수치해석 GNN 프로젝트 최종 보고서

팀원1: 융합전자공학부 2018006153 권도혁

팀원2: 융합전자공학부 2019021058 공민경

1. 문제 정의

프로젝트 주제는 그래프를 이용한 머신러닝으로 분자 구조에 대한 데이터를 학습시켜 쌍극자 모먼트(μ)를 예측하는 것입니다. 사용된 원소집합은 {C, H, N, O, F}이고, 각 원자번호 {6, 1, 7, 8, 9}를 가집니다. 주어진 데이터는 분자구조를 문자열로 나타낸 SMILES 표기, 등방향 분극률, XYZ 좌표 등의 정보를 제공합니다. 이러한 데이터를 사용해 분자 구조를 그래프 구조로 만들어 타겟값인 쌍극자 모먼트를 포함하여 학습하고, 학습이 끝난 모델을 이용해 각 분자에 대한 쌍극자 모먼트를 예측하고, 이 결과가 잘 반영되었는지 검증합니다.

2. 초기 문제 접근 방법

우선 초기 학습 모델은 GNN의 기본 형태로 수업시간에 가장 중요하게 다루어졌던 Message Passing을 이용한 Neural Network model을 생각하였습니다. 마침 수업시간 실습(Lab3)에서 MPNN Layer를 구현하였고 보다 쉽게 프로젝트를 진행할 수 있었습니다. 조교님께서 제공해주신 코드를 통해 mol 데이터를 csv로 변화하여 각 train_list와 test_list를 생성하고, MPNN을 이용하여 그래프 구조를 설계하여 학습하는 방향으로 진행했습니다. Pytorch geometric이 제공하는 MessagePassing 클래스를 이용하고, MyNet 클래스를 형성하여 torch에서 제공하는 Module을 이용해 Neural Network 학습 환경을 구축했습니다.

$$x_i^{(k)} = \gamma^{(k)}\left(x_i^{(k-1)}, \square_{j \in N_{(i)}} \phi^{(k)}\left(x_i^{(k-1)}, x_j^{(k-1)}, e_{j,i}\right)\right)$$

PyG의 MessagePassing 클래스를 이용하면, 위의 수식에서 메시지 전파를 자동으로 처리해주기 때문에 위 식에서 Φ message()와 γ update() 함수만 따로 구현하였습니다.

```
class MyNet(torch.nn.Module):
    def __init__(self):
        super(MyNet, self).__init__()
        self.mpnn1 = MPNNLayer(1, 16)
        self.mpnn2 = MPNNLayer(16, 16)
        self.mpnn3 = MPNNLayer(16, 1)
        self.output = nn.MaxPool1d(100, 100)

    def forward(self, data):
        x, edge_index, edge_attr = data.x, data.edge_index, data.edge_attr

        x = self.mpnn1(x, edge_index, edge_attr)
        x = self.mpnn2(x, edge_index, edge_attr)
        x = self.mpnn3(x, edge_index, edge_attr)
        x = x.flatten()

        return torch.mean(x).reshape(1)
```

본 Project의 task는 Regression task 이므로 마지막 MPNN Layer의 출력 채널을 1개로 설정하고, 스칼라 값을 출력하기 위해 flatten() 메서드로 생성된 one-dimensional vector에 torch.mean을 취하고 reshape() 함수를 사용하여 스칼라 값을 반환했습니다. Output vector 전체를 커버하는 사이즈(100으로 설정)로 Max pooling을 이용해보기도 하였으나(위 사진의 self.output layer), 성능 향상에 어려움이 있어 torch.mean을 이용하게 되었습니다. 그 후 모델 학습 최적화를 위해 epoch 값을 증가시키며 score를 꾸준히 확인하였습니다. 약 150회 이상부터는 overfitting이 발생하여 성능이 감소하였고, epoch값을 줄이고 aggregation() 함수를 추가하여 최적화하고자 하였으나, aggregation() 추가는 overfitting 정도가 더 심해지는 것을 확인할 수 있었습니다.

(위 모델 + epoch 150회 Score: 0.91624)

이후 수업시간에 학습하였던 GCN 구현에도 도전하였으나, pytorch 문법 오류 등으로 완성을 시키지 못하였고, 조교님께서 올려주신 DimeNet 논문을 읽고 새로운 방향을 잡게 되었습니다.

3. 학습 GNN 모델: DimeNet

처음 사용한 학습 모델은 GNN의 대표적인 프레임워크인 MPNN입니다. GNN 모델을 이용한 분자 예측에서 그래프는 하나의 분자, 노드는 원자, 엣지는 미리 정의된 분자 구조 그래프 또는 원자 간의 거리로 결정됩니다. GNN의 메시지는 다음과 같이 정의됩니다. $m_{j\to i} = \phi^{(k)}\left(\boldsymbol{x_i^{(k-1)}}, \boldsymbol{x_j^{(k-1)}}, \boldsymbol{e_{j,i}}\right)$ 즉, 메시지 함수는 타겟 노드의 현재 상태값, 타겟 노드의 이웃의 현재 상태, 그리고 해당 노드와 이웃을 연결하는 엣지의 정보를 합하여 타겟 노드의 다음 메시지로 전달합니다.

MPNN에서 메세지 함수와 업데이트 함수는 아래의 수식으로 정의됩니다. $x_i^{(k-1)}$는 $(k-1)$ 레이어의 노드 i의 특성을 나타내고, □는 aggregation 함수, $e_{j,i} \in \mathbb{R}^D$ 노드 j에서 노드 i의 노드 특성을 나타냅니다. MPNN은 회귀(Regression)를 이용해 업데이트 합니다.

$$x_i^{(k)} = \gamma^{(k)}\left(x_i^{(k-1)}, \quad_{j\in\mathcal{N}(i)}\phi^{(k)}\left(x_i^{(k-1)}, x_j^{(k-1)}, e_{j,i}\right)\right)$$

따라서 일반적인 non-directional GNN은 분자 예측을 할 때 효과적인 구조를 가지지만, GNN의 업데이트 메세지는 이전 임베딩 원자와 원자쌍 거리 정보로만 구성되고, 분자 예측에 중요한 방향 정보인 결합각, 회전 등에는 독립된 특성을 가집니다.

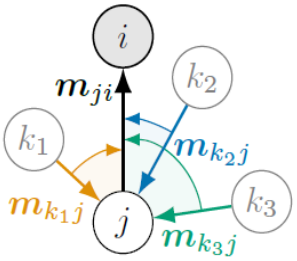
또한, GNN은 전체 원자 간 거리에 대한 정보를 담은 행렬을 사용하지 않고 cutoff 거리 c를 사용합니다. 이 경우, 결합 길이와 이웃 원소가 같은 분자간의 cutoff 거리가 2.5Å보다 작거나 같다면, GNN은 이 분자가 어떤 것인지 구분하지 못합니다. GNN(MPNN) 모델의 이러한 한계점을 극복하고자, molecular data의 regression task에서 높은 성능을 가지는 모델을 찾게 되었고, 그 중에서도 조교님께서 올려주신 논문 중 가장 성능이 좋았던 DimeNet을 사용하였습니다.

DimeNet은 기존 GNN의 문제점을 보완하기 위해 방향 정보, 결합 사이의 각도를 담은 메세지를 원자 사이에 임베딩하고, 이 메세지를 이용하여 신뢰전파 알고리즘 같이 관측된 노드를 바탕으로

관측되지 않은 노드의 분포를 계산하여 업데이트합니다.

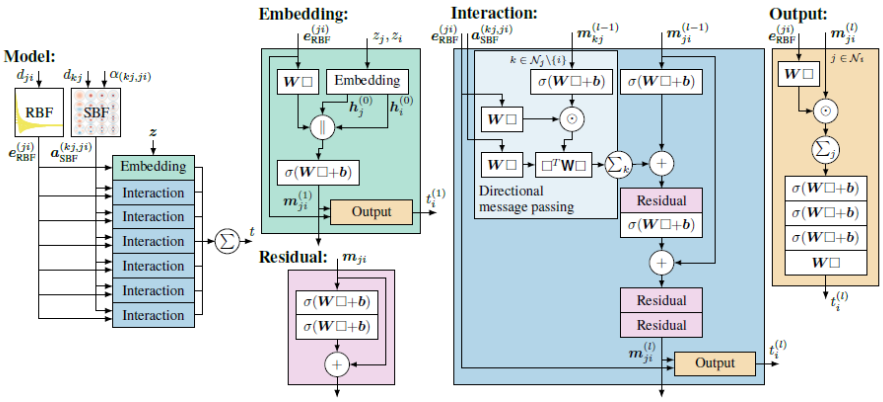
DimeNet 메세지의 핵심을 방향 임베딩, 메시지 임베딩으로 나누었습니다. 먼저, 방향성을 가진 임베딩이 가능한 것은 원소의 물리적 특성값은 회전에 불변하기 때문입니다. 원자가 회전을 해도 이웃 간 거리와 결합각이 변하지 않는 것을 이용하여 차단거리 내의 원자들이 서로 반응할 때만 불변성이 깨지는 경우를 모델에 추가적으로 반영하면 됩니다. DimeNet은 원자 i와 이웃 노드 j에 대해 각 인접한 원자의 방향으로 동일하게 학습하는 별도의 임베딩 m_{ji}를 만들어 구현했습니다. 임베딩 m_{ji}는 분자와 함께 회전하므로 이웃 노드와의 상대적인 방향 정보가 보존 가능합니다.

메시지 임베딩은 원자 쌍 ji에 대한 방향 임베딩 m_{ji}를 원자 j에서 원자 i로 보낸 메세지로 생각할 수 있습니다. 임의의 원자 i에 대해 m_{ji} 메세지 집합을 이용해 임베딩하고 메세지 m_{ji}는 이웃 노드들에서 들어오는 방향의 메세지로 업데이트 합니다. 따라서 메세지는 업데이트 함수와 aggregation을 이용해 다음과 같이 정의됩니다.



$$m_{ji}^{(l+1)} = f_{update}\left(m_{ji}^{(l)}, \sum_{k \in N_j \setminus \{i\}} f_{int}\left(m_{kj}^{(l)}, e_{RBF}^{(ji)}, a_{SBF}^{(kj,ji)}\right)\right)$$

$a_{SBF}(Spherical\ Bessel\ Function)$는 원자 결합각 $\angle kji$와 k와 j사이의 원자간 거리 d_{kj}를 SBF로 임베딩 한 값입니다. 또한, $e_{RBF}^{(ji)}(Radial\ Bessel\ Function)$는 원자간 거리 d_{ji}의 방사형 기저 함수 임베딩 값을 나타냅니다.



DimeNet의 전반적인 학습과정은 위의 아키텍처 그림을 통해 확인할 수 있습니다. 크게는 위에서 언급한 RBF와 SBF를 통해 원자간 거리와 각도 정보를 임베딩하고, 임베딩 된 값과 메세지 m_{kj}를 Embedding, Interaction block에 넣어 나오는 값을 Aggregation하여 출력합니다. 이 때 Embedding block에서는 중심 원자와 주변 원자간 거리에 대한 정보를 담은 RBF 임베딩 값과 원자 종류를 이용하여 출력값을 형성하고, Interaction block에서는 RBF와 SBF 임베딩 값 모두와 이전 layer로부터 전달된 message값을 이용하여 출력값을 형성합니다. 이렇게 한 개의 Embedding block과 여러 개의 Interaction block에서 나온 출력값들의 aggregation이 최종 Output이 됩니다.

4. 실험 결과 비교 분석

실험에 쓰인 모델은 다음과 같이 설정하였습니다.

1. Model baseline: Python Geometric class
2. Model initial parameter:
    a. hidden_channels=128 -> 많이 사용하는 임베딩 사이즈 이용
    b. out_channels=1 -> 예측값 1개
    c. num_blocks=6 -> Building block 수
    d. num_bilinear=8 -> Bilinear tensor 수
    e. num_spherical=7 -> spherical harmonics 수
    f. num_radial=6 -> radial basis function 수
3. Loss function: torch.nn.L1Loss() -> MAE
4. Optimizer: ADAM (learning rate: 1e-5, amsgrad 이용)
5. Scheduler: Exponential learning rate decay (3000 step 당 0.98)
6. Stochastic moving average (step 당 0.001)
7. 학습 Epoch 수: 10

해당 설정 값들은 논문[1] Appendix B (Experimental setup)에 작성된 QM9 dataset 성능 분석에 활용한 hyperparameter값을 그대로 이용하되, dataset size의 차이로 인해 일부만 수정한 값입니다.

또한 DimeNet 클래스에 포함된 from_qm9_pretrained 메서드를 이용하여 QM9 dataset에서 우수한 성능을 보인 모델의 파라미터를 받아 학습을 시작하였습니다.

추가로, data 중 x 값 (원자 넘버링)에 H를 0으로 표시하였다는 점이 학습에 방해가 된다고 판단하여 모든 값에 1을 더하여 학습하였습니다. 이는 실제로 점수 상승에 효과가 있었습니다.

최종 학습이 완료된 모델의 MAE score는 0,00712이었습니다.

5. 토의

본 프로젝트에서 다른 팀들에 비해 report한 Loss값이 매우 낮았습니다. 가장 큰 영향을 준 부분은 본 프로젝트의 평가 data가 qm9 dataset의 subset이었는데, 이를 이미 학습을 완료한 pretrained model을 사용하였기 때문입니다. Qm9이 평가 data와 다른 data라도 molecular를 학습하고 모먼트값을 예측하는 task에 대해 학습한 경험이 있는 pretrained model을 이용하면 성능이 좋을 것으로 예측하고 해당 메서드를 사용하게 되었습니다. 그러나 data 구성과 그 형식이 달라 평가 data가 qm9의 subset data인 것은 인지하지 못하고 test data로 pretrain 된 model을 사용하

[1] J.Gasteiger et al. 'Directional Message Passing for Molecular Graphs', ICLR, 2020

게 되었습니다. 이러한 점에서 leaderboard에 report 된 점수는 올바르게 학습 및 평가된 모델의 점수라고 볼 수 없습니다.

그러나 본 프로젝트를 수행하면서 여러 논문을 읽고, GNN에서 Message passing에 다양한 사람들의 아이디어가 들어가있는 것을 확인하였습니다. 더하여, DimeNet과 같이 domain knowledge가 많이 쓰인 아이디어는 곧 해당 data에서 매우 높은 성능의 실험 결과를 얻을 수 있게 됨을 알 수 있었습니다. 추가로 DimeNet 논문에 작성된 학습 세팅 방법을 이용하기 위해 EMA, SWA 등의 다양한 scheduling 기법에 대해 학습할 수 있었습니다. 이는 후에 꼭 GNN이 아니더라도 learning model을 제작할 때 유용하게 사용할 수 있을 것입니다.

Data 가공부터 학습 모델 선정, loss function 및 optimizer 선택, hyperparameter optimization, validation 및 test까지 전 과정을 직접 구현해보고, 레퍼런스를 찾아보며 결과를 얻기 위해 노력한 과정을 통해 매우 많은 것을 얻을 수 있었던 프로젝트였습니다.

6. 팀원의 역할

프로젝트 시작에 앞서, 강의 중 배운 이론을 바탕으로 각자 자료조사를 진행하게 되었고, 팀원1이 MPNN을 이용한 그래프 학습에 대한 아이디어로 초기 모델을 구현하고 epoch 값을 증가시키며 최적화를 하여 중간 제출을 완료했습니다. 그 후, 팀원2가 epoch값 조정과 aggregation함수를 이용해 최적화하는 시도를 했습니다. 최종적인 모델로 DimeNet을 이용하여 구현하기로 결정 후, 팀원1이 코드를 완성하여 모델 학습을 완료시키고, 팀원2가 DimeNet 이론에 대한 자료조사 및 성능에 대한 이론적 분석을 진행했습니다. 각자 맡은 부분에 대한 PPT와 발표 대본을 만들고 발표를 마쳤습니다.