
2. Get patch representation $\vec{h}_i$ by encoder: $H = \varepsilon(X, A) = \{\vec{h}_1, \vec{h}_2, \cdots, \vec{h}_N\}$
3. Get patch representation $\tilde{\vec{h}}_i$ by encoder: $H = \varepsilon(\tilde{X}, \tilde{A}) = \{\tilde{\vec{h}}_1, \tilde{\vec{h}}_2, \cdots, \tilde{\vec{h}}_N\}$ from negative sample graph
4. Summarize input graph by readout function: $\vec{s} = R(H)$
5. Update ε, R, D maximizing mutual information

Method



1. Sample a negative example by using corruption function C
2. Get patch representation $\vec{h}_i$ by encoder: $H = \varepsilon(X, A) = \{\vec{h}_1, \vec{h}_2, \cdots, \vec{h}_N\}$
3. Get patch representation $\tilde{\vec{h}}_i$ by encoder: $H = \varepsilon(\tilde{X}, \tilde{A}) = \{\tilde{\vec{h}}_1, \tilde{\vec{h}}_2, \cdots, \tilde{\vec{h}}_N\}$ from negative sample graph
4. Summarize input graph by readout function: $\vec{s} = R(H)$
5. Update ε, R, D maximizing mutual information

Method



1. Sample a negative example by using corruption function C
2. Get patch representation $\vec{h}_i$ by encoder: $H = \varepsilon(X, A) = \{\vec{h}_1, \vec{h}_2, \cdots, \vec{h}_N\}$
3. Get patch representation $\tilde{\vec{h}}_i$ by encoder: $H = \varepsilon(\tilde{X}, \tilde{A}) = \{\tilde{\vec{h}}_1, \tilde{\vec{h}}_2, \cdots, \tilde{\vec{h}}_N\}$ from negative sample graph
4. Summarize input graph by readout function: $\vec{s} = R(H)$
5. Update ε, R, D maximizing mutual information

Method

- Mutual Information Maximization

$$\mathcal{L} = \frac{1}{N+M}\left(\sum_{i=1}^{N}\mathbb{E}_{(\mathbf{X},\mathbf{A})}\left[\log\mathcal{D}\left(\vec{h}_i,\vec{s}\right)\right]+\sum_{j=1}^{M}\mathbb{E}_{(\tilde{\mathbf{X}},\tilde{\mathbf{A}})}\left[\log\left(1-\mathcal{D}\left(\vec{\tilde{h}}_j,\vec{s}\right)\right)\right]\right)$$

  - Based on MINE[Hjelm et al. (2018)] which has noise-contrastive objective with binary cross-entropy loss between the samples from the joint and marginal distribution: in this case positive sample vs negative sample
  - Maximize MI between $\vec{h}_i$ and $\vec{s}$, based on Jensen Shannon divergence between the joint and the product of marginals

$$I(X;Y) = \sum_{y\in Y}\sum_{x\in X} p(x,y)\log(\frac{p(x,y)}{p(x)p(y)}) = D_{JS}(p(x,y)||p(x)p(y))$$

Experiment

Table 1: Summary of the datasets used in our experiments.

Dataset	Task	Nodes	Edges	Features	Classes	Train/Val/Test Nodes
Cora	Transductive	2,708	5,429	1,433	7	140/500/1,000
Citeseer	Transductive	3,327	4,732	3,703	6	120/500/1,000
Pubmed	Transductive	19,717	44,338	500	3	60/500/1,000
Reddit	Inductive	231,443	11,606,919	602	41	151,708/23,699/55,334
PPI	Inductive	56,944 (24 graphs)	818,716	50	121 (multilbl.)	44,906/6,514/5,524 (20/2/2 graphs)

- Dataset for the experiment
  - Cora, Citeseer, Pubmed for node classification task
  - Reddit, PPI for inductive task

Experiment

- Transductive learning
  - For transductive dataset; Cora, Citeseer, Pubmed
  - One layer GCN model with following propagation rule,

$$\mathcal{E}(\mathbf{X}, \mathbf{A}) = \sigma\left(\hat{\mathbf{D}}^{-\frac{1}{2}}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-\frac{1}{2}}\mathbf{X}\mathbf{\Theta}\right)$$

  - In transductive setting, corruption function only corrupts node feature matrix, leaving adjacency matrix identical
- Inductive learning on large graph
  - In inductive learning since number of nodes are not fixed, they apply the mean-pooling rule from GraphSage

$$\mathrm{MP}(\mathbf{X}, \mathbf{A}) = \hat{\mathbf{D}}^{-1}\hat{\mathbf{A}}\mathbf{X}\mathbf{\Theta} \qquad \widetilde{\mathrm{MP}}(\mathbf{X}, \mathbf{A}) = \sigma\left(\mathbf{X}\mathbf{\Theta}' \| \mathrm{MP}(\mathbf{X}, \mathbf{A})\right) \qquad \mathcal{E}(\mathbf{X}, \mathbf{A}) = \widetilde{\mathrm{MP}}_3(\widetilde{\mathrm{MP}}_2(\widetilde{\mathrm{MP}}_1(\mathbf{X}, \mathbf{A}), \mathbf{A}), \mathbf{A})$$

  - Use 3 layer mean pooling with skip connection
- Inductive learning on multiple graphs
  - Randomly select different graph with dropout ratio for corrupted graph

Experiment

Transductive

Available data	Method	Cora	Citeseer	Pubmed
X	Raw features	47.9 ± 0.4%	49.3 ± 0.2%	69.1 ± 0.3%
A, **Y**	LP (Zhu et al., 2003)	68.0%	45.3%	63.0%
A	DeepWalk (Perozzi et al., 2014)	67.2%	43.2%	65.3%
X, **A**	DeepWalk + features	70.7 ± 0.6%	51.4 ± 0.5%	74.3 ± 0.9%
X, **A**	Random-Init (ours)	69.3 ± 1.4%	61.9 ± 1.6%	69.6 ± 1.9%
X, **A**	**DGI** (ours)	**82.3** ± 0.6%	**71.8** ± 0.7%	**76.8** ± 0.6%
X, **A**, **Y**	GCN (Kipf & Welling, 2016a)	81.5%	70.3%	79.0%
X, **A**, **Y**	Planetoid (Yang et al., 2016)	75.7%	64.7%	77.2%

Inductive

Available data	Method	Reddit	PPI
X	Raw features	0.585	0.422
A	DeepWalk (Perozzi et al., 2014)	0.324	—
X, **A**	DeepWalk + features	0.691	—
X, **A**	GraphSAGE-GCN (Hamilton et al., 2017a)	0.908	0.465
X, **A**	GraphSAGE-mean (Hamilton et al., 2017a)	0.897	0.486
X, **A**	GraphSAGE-LSTM (Hamilton et al., 2017a)	0.907	0.482
X, **A**	GraphSAGE-pool (Hamilton et al., 2017a)	0.892	0.502
X, **A**	Random-Init (ours)	0.933 ± 0.001	0.626 ± 0.002
X, **A**	**DGI** (ours)	**0.940** ± 0.001	**0.638** ± 0.002
X, **A**, **Y**	FastGCN (Chen et al., 2018)	0.937	—
X, **A**, **Y**	Avg. pooling (Zhang et al., 2018)	0.958 ± 0.001	0.969 ± 0.002

- DGI allows every node to have access to structural properties of the entire graph, whereas supervised GCN is limited to only two-layer neighborhoods

Open Source Code

- Link
  - https://pytorch-geometric.readthedocs.io/en/latest/_modules/torch_geometric/nn/models/deep_graph_infomax.html#DeepGraphInfomax

- Experiment Code
  - https://github.com/snubeaver/pytorch_geo/blob/master/examples/infomax.py

Open Source Code Review

```
def __init__(self, hidden_channels, encoder, summary, corruption):
    super(DeepGraphInfomax, self).__init__()
    self.hidden_channels = hidden_channels
    self.encoder = encoder
    self.summary = summary
    self.corruption = corruption

    self.weight = Parameter(torch.Tensor(hidden_channels, hidden_channels))

    self.reset_parameters()
```

- Init 함수에서는 encode함수와 summary, corruption 함수를 선언하고 학습가능한 weight를 만든다.
- 인코더는 1개의 gcn 레이어에 이어서 PReLU 활성함수를 통과하도록 선언한다.
- Summary 함수는 인코더를 통과한 z를 column wise mean을 한 뒤 sigmoid결과값을 출력한다.
- Corruption 함수는 edge_index는 같은것을 사용하고 노드 feature 벡터만 순서를 바꾸어 전달한다.

Open Source Code Review

```python
def forward(self, *args, **kwargs):
    """Returns the latent space for the input arguments, their
    corruptions and their summary representation."""
    pos_z = self.encoder(*args, **kwargs)
    cor = self.corruption(*args, **kwargs)
    cor = cor if isinstance(cor, tuple) else (cor, )
    neg_z = self.encoder(*cor)
    summary = self.summary(pos_z, *args, **kwargs)
    return pos_z, neg_z, summary
```

- 먼저 입력된 노드벡터와 인접행렬을 바탕으로 gcn 인코더를 통과한다. 결과값 z는 이후 summary함수에 들어가 글로벌 벡터인 summary를 만드는데 사용된다.
- 이후 corruption함수를 통해 shuffle된 노드벡터를 받아온다. 이 역시 인코더를 통과시켜 negative sample인 neg_z를 만들어준다.
- 세개의 벡터값을 반환한다

Open Source Code Review

```python
def loss(self, pos_z, neg_z, summary):
    r"""Computes the mutal information maximization objective."""
    pos_loss = -torch.log(
        self.discriminate(pos_z, summary, sigmoid=True) + EPS).mean()
    neg_loss = -torch.log(
        1 - self.discriminate(neg_z, summary, sigmoid=True) + EPS).mean()

    return pos_loss + neg_loss
```

- 앞서 forward에서 나온 세개의 벡터는 loss함수에 투입된다. 이때 로스는 분별기 함수인 discriminator의 결과값인 0,1에 대해 binary cross entropy로 만들어진다.

```python
def discriminate(self, z, summary, sigmoid=True):
    r"""Given the patch-summary pair :obj:`z` and :obj:`summary`, computes
    the probability scores assigned to this patch-summary pair.

    Args:
        z (Tensor): The latent space.
        sigmoid (bool, optional): If set to :obj:`False`, does not apply
            the logistic sigmoid function to the output.
            (default: :obj:`True`)
    """
    value = torch.matmul(z, torch.matmul(self.weight, summary))
    return torch.sigmoid(value) if sigmoid else value
```

- 분별기는 샘플된 z 벡터와 학습가능한 weight 그리고 summary 벡터를 곱해준 값을 sigmoid함으로써 0,1값으로 출력해 준다.

Open Source Code Review

```
def train():
    model.train()
    optimizer.zero_grad()
    pos_z, neg_z, summary = model(data.x, data.edge_index)
    loss = model.loss(pos_z, neg_z, summary)
    loss.backward()
    optimizer.step()
    return loss.item()
```

- 학습은 모델에 학습하고자 하는 데이터의 노드벡터와 인접행렬 정보를 넘겨주고 얻은 손실을 backward()해줌으로써 진행된다.
- 실험 세팅은Validation 셋을 통해 가장 좋은 epoch를 선정했으며 early stopping을 추가해 train loss가 감소할 때 학습을 중단했다. Early stopping은 epoch가 절반 이상 지나간 이후부터 moving window K개의 trian_loss의 평균보다 마지막 train_loss가 높아질 경우 학습을 중단하도록 설정되었다.

Reproduced Experiment using open-source

- Cora Dataset
  - 300 epochs, latent dimension=512, Adam optimizer
  - Accuracy: 0.8190
- Citeseer Dataset
  - 400 epochs, latent dimension=512, Adam optimizer
  - Accuracy: 0.7220
- Pubmed Dataset
  - 400 epochs, latent dimension=512, Adam optimizer
  - Accuracy: 0.7390

Q&A