Semi-Supervised Classification with Graph Convolutional Neural Networks

Обзор статьи

Томинин Владислав Дмитриевич

МФТИ, Долгопрудный

План рассказа

ИППИ РАН

1. **Введение**
   - Постановка задачи
   - Аналогия с CNN
   - Существующие подходы
   - Устройство GCN
   - Задачи
2. **Выбор правила расспространения**
   - Нестрогие размышления
   - Недостатки введенного правила
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
   - Распространение ошибки

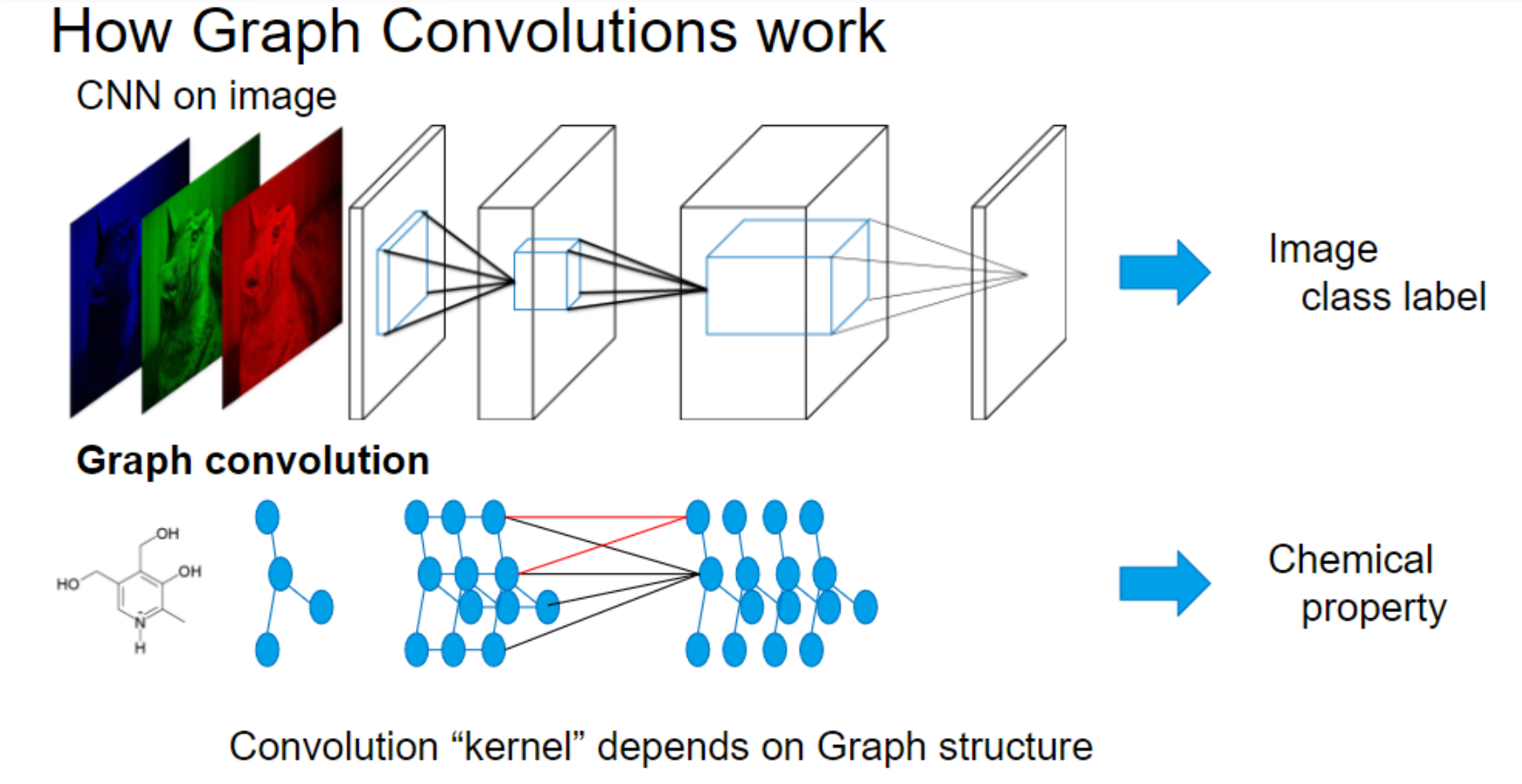
Постановка задачи

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 Лапласса (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 посредством некоторой модели.

Постановка задачи



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

Существующие подходы

Граф Лапласса в функции потерь

$$\mathcal{L} = \underbrace{\mathcal{L}_0}_{loss} + \underbrace{\lambda \left(\sum_{i,j} A_{ij} \| f(X_i) + f(X_j) \| \right)^2}_{\mathcal{L}_{reg}} =$$

$$= \mathcal{L}_0 + \lambda f(X)^T \Delta f(X)$$

Недостатки

Основывается на предположении, что связанные узлы в графе имеют одну и ту же метку, что может ограничить возможности моделирования : ребра графа не обязательно кодируют сходство вершин, а могут нести иную информацию.

Недостатки

Требует **явного** задания $\mathcal{L}_0$

Наш подход

Плюсы

Обучение на заданной функции потерь $\mathcal{L}_0$, избегая $\mathcal{L}_{reg}$

Плюсы

Наличие матрицы смежности, как аргумента в $f(\cdot)$ позволит градиенту расспространяться, затрагивая вершины **без** меток

Устройство GCN

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

Задачи

Преследуются две основные задачи :

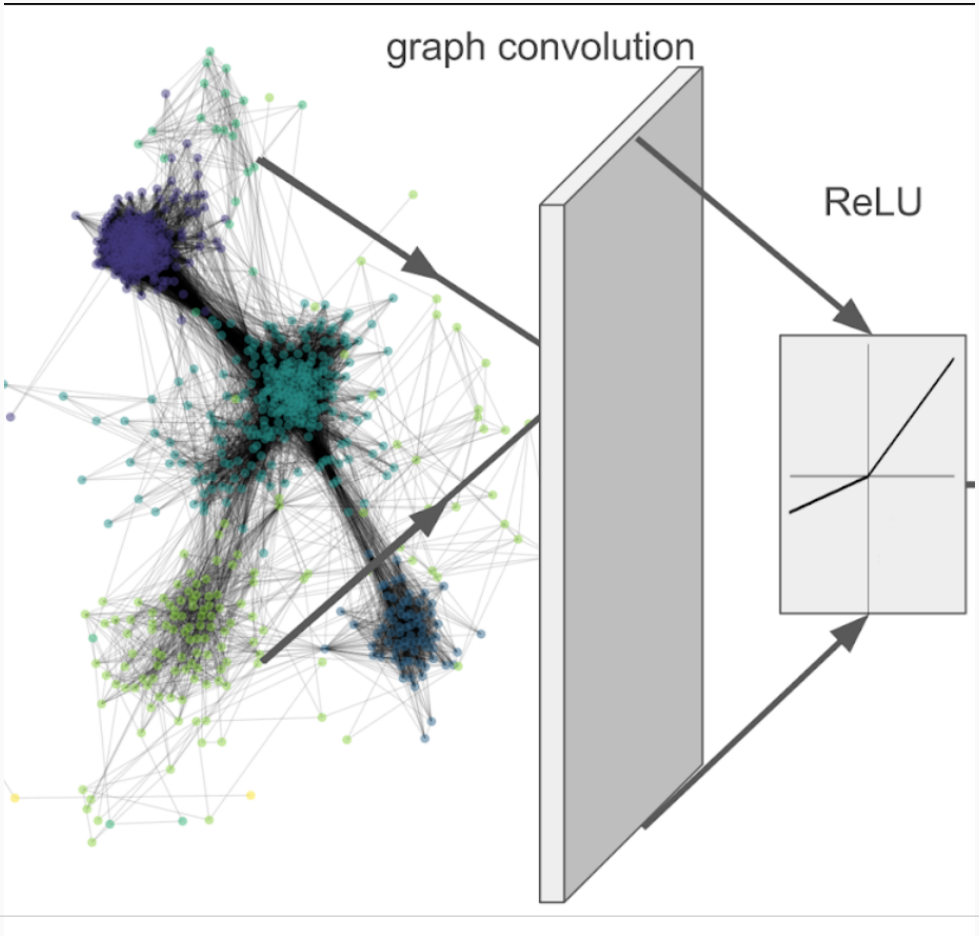
- Ввести простое и корректное правило расспространения для модели GCN и показать, как его можно мотивировать из аппроксимации первого порядка сверток спектрального графа
- Показать, как введенное правило может быть использована для задачи классификациии узлов в графе

Нестрогие размышления

Рассмотрим простое правило распространения вида :

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

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



Недостатки введенного правила

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

Недостатки

При построении признаков вершин не учитывам признаки самой вершины

Решение

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

Недостатки

Изменение нормы вектора признаков при распространении

Решение

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

Итоговое правило расспространения

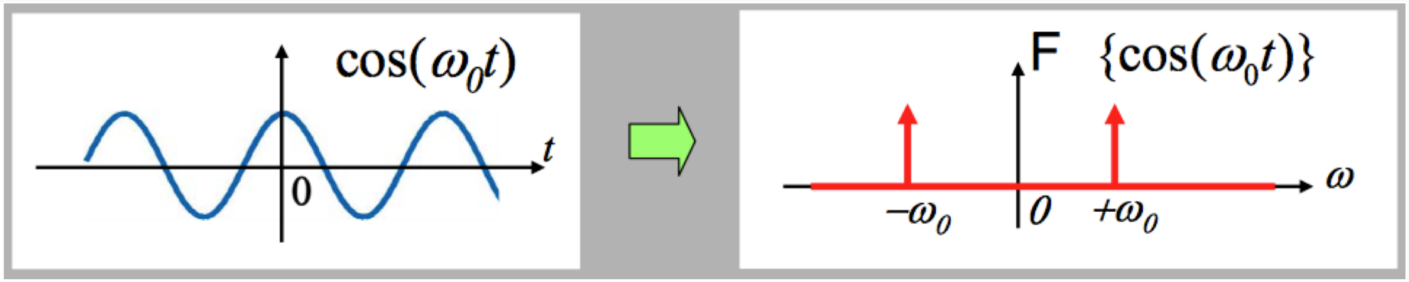
Таким образом, последовательными улучшениями правила, получаем следующий результат

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

Потребность в частоте

Фурье для функций

Классическое преобразование преобразование Фурье обеспечивает представление частотной области функции

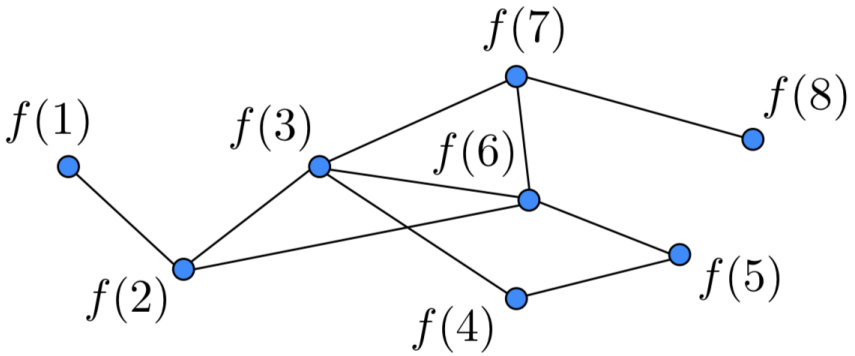


Для создания признакового описания графа нам нужен метод для генерации более «глубокой» версии графа, которая отражает его структуру.

Фурье для графа?

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

Граф Лапласса приводит к преобразованию Фурье!



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

L f

Сигнал $f : V \to \mathbb{R}^N$

Дифференциальный оператор

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

Квадратичная форма Лапласиана

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

Свойства матрицы Лапласса

- L симметрична
- L положительно полуопределена

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

$$L = \begin{pmatrix} \overrightarrow{\chi_0} & \cdots & \overrightarrow{\chi}_{N-1} \end{pmatrix} \begin{pmatrix} \lambda_0 & & 0 \\ & \ddots & \\ 0 & & \lambda_{N-1} \end{pmatrix} \begin{pmatrix} \overrightarrow{\chi_0} & \cdots & \overrightarrow{\chi}_{N-1} \end{pmatrix}^T$$

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

Преобразование Фурье для графа

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

$\Downarrow$

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

$\Downarrow$

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

$\Downarrow$

eigenvectors: χ_ℓ

$\Downarrow$

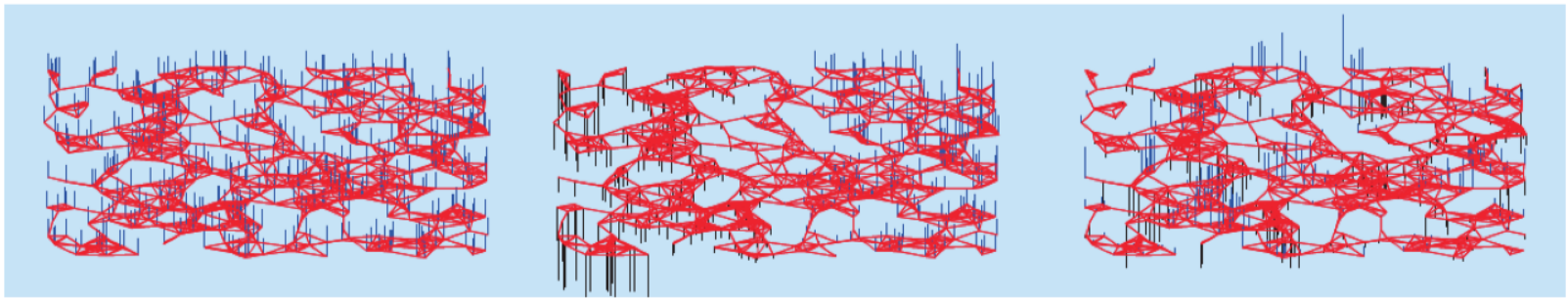
$f : V \to \mathbb{R}^N$

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

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

Собственные значения - аналог частоты

Возьмем рандомно инициализованный граф и представим сигнал для каждой вершины в виде $f(i) = \sum_{k=1}^{N} \chi_k f_k(i)$ и посмотрим на значения $f_k(i)$ для разных собственных значений.



χ_0 χ_1 χ_{50} [Shuman13]

Low frequency → High frequency

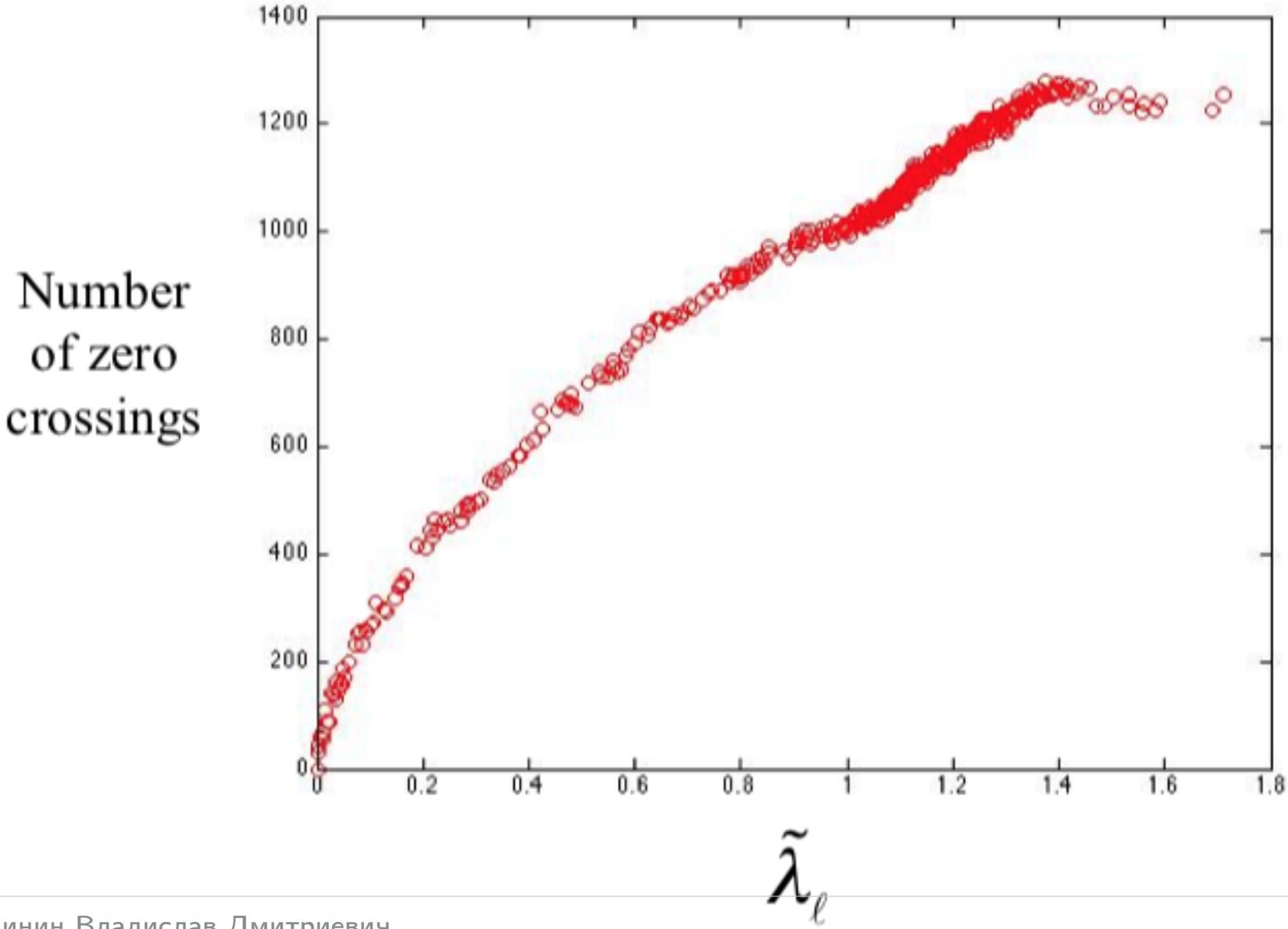
$L = \chi \Lambda \chi^T$

$\chi_0^T L \chi_0 = \lambda_0 = 0$

$\chi_{50}^T L \chi_{50} = \lambda_{50}$

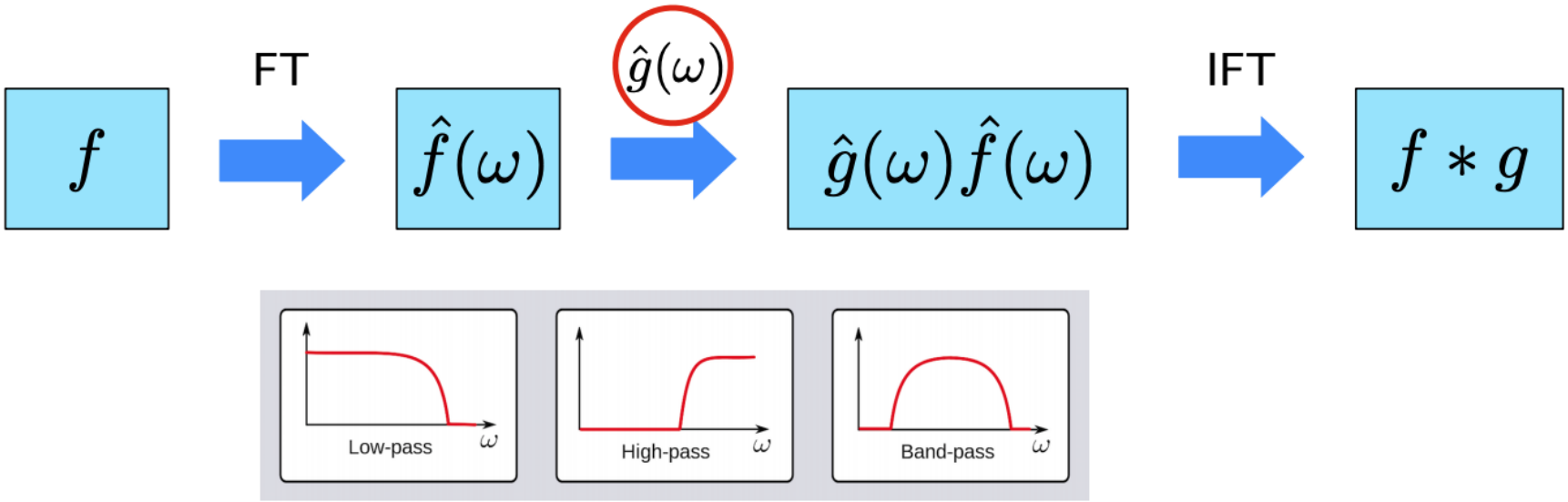
Чем больше «частота» собственного вектора, тем быстрее меняется сигнал, соответствующий этому собственному вектору.

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



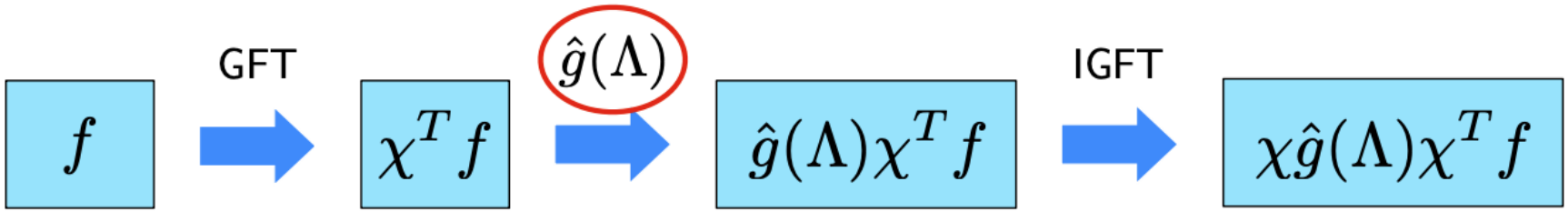
Класссическая фильтрация по частоте

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



Фильтрация на графе

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



$$\hat{g}(\Lambda) = \begin{bmatrix} \hat{g}(\lambda_0) & & 0 \\ & \ddots & \\ 0 & & \hat{g}(\lambda_{N-1}) \end{bmatrix}$$

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

Поиск подходящего фильтра

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

Проблема : непараметризованные фильтры, все параметры которых свободны, не локализованы в пространстве, а стоимость их обучения $O(N)$.
Решение : рассмотрим фильтр $\widetilde{g}_{poly}(\theta) = \sum_k \theta_k \Lambda^k$

Утверждение

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

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

Полиномы Чебышева

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

Утверждение

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

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

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

$$\widetilde{\Lambda} = \frac{2}{\lambda_{max}}\Lambda - I_N \qquad \widetilde{L} = \frac{2}{\lambda_{max}}L - I_N$$

Итог

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

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

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 $\boxed{\theta_0 = -\theta_1}$

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

Проблема : снова проблемы с нормализацией матрицы полученного правила

Нормализуем полученное правило

$I_N + D^{-0.5}AD^{-0.5} \to \widetilde{D}^{-0.5}\widetilde{A}\widetilde{D}^{-0.5}$

Таким образом, приходим к следующему правилу :

$$Z = \widetilde{D}^{-0.5}\widetilde{A}\widetilde{D}^{-0.5}X\theta \tag{1}$$

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

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

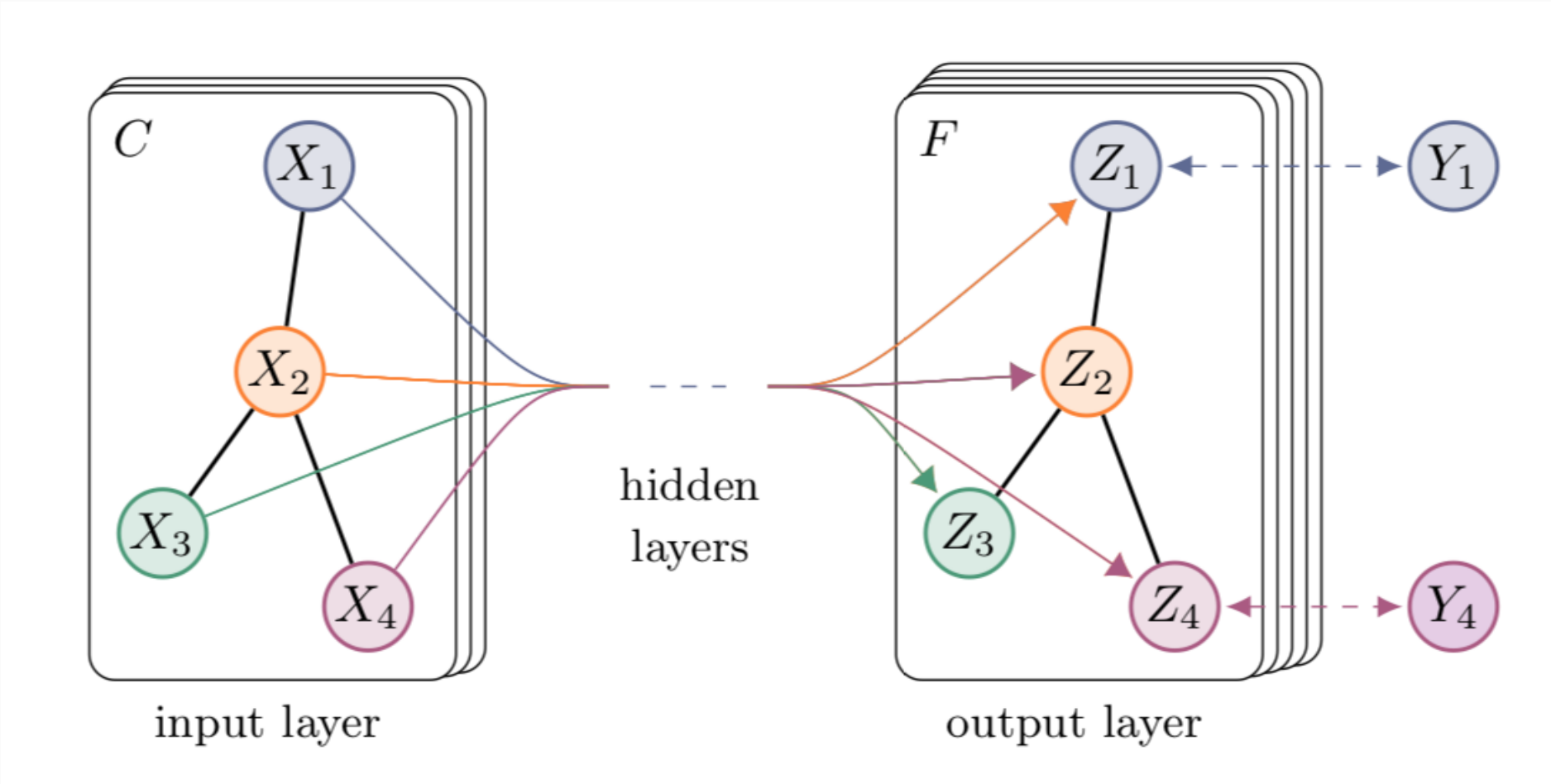
Profit

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

Обещанная классификация узлов в графе

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

$$Z = f(X, A) = softmax(A' ReLu(A' X W_0) W_1) \quad (2)$$



Функция ошибки

Правило

$$Z = f(X, A) = softmax(A' ReLu(A' X W_0) W_1)$$
$$W_0 \in \mathbb{R}^{C \times H}; W_1 \in \mathbb{R}^{H \times F}$$
$$softmax(x_i) = \frac{e^{x_i}}{Z} \qquad Z = \sum_i e^{x_i}$$

Функция потерь

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

$$\mathcal{L} = -\sum_{l \in \mathcal{Y}_l} \sum_{f=1}^{F} Y_{lf} \ln Z_{lf}$$

$\mathcal{Y}_l$ - 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 расспространения 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.