

Semi-Supervised Classification with Graph Convolutional Neural Networks

Обзор статьи

Томинин Владислав Дмитриевич

МФТИ, Долгопрудный

1 Введение

- Постановка задачи
- Аналогия с CNN
- Устройство GCN
- Задачи

2 Выбор правила $f(H^i, A)$

- Нестрогие размышления
- Недостатки введенного правила
- Существующие подходы

3 Теоретическое обоснование

- Преобразование Фурье
- Собственные значения - аналог частоты
- Выбор фильтра из спектральных соображений
- Приближения фильтра полиномами Чебышева
- Линейное приближение полученных фильтров

4 Классификация узлов в графе

- Устройство классификатора
- Функция ошибки
- Функция потерь
- Архитектура классификатора
- Распространение ошибки



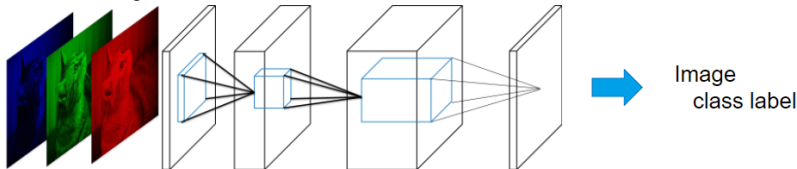
Рассмотрим граф $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ с N вершинами $v_i \in \mathcal{V}$ и ребрами $(v_i, v_j) \in \mathcal{E}$.

- $A \in \mathbb{R}^{N \times N}$ - матрица весов графа \mathcal{G}
- $D_{ii} = \sum_j A_{ij}$ - матрица степеней вершин графа
- $L = D - A$ - ненормализованный граф Лапласа (также обозначается, как Δ)
- $X \in \mathbb{R}^{N \times D}$ - матрица признаков вершин графа
- $f(\cdot)$ - дифференцируемая функция, модель кодирования структуры графа. В нашем случае $f(\cdot) = f(X, A)$

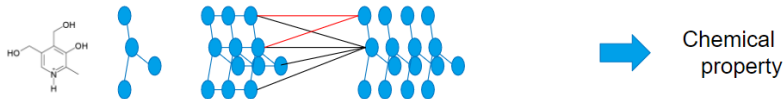
Задача - по достаточно разреженному графу и нескольким вершинам с пометками определить метки остальных вершин путем сглаживания информации меток по графику с помощью некоторой модели.

How Graph Convolutions work

CNN on image



Graph convolution



Convolution "kernel" depends on Graph structure

Рассмотрим ребро графа, как отношение соседства двух вершин. Основная идея - обобщить работу CNN на граф.

Формально, графовая сверточная сеть (GCN) - это нейронная сеть, которая работает на графах. Учитывая устройство графа $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, GCN принимает в качестве входных данных :

- матрицу признаков вершин X
- матрицу смежности A

Таким образом, скрытый слой в GCN можно записать как :

$$H^i = f(H^{i-1}, A)$$

Отличие моделей GCN в выборе $f(H^i, A)$

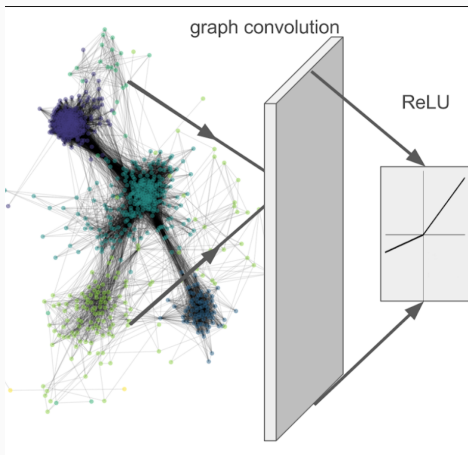
Преследуются две основные задачи :

- Ввести простое и корректное правило $f(H^i, A)$ для модели GCN и показать, как его можно мотивировать из аппроксимации первого порядка свертки спектрального графа
- Показать, как введенное правило может быть использовано для задачи классификации узлов в графе

Рассмотрим простое правило вида :

$$f(H^i, A) = \sigma(AH^i W^i)$$

W^i - матрица весов для i -го слоя, $\sigma(\cdot)$ - функция активации (например, ReLU).



Какие есть явные недостатки правила $f(H^i, A) = \sigma(AH^iW^i)$?

Недостатки

При построении признаков вершин не учитываем признаки самой вершины

Решение

Прибавим к A единичную матрицу $\tilde{A} = A + I_N$

Недостатки

Изменение нормы вектора признаков при применении правила

Решение

Нормализация матрицы A : $\tilde{A}H^i \rightarrow \tilde{D}^{-1}\tilde{A}H^i \rightarrow \tilde{D}^{-0.5}\tilde{A}\tilde{D}^{-0.5}H^i$. Последнее правило не простое усреднение узлов!

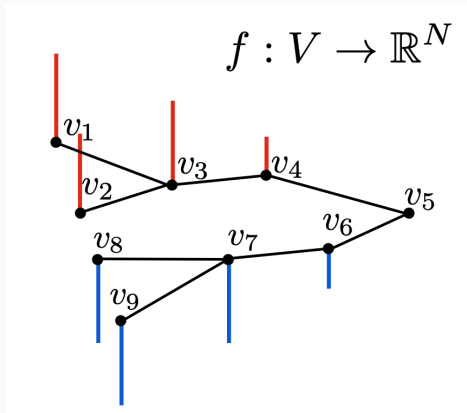
Таким образом, последовательными улучшениями правила, получаем следующий результат

$$f(H^i, A) = \sigma(\tilde{D}^{-0.5} \tilde{A} \tilde{D}^{-0.5} H^i W^i)$$

Какие существуют подходы?

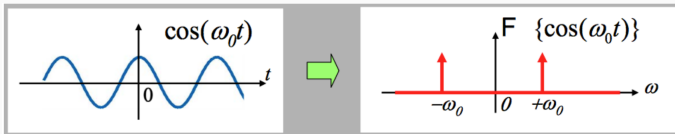
Существует два основных подхода определения графовых фильтров :

- Вершинные (пространственные) конструкции
- Частотные (спектральные) конструкции



Фурье для функций

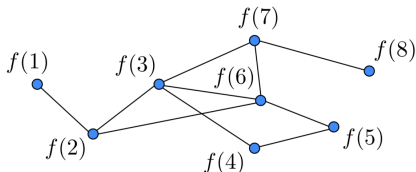
Классическое преобразование Фурье обеспечивает представление частотной области функции



Для создания признакового описания графа нам нужен метод для генерации более «глубокой» версии графа, которая отражает его структуру.

Фурье для графа?

Понятие частоты для сигналов графа... Нам нужна матрица Лапласа!



$$\begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 3 & -1 & 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 4 & -1 & 0 & -1 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & -1 & 0 & -1 & 4 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 & -1 & 3 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} f(1) \\ f(2) \\ f(3) \\ f(4) \\ f(5) \\ f(6) \\ f(7) \\ f(8) \end{pmatrix}$$

L

f

Сигнал $f : V \rightarrow \mathbb{R}^N$

Дифференциальный оператор

$$(Lf)_i = \sum_{j=1}^N A_{ij}(f(i) - f(j))$$

Квадратичная форма Лапласиана

$$f^T Lf = \frac{1}{2} \sum_{i,j=1}^N A_{ij}(f(i) - f(j))^2$$

Нормализованный Лапласиан

$$L_{norm} = I_N - D^{-0.5} A D^{-0.5}$$

- L симметрична
- L положительно полуопределена

Существует ОНБ из собственных векторов $L = \chi^T \Lambda \chi$

$$L = \left(\vec{\chi}_0 \quad \dots \quad \vec{\chi}_{N-1} \right) \begin{pmatrix} \lambda_0 & & 0 \\ & \ddots & \\ 0 & & \lambda_{N-1} \end{pmatrix} \left(\vec{\chi}_0 \quad \dots \quad \vec{\chi}_{N-1} \right)^T$$

$$0 = \lambda_0 < \lambda_1 \leq \dots \leq \lambda_{N-1}$$

one-dimensional Laplace operator: $\frac{d^2}{dx^2}$



eigenfunctions: $e^{j\omega x}$



Classical FT: $\hat{f}(\omega) = \int (e^{j\omega x})^* f(x) dx$

$$f(x) = \frac{1}{2\pi} \int \hat{f}(\omega) e^{j\omega x} d\omega$$

graph Laplacian: L



eigenvectors: χ_ℓ



$f : V \rightarrow \mathbb{R}^N$

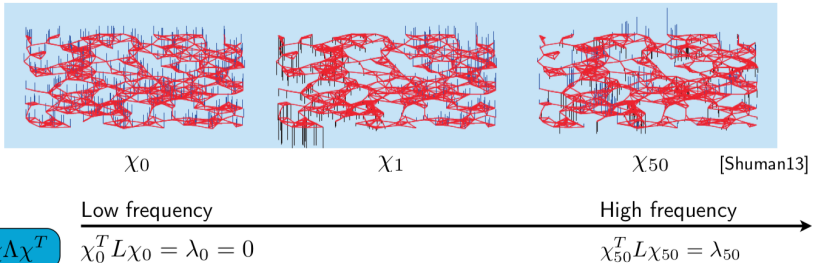
Graph FT: $\hat{f}(\ell) = \langle \chi_\ell, f \rangle = \sum_{i=1}^N \chi_\ell^*(i) f(i)$

$$f(i) = \sum_{\ell=0}^{N-1} \hat{f}(\ell) \chi_\ell(i)$$

Что является «частотой»?

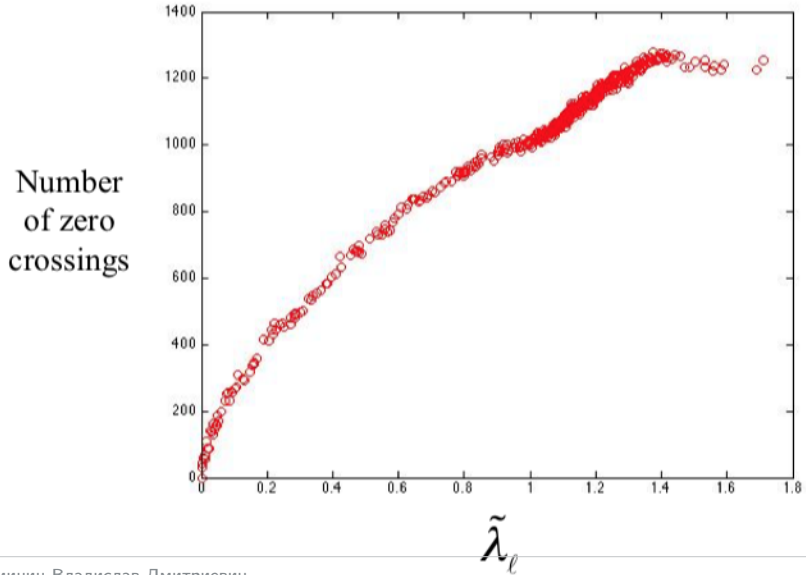
Собственные значения - аналог частоты

Возьмем случайно инициализированный граф, найдем для этого графа собственные вектора u_i и посмотрим на значения $u_i(j)$ для разных собственных значений λ_i .

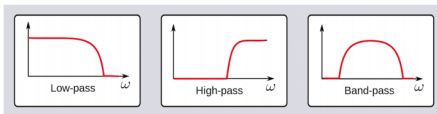
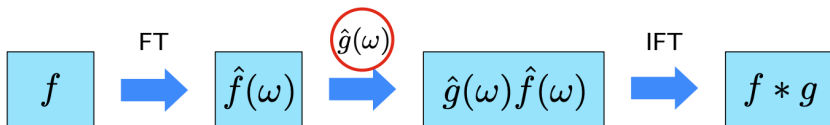


Чем больше «частота» (собственное значение) собственного вектора, тем быстрее меняется соответствующий этому собственному вектору сигнал при переходе от вершины к вершине.

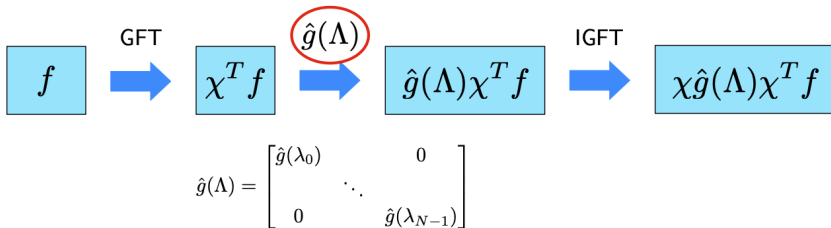
Количество смен знака $f_k(i)$ от величины λ_k



Классическое Фурье : $\tilde{f}(\omega) = \int (e^{j\omega t})^* f(x) dx$ $f(x) = \frac{1}{2\pi} \int (e^{j\omega t}) \tilde{f}(\omega) d\omega$



Преобразование Фурье : $\tilde{f}(l) = \langle \chi_l, f \rangle = \sum_{i=1}^N \chi_l^*(i) f(i)$ $f(i) = \sum_{l=0}^{N-1} \tilde{f}(l) \chi_l(i)$



Если взять $\tilde{g}(\Lambda) = \text{diag}\{\lambda_0, \dots, \lambda_{N-1}\}$, то в результате свертки получим $\chi^T \Lambda \chi f = Lf$

$$\tilde{g}(\theta) = \text{diag}\{\theta\}; \theta \in \mathbb{R}^N$$

Проблема : нам бы хотелось получить фильтр, локализованный в пространстве. Также стоимость обучения непараметризованного фильтра равняется $O(N)$. Хотелось бы сделать это быстрее.

Решение : рассмотрим фильтр $\tilde{g}_{poly}(\theta) = \sum_k \theta_k \Lambda^k$

Утверждение

Для произвольных вершин i, j : если $d(i, j) > k \rightarrow (L^k)_{ij} = 0$

$$(\tilde{g}_{poly}(\theta)\delta_i)_j = \sum_k \theta_k (\Lambda^k)_{ij}$$

Таким образом, переход к фильтру $\tilde{g}_{poly}(\theta)$ помогает не учитывать связь между $\geq k$ -удаленными соседями

Проблема : для применения спектрального подхода нужно найти собственные векторы Лапласиана, а потом произвести перемножение матриц $\chi g_{poly}(\theta) \chi^T$, последнее уже требует $O(N^2)$ операций.

Утверждение

Оказывается, $g_{poly}(\theta) \approx \sum_{k=1}^K \theta'_k T_k(\tilde{\Lambda})$, $T_k(\tilde{\Lambda})$ - полином Чебышева

$$T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x); \quad T_0(x) = 1; \quad T_1(x) = x$$

$$g_{poly}(\theta) * x \approx \sum_{k=1}^K \chi \theta'_k T_k(\tilde{\Lambda}) \chi^T x = \sum_{k=1}^K \theta'_k T_k(\tilde{L}) x$$

$$\tilde{\Lambda} = \frac{2}{\lambda_{max}} \Lambda - I_N \quad \tilde{L} = \frac{2}{\lambda_{max}} L - I_N$$

Итог

Получаем фильтр $g_{poly}(\theta)$, который учитывает соседей порядка K , а также имеет сложность вычисления $O(|\mathcal{E}|K)$

Ограничение

$$K = 1 \rightarrow g_{poly}(\theta) \star x \approx \theta'_0 x + \theta'_1 (L - I_N) x = \boxed{\theta'_0 x - \theta'_1 D^{-0.5} A D^{-0.5} x}$$

Мы все еще можем восстановить богатый класс функций фильтра, сложив несколько таких слоев, но мы не ограничены явной параметризацией, заданной полиномами Чебышева. Кроме того, для фиксированного вычислительного бюджета послойное линейное правило позволяет строить более глубокие модели.

Ограничим количество параметров $\boxed{\theta_0 = -\theta_1}$

$$g_{poly}(\theta) \star x \approx \theta (I_N + D^{-0.5} A D^{-0.5}) x$$

Модель нейронной сети, основанная на свертках графа, может быть построена в виде объединения нескольких сверточных слоев вида $g_{poly}(\theta) \star x = \sum_{k=1}^K \theta'_k T_k(\tilde{L})x$, каждый сопровождается нелинейностью.

Проблема : снова проблемы с нормализацией матрицы полученного правила

Нормализуем полученное правило

$$I_N + D^{-0.5}AD^{-0.5} \rightarrow \tilde{D}^{-0.5}\tilde{A}\tilde{D}^{-0.5}$$

Таким образом, приходим к следующему правилу :

$$g_{poly}(\theta) \star x \approx \theta \tilde{D}^{-0.5} \tilde{A} \tilde{D}^{-0.5} x$$

Пусть в графе N вершин, имеющих по C признаков

В соответствии с вышесказанным, рассмотрим C сигналов длины N . Как задать свертку для данного графа?

К каждому из сигналов применим фильтр с индивидуальным значением θ , а затем просуммируем полученные векторы, тем самым получив новый столбец признаков

Таким образом, предложенный подход позволяет учитывать информацию о признаках из разных позиций, то есть полученный таким правилом вектор содержит в j -той позиции информацию о всех признаках предыдущего уровня.

Применяя такую операцию свертки F раз независимо друг от друга, получаем новую матрицу признаков размерности $N \times F$.

$$Z = \tilde{D}^{-0.5} \tilde{A} \tilde{D}^{-0.5} X \Theta \quad (1)$$

$$X \in \mathbb{R}^{N \times C}; \Theta \in \mathbb{R}^{C \times F}; Z \in \mathbb{R}^{N \times F}$$

где C - количество входных каналов (например, вектор размерности C для каждой вершины), F - количество фильтров.

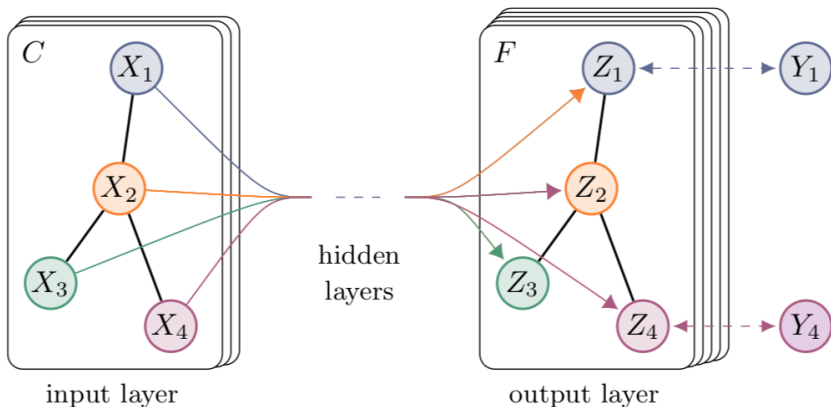
Profit

Такая операция занимает в итоге $O(|\mathcal{E}|FC)$ времени.

Обещанная классификация узлов в графе

Перед обучением вычислим $A' = \tilde{D}^{-0.5} \tilde{A} \tilde{D}^{-0.5}$

$$Z = f(X, A) = \text{softmax}(A' \text{ReLU}(A' X W_0) W_1) \quad (2)$$



$$Z = f(X, A) = \text{softmax}(A' \text{ReLU}(A' X W_0) W_1)$$

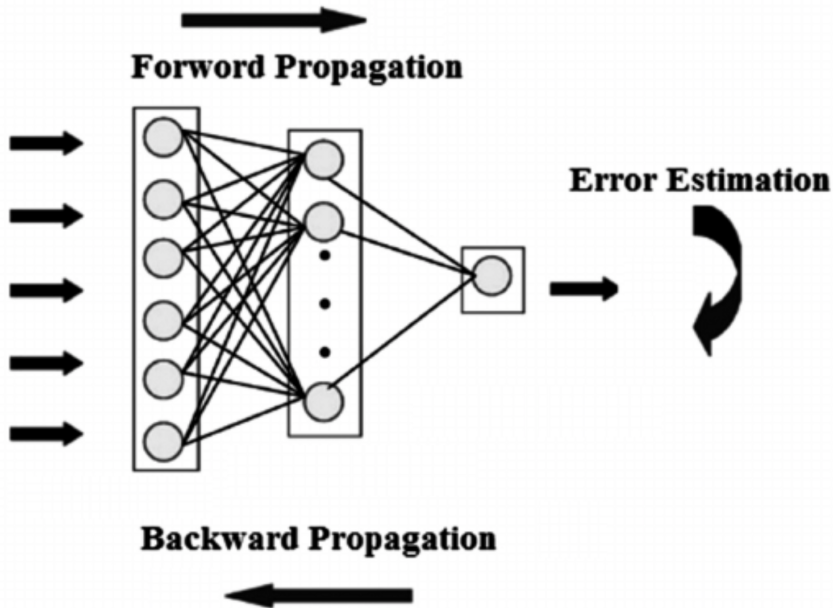
$$W_0 \in \mathbb{R}^{C \times H} \quad W_1 \in \mathbb{R}^{H \times F} \quad \text{softmax}(x_i) = \frac{e^{x_i}}{Z} \quad Z = \sum_i e^{x_i}$$

Функция потерь

В качестве функции потерь будет использоваться кросс-энтропия, определяемая следующим выражением :

$$\mathcal{L} = - \sum_{I \in \mathcal{Y}_I} \sum_{f=1}^F Y_{If} \ln Z_{If}$$

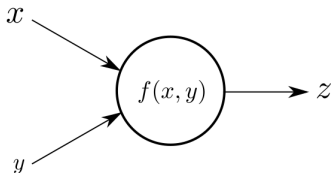
\mathcal{Y}_I - множество узлов, имеющих метки



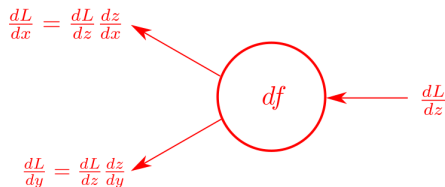
Модель

Модель обучения схожа с моделью собрата GCN - это CNN. На каждом шаге обучения мы высчитываем получившиеся метки в вершинах и высчитываем по ним loss. Далее в игру вступает метод обратного распространения ошибки : наша задача сдвинуть веса матриц W_0 , W_1 в сторону, противоположную направлению градиента, тем самым уменьшить ошибку. Причем градиенты высчитываются рекурсивно, что хорошо показано на картинке.

Forwardpass



Backwardpass



Спасибо за внимание!



Мой GitHub : <https://github.com/VladislavTominin/ML>