GRAND : Graph Neural Diffusion

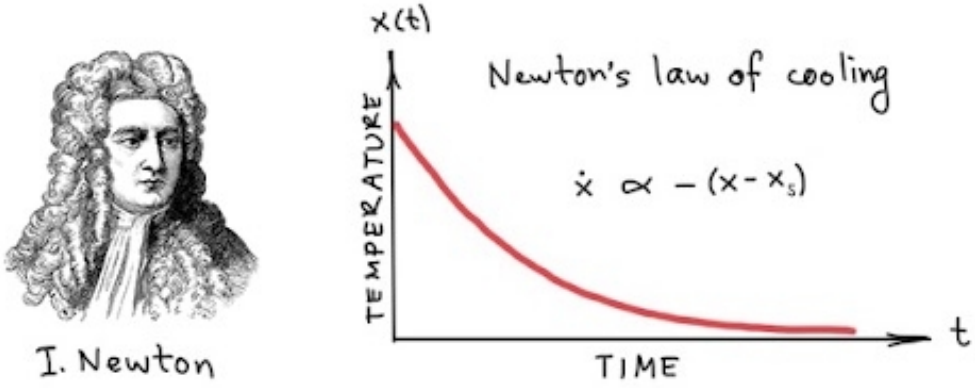
Twitter tech blog 를 참고하였습니다.

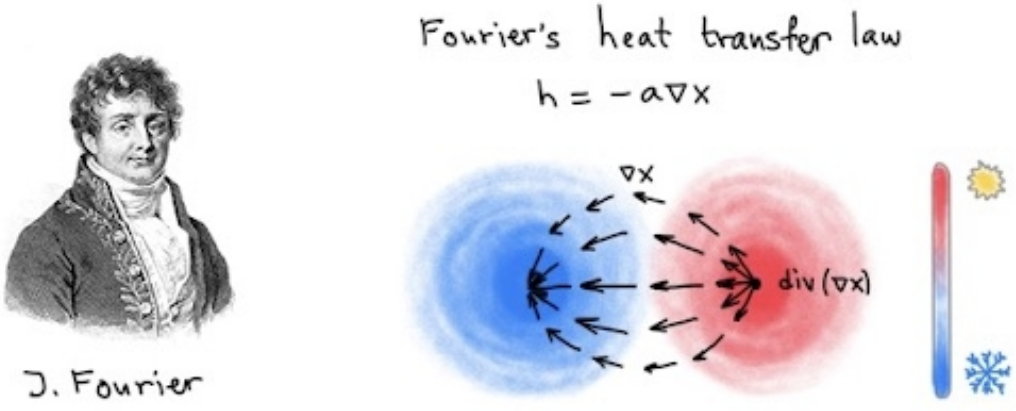
- 트위터에서 엄청난 크기의 데이터의 핵심 이점은 사람들이 어떻게 서로 상호작용 하는지 (트윗, 팔로우, 대화 등)를 설명하는 huge graphs라는 것입니다. 트위터에서는 이런 그래프를 multimeida content와 연결하고 이해하며, 어떻게 상호작용할 것 인지를 풀어내는 것이 주된 ML challenges라고 합니다.
- GNN은 message passing operation과 MLP의 이점을 결합하여 작동합니다. Message passing은 그래프에서 nodes간의 정보를 교환할 수 있도록 해줍니다.
- Message passing은 diffusion의 form을 갖추고 있고 그래서 GNNs은 diffusion을 설명하는 differential equations 과 관련이 있습니다.
- GNNs을 discrete partial differential equations (PDEs)로 생각하는 것은 새로운 class의 architectures를 생각해 볼 수 있게 해줍니다.
- 이러한 architectures들로 graph ML에서 issues 인 depth, over smoothing, bottlenecks and graph rewiring 문제들을 다룰 수 있습니다.
- Newton's law of cooling에서 heat diffusion equation (열 확산 방정식)이 나올 수 있는데, 가장 간단한 표현 식이 Partial Differential Equation (PED) 입니다.

  $\dot{x} = a\Delta x.$
- 여기서 x(u,t) 는 time t 이고 position u 일때 온도를 나타냅니다.
- $\dot{x}$ (temporal derivative)는 온도 변화 비율입니다.
- 오른쪽 term은 spatial second-order derivative 또는 Laplacian Δx 입니다. 이는 한 지점의 온도와 그 주변의 지역적 차이를 나타냅니다.
- a 는 termal diffusivity 라는 coefficient입니다. a가 상수이면 PDE는 선형이고 이 식의 해는 time-dependent Gaussian kernel과 initial temperature distribution의 convolution으로 주어집니다.

  $x(u,t) = x(u,0) * exp(-|u|^2/4t).$
- 조금 더 일반화하면, thermal diffusivity 는 시간과 공간에 따라 다양하며 다음과 같은 PDE form을 가지게 됩니다.

  $\dot{x}(u,t) = div(a(u,t)\Delta x(u,t))$
-





- 위 그림과 같이 Newton's law of cooling을 따르면, "물체의 온도의 변화 비율"($\dot{x}$) 은 그것의 온도와 그 주변의 온도 차이의 비율입니다. 이 solution의 결과는 지수적으로 감소하는 경향을 보입니다.
- Fourier's heat transfer law는 좀 더 세분화된 local model입니다.
  - 온도는 scalar field x 입니다.
  - Scalar filed의 negative gradient는 vector field $-\Delta x$ 입니다. 이는 더 뜨거운 영역에서 더 차가운 영역으로의 열 이동을 표현합니다.
  - divergence div($-\Delta x$) 는 vector field $-\Delta x$ 의 net flow로 어떤 point 주변의 극소 영역을 통과합니다.
- Diffusion PDE는 어떤 에너지나 matter 좀 더 추상적으론 정보의 transfer가 포함된 다양한 물리적 프로세스에서 많이 일어납니다.
- 이미지 처리에서 이런 설명은 low-pass filtering에서 나타납니다. 이러한 필터는 noise를 제거할때, 의도치 않게 edges 영역을 블러하게 만드는 효과가 있습니다.
- Petro Perona 와 Jitendra Malik는 adaptive diffusivity coefficient를 생각했습니다. 이는 image gradient norm $|\nabla x|$ 에 inversely dependent 하게 됩니다.
- 이 방법은 "flat" 한 영역에서는 diffusion이 강하고 ($|\nabla x| \approx 0$), 밝기 변화가 불연속적인 곳에서는 약합니다.($|\nabla x|$ 가 큰 경우)
- 그 결과 nonlinear filter 는 edges 영역은 보존하면서 noise를 지울 수 있습니다.

> Left: original image of Sir Isaac Newton, middle: Gaussian filter, right: nonlinear adaptive diffusion.

- Perona-Malik diffusion과 그리고 유사한 schemes 는 다양한 영역에서 발전, 적용 되었습니다.
- 오랜시간 variational and PDE-based methods가 영상처리와 컴퓨터 비전 영역에서 거의 20년을 지배해오다가 2000년에 들어서면서 딥러닝에게 그 자리를 내주게 됩니다.

Diffusion equations on graphs

- GNNs는 근접 노드간의 정보 전달을 메세지 패싱의 형태로 작동하며 이는 diffusion과 동일한 컨셉의 프로세스입니다.
- 이 경우 base space는 그래프이고 diffusion은 edges를 통해 이루어집니다. 여기서 공간 도함수(spatial derivatives)의 유추(analogy)는 근점 노드 features의 차이입니다.
- graphs에서 diffusion process의 일반적인 형태는 다음과 같습니다.

  $\dot{X} = div(A(X(t))\nabla X(t))$

  - X(t) 는 n x d 행렬로, time t 에서 node features입니다.
  - gradient 는 edge 별로 각각의 node feature vectors의 difference를 할당하는 operator입니다.

    $(\nabla X)_{uv} = x_v - x_u$
  - node v에서 divergence는 v에서 나온 edges의 features를 다 더합니다.

    $\sum_v W_{uv} X_{uv}$
- 확산율은 A(x)=diag(a(x_u, x_V)) 형태의 matrix-valued function에 의해 표현됩니다. 여기서 이전과 마찬가지로 a 는 x_u, x_v features의 유사도에 기반한 edge u ~ v 사이의 diffusion의 강도를 정의하는 함수 입니다.
- graph diffusion equation은 matrix differential equation의 형태로 쉽게 적어볼 수 있습니다.

  $\dot{x}(t) = (A(X(t)) - I)X(t)$
- 대부분의 경우 이 differential equation은 closed-form solution을 가지지 않습니다. 그래서 numerically하게 풀어지게 됩니다. nonlinear diffusion equations을 풀 수 있는 다양한 numerical techniques가 있습니다. 이는 공간과 시간 이산화 선택에 따라 달라집니다.
- 가장 간단한 이산화 방법은 temporal derivative $\dot{X}$ 를 forward time difference로 교체하는 것입니다.

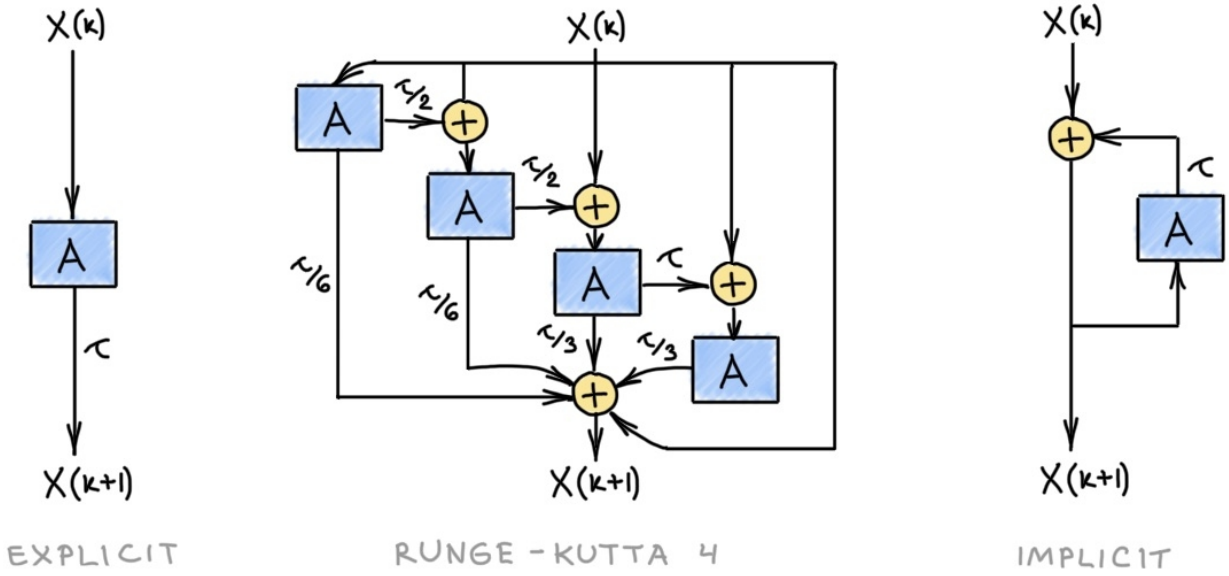
  $[X(k+1) - X(k)]/\tau = [A(X(k)) - I]X(k)$
  - 여기서 k 는 이산 시간 index (iterraiton number)입니다. 그리고 τ 는 step size 입니다. (t=kτ)
  - 위 식을 아래와 같이 정리할 수 있습니다.

    $X(k+1) = [(1-\tau)I + \tau A(X(k))]X(k) = Q(k)X(k)$

- forward Euler scheme을 얻었습니다. 여기서 다음 interation X(k+1)은 X(k)에 linear operator Q(k)를 적용하여 얻을 수 있습니다.
- 이 방법은 time step 이 충분히 작을때 (τ < 1) 수치적으로 안정적입니다.
- Temporal derivative를 이산화하기 위해서 backward time difference를 사용하는 것은 (semi-)implicit schem을 유도하게 됩니다.

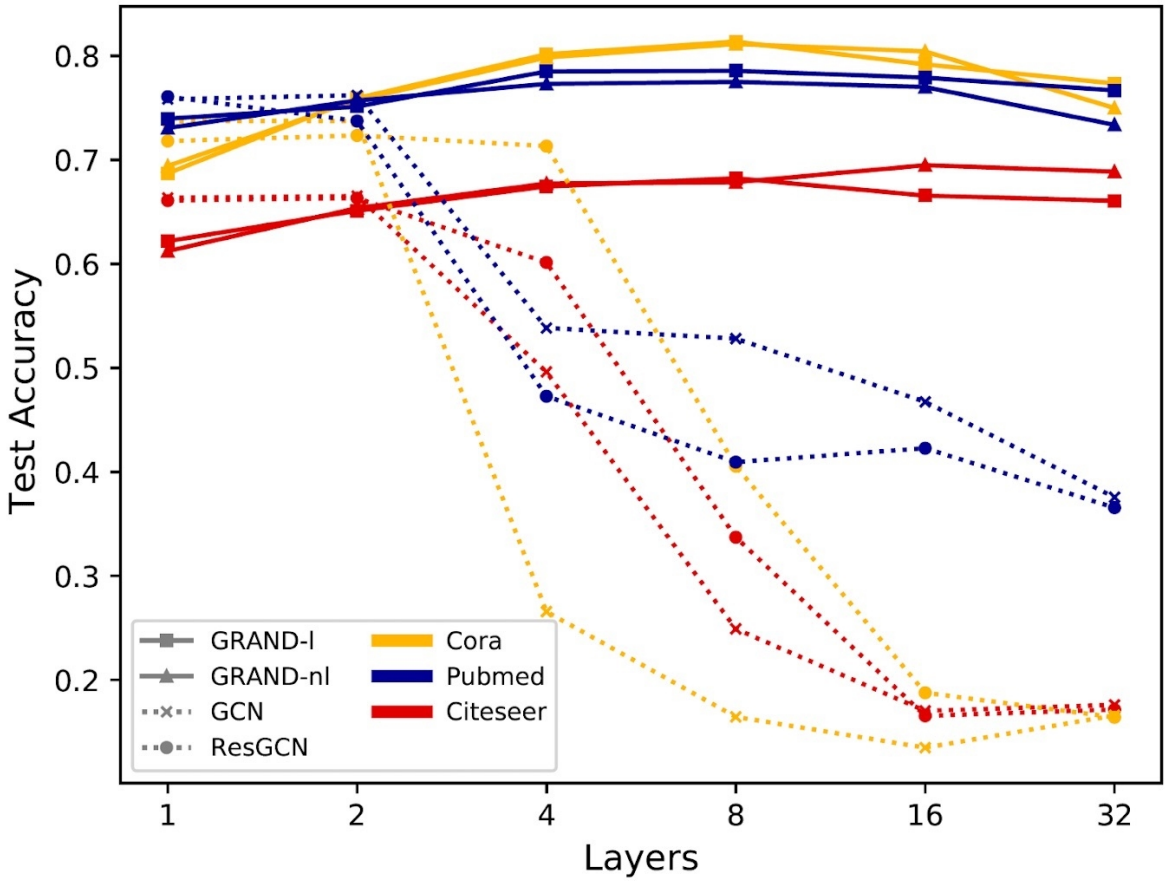
$$[(1+\tau)I - \tau A(X(k))]X(k+1) = B(k)X(k+1) = X(k)$$

  - 이 scheme이 "implicit"하다고 하는 이유는 이전 iterate에서 다음을 추론하기 위해선 B의 역에 해당하는 선형 시스템을 풀어야하기 때문입니다.
  - 이는 lienar solver의 몇 번의 반복으로 풀어집니다.
  - 이 scheme은 무조건적으로 stable하고 이는 $\tau > 0$ 이면 괜찮다는 것을 의미합니다.
- 컨셉적으로는 간단한단 이산화 기법이 실제로는 잘 먹히지는 않습니다. PDE 에서는 Runge-Kutta와 같은 multi-step schemes를 일반적으로 사용합니다. 여기서 다음 iterate는 몇 개의 이전 iterate들의 linear combinationd으로 계산됩니다.
- explicit 과 implicit인 경우 모두 multi-step으로 만들 수 있습니다. 나아가 step size도 adaptive하게 할 수 있습니다.



> Left-to-right: single step explicit Euler, multi-step fourth-order Runge-Kutta, single step implicit. A denotes the diffusion operator;τ is the time step size.

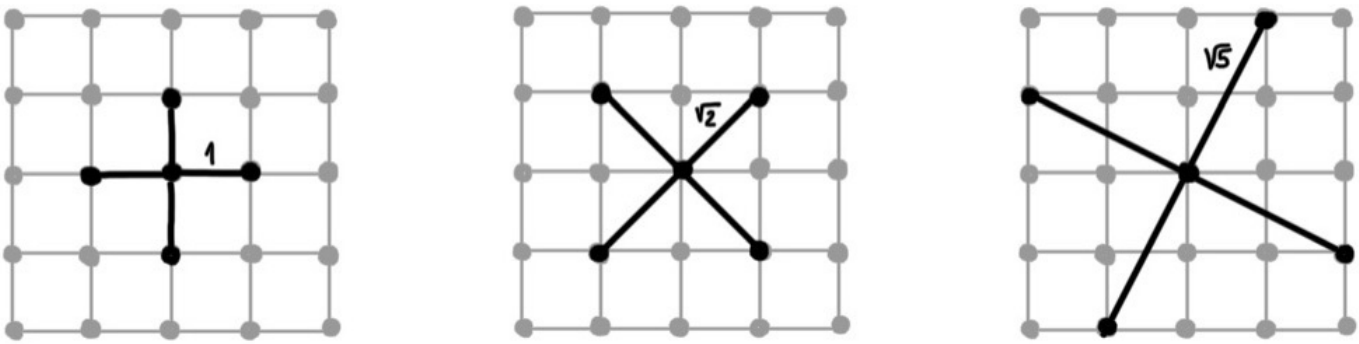
- 주어진 task에 최적화된 parametric diffusivity funciton을 가지는 diffusion equations는 GNN-like architectures의 넓은 범주를 정의합니다.
- 이를 저자들은 Graph Neural Diffusion 또는 GRAND라고 줄여서 부른다고 합니다.
- Output은 어떤 time T에서 diffusion equation의 solution X(T)입니다.
- 많은 GNN architectures는 GRAND의 한 경우로 표현될 수 있습니다. 예를 들어 위에서 봤던 explicit scheme은 GAT의 식이 될 수 있습니다. 여기서 diffusivity는 attention의 역할을 합니다.
- GNNs의 대부분은 explicit single-step Euler scheme을 사용하는데, discrete time index k 는 GNNs에서 convolutional 또는 attentional layer에 대응됩니다.
- 여러번의 diffusion 반복해서 돌리는 것은 GNN layer를 여러번 적용하는 것과 같습니다.
- GRAND는 deep GNNs에서 성능저하의 [widely recognised problem](#) 를 다룰 수 있게 합니다.



> In Graph Neural Diffusion, explicit GNN layers are replaced with the continuous analog of diffusion time. This scheme allows training deep networks without experiencing performance degradation.

- Implicit schemes는 diffusion operator의 역을 필요로 하는 반복에서 계산 복잡 존재하는데 이때 큰 time step을 사용함으로써 더 적은 반복을 할 수 있습니다.
- diffusion operator(여기서는 matrix A)는 그래프의 adjacency matrix(1-hop filter)의 구조와 같습니다. 반면의 이것의 inverse는 dense matrix 로 multi-hop filter로 해석됩니다.

- Matrix inversion의 효율성은 matrix의 구조에 매우 의존적이기 때문에, 몇몇 상황에서는 input graph로부터 diffusion을 위해 사용된 graph를 decouple 하는 것이 이점이 될 수도 있습니다.
- 이런 기술을 graph rewiring이라고 부릅니다. diffusion framework 는 그래프 rewiring을 graph를 어떤 연속적인 object의 spatial discretization(공간적 이산화)로 고려함으로써 원칙적인 관점을 제공합니다.



> Discretizations of the 2D Laplacian operator. Any convex combination of discretizations is also a valid one. Choosing a discretization is a Euclidean analogy of "graph rewiring".

Conclusion

- Graph Neural Diffusion은 원칙적인 수학적 framework를 제공하여 많은 유명한 GNNs architectures를 연구할 수 있고 새로운 architectrues의 발전을 위한 청사진을 제공합니다.
- 이러한 해석은 feature over-smoothing과 같은 GNNs의 일반적인 이슈와 DNN 디자이닝의 어려움 그리고 graph rewiringr과 같은 heuristic techniques에 새로운 관점을 제공합니다.
- 보다 광범위하게, 우리는 그래프 ML, 미분 방정식 및 기하학 간의 연결을 탐구하고 이러한 주제에 대한 방대한 문헌을 활용하면 해당 분야에서 새롭고 흥미로운 결과를 얻을 수 있다고 믿습니다.