**GCN 발표**

제가 이번 주 발표하는 논문은 GCN : semi-supervised classification with GCN 입니다. 이 논문은 2017년 ICLR 발표되었습니다.

GCN은 기존에 image에 적용하던 CNN 기법을 graph로 확장시킨 방법입니다.

CNN은 convolution 연산을 이용해서 이미지의주변 픽셀들을 연결시켜서 feature를 뽑아내는 방법입니다.

GCN 또한 convolution을 통해서 연결되어있는 노드들간의 정보를 이용하여 학습하는 network입니다.

이 논문은 일부만 라벨링이 된 graph의 node들을 분류하는 task를 다룹니다. 이 task는 결국 이 논문의 제목인 graph-based semi-supervised learning으로 귀결되게 되는데

기존의 이 문제에 대한 접근은 graph Laplicain regularization을 loss function에 사용하는 법이었습니다.

Loss function 왼쪽은 supervised set에 대한 loss 오른쪽이 그외에 label이 없는 datㅁ에 대한 loss입니다.

오른쪽 식을 자세히 보게되면 연결되어있지 않은 노드들에 대해서는 adjacency matrix element 값이 0이기 때문에 loss가 0이고, 연결되어있는 두 노드에 대해서는 label 비슷하여야 loss가 작아지게 됩니다.

이렇게 loss function을 세팅한 데에는 연결되어있는 node들은 유사한 label 값을 가진다라는 가정이 있기 때문입니다.

하지만 이런 경우에는 그래프를 학습하는데에 제한을 주게 됩니다.

그래서 이 문제를 해결하고자 이 논문에서는 저 regularization을 사용하지 않는 방법으로

그래프를 neural network에 넣고 supervised loss를 통해 전체 node들을 학습시키는 방법의 GCN을 제안하고 있습니다.

Motivation

이 파트에서는 GCN에서 사용하는 propagation rule을 Graph Convolution 식에서 유도하는 과정에 대해 설명합니다. 유도 과정은 크게 3가지의 단계를 거치게 됩니다.

spectral graph convolution을 graph signal과 g filter에 대해서 적용시키면 Fourier domain에서의 곱으로 표현이 되어 첫번 째 줄의 식으로 됩니다. 여기에서 g filter를 diagonal이라는 가정을 하게 되면 convolution은 두번 째 줄의 식처럼 됩니다.

이 식은 Laplacian matrix를 eigen decompoistion을 하고 거기서 얻은 U matrix를 가지고 행렬곱을 하게되는데, 이 경우 O(N^2)보다 커지게 되어서 computation이 굉장히 커지게됩니다.

이 경우에 computation이 O(n^2)이 되기 때문에 연산이 느리고 expensive 하게 됩니다. 그렇기 때문에 이 식을 chebyshev polynomial를 이용하여 근사치를 사용합니다. 그래서 기존의 식은 이렇게 다항식의 꼴로 변환이 됩니다.

그리고 여기서는 K=1로 설정하여서 graph laplacian spectrum에 대해 linear하게 function을 만듭니다. 그리고 parameter의 갯수가 많아지면 학습시 overfitting이 될 수 있어서 theta를 이와 같이 설정해주어서 parameter의 갯수를 줄였습니다.

그럼 convolution 식은 시그마 다항식에서 1차 식으로 정리가되고, parameter 설정을 적용하고 람다 max를 2로 근사하게 되면 normalized L로 되어 최종적으로 아래처럼 정리되게 됩니다.

\*\*\* 왜 gradient 문제가 생기는지?

근데 여기까지의 식은 exploiting or vanishing gradient 이슈가 생길 수 있기 때문에 renormalization을 한 번 해주게 됩니다. 그래서 최종적으로 convolution은 아래와 같이 근사된 식으로 나타게 됩니다.

이를 바탕으로 node classification을 위해 실제 two layer GCN을 구현하게되면 다음과 같습니다. A hat 행렬을 미리 연산하고 이를 propagation rule에 적용하면 모델은 다음과 같이 간단한 forwarding model로 표현할 수 있습니다. 그리고 학습을 위해 cross-entropy loss function을 이용하였습니다.

이 논문에서 실험은 2가지가 진행되었는데, 첫번 째 실험은 node classification task에 대해서 GCN의 성능을 기존의 다른 방법들을 baseline으로 하여 확인하는 실험.

그리고 GCN의 propagation 모델을 바꾸어가면서 제안한 propagation rule이 우수한 것을 보여주었습니다.

실험에서는 총 4개의 데이터 셋을 사용하였고 3개는 citation networks, 1개는 Knowledge graph를 사용했습니다.

첫 번째 실험결과를 보게되면 그 때 당시 sota 모델이었던 Planetoid보다 정확도가 높았고, 학습 시간도 훨씬 짧게 나왔습니다.

두 번째 실험결과에서는 chebyshev를 1차가 아닌, 2차, 3차로 한 경우, renormalization 하기 전 식들을 기반으로 한 경우로 실험을 돌렸고, 역시 1차 chebyshev를 renormalization 한 식이 제일 성능을 높았음을 증명하였습니다.

이 논문에서 제시한 limitation과 future works는 세가지 입니다.

첫번째는 GCN은 full batch로 전체 그래프가 한번에 학습되기 때문에 메모리 요구량이 사이즈에 비례해서 증가하게 되는 문제가 있어서 GCN을 mini batch로 진행할 수 있도록 하여야 합니다.

두 번째는 GCN에서는 graph의 edge direction이라던지 representation에 대한 정보를 고려하지 않고 있습니다. 이 부분에 대해서도 하나의 limitation 이라고 얘기하고 있습니다.

세 번째는 propagation rule을 유도할 때 self connection과 neighbor node connection의 가중치가 같다고 하였는데, 이 둘 사이에 가중치를 두어 학습을 통해 조절하는 방법을 제시하였습니다.

여기까지가 논문의 본문이고

이 슬라이드는 제가 직접 구현한 GCN 모듈입니다.

이 논문의 핵심은 GCN 구조를 구현하고 데이터에 적용해보는 것이라고 생각해서 모델을 만들고 이를 Cora dataset에 적용해보는 것을 구현을 마무리했습니다.

저는 torch library와 adamW optimizer를 이용했고 그 외에는 논문의 값들과 같게 진행하였습니다.

그 결과 accuracy가 0.8150의 결과를 얻게 되었습니다.

이상으로 발표를 마치겠습니다.