Слайд 3 - введение Мы рассматриваем проблему классификации узлов (например документы) в графе (например сеть цитирования), где размечено только небольшое подмножества узлов. Наша цель - классификация объектов графа с помощью имеющейся информации :

- структуры графа
- знания правильного класса для размеченных вершин

Слайд 4 - 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 $f(\cdot) = f(X, A)$

Наша задача - придумать алгормти, классифицирующий неразмеченные вершины с помощью признаков вершин и структуры графа.

Слайд 5 - аналогия с CNN Все мы знаем о такой архитектуре, как сверхточные нейронные сети и об успехах этого подхода в области задач по распознаванию и классификации изображений. Хотелось бы показать некоторую аналогию CNN и GCNN. Для двух вершин графа, связанных ребром можно считать, что они находятся в отношении соседства, как и соседние пиксели на изображении. Таким образом, обобщив инструмент свертки в CNN хотелось бы получить такой же сильный инструмент для задач классификации на графе.

Слайд 6 - устройство GCN Рассмотрим, что представляет из себя GCN. По сути графовая сверточная сеть 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 $f(H^i, A)$

Слайд 7 - 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 использовано для задачи классификациии узлов в графе

Слайд 8 - простое правило В качестве примера предлагаю рассмотреть простой пример правила

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

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

Слайд 9 - недостатки введенного правила Какие есть явные недостатки правила $f(H^i, A) = \sigma(AH^iW^i)$?

1. При построении признаков вершин не учитывам признаки самой вершины

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

2. Изменение нормы вектора признаков при применении правила

   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!

Слайд 10 - итоговое правило Таким образом, последовательными улучшениями правила, получаем следующий результат

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

Слайд 11 - Существующие подходы Таким образом, в ходе некоторых рассуждений мы пришли к правилу преобразования признаков вершин, но как обосновать этот выбор?

Существует два основных подхода определения графовых фильтров

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

Нас будет интересовать спектральный подход, наша цель построить фильтр так, чтобы он учитывал свойства сигнала.

Слайд 12 - потребность в частоте Классическое преобразование преобразование Фурье обеспечивает представление частотной области функции, из этой информации, как правило, можно узнать много информации.

Для создания признакового описания графа нам нужен схожий метод для генерации более «глубокой» версии графа, которая отражает его структуру, то есть нам нужно ввести преобразование Фурье на графе!

Слайд 13 - граф Лапласса Для того, чтобы ввести преобразование Фурье, рассмотрим матрицу Лапласса.

Сигнал $f : V \to \mathbb{R}^N$

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

Квадратичная форма Лапласса$f^T Lf = \frac{1}{2}\sum_{i,j=1}^{N} A_{ij}(f(i) - f(j))^2$
Хотелось бы подчеркнуть аналогию между матрицей лапласса и дискретным лапласcианом,

Слайд 14 - свойства матрицы Лапласса

- L симметрична
- L положительно полуопределена

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

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

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

Слайд 15 - преобразование Фурье для графа Как мы уже видели, матрица Лапласса имеет некоторые сходства с Лапласcианом. Рассмотрим два оператора и увидим похожесть собственных векторов χ_l на экспоненты, которые являются собственными функциями оператора d^2/dx^2.

Получаем некоторое определение преобразования Фурье для графа!

Слайд 16 - собственные значения - аналог частоты Посмотрим, как наши аналогии работают в реальности.

Возьмем рандомно инициализованный граф, посчитаем для этого графа собственные вектора u_i и посмотрим на значения $u_i(j)$ для разных собственных значений λ_i. Синие штрихи означают, что рассмотренное значение > 0, черные - его отрицательность.

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

Слайд 17 - Количество смен знака $f_k(i)$ от величины λ_k Количество смен знака значения, соответствующего λ_i, для смежных вершин, можно видеть на экране.

Слайд 18 - класссическая фильтрация по частоте пусть на вход поступает какой-то сигнал $f(t)$, а мы хотим выделить его составляющую, которая соответсвует определенной частоте. Порядок действий вы видите на слайде.

Слайд 19 - фильтрация на графе Почему бы не сделать то же самое с нашим графом? Таким образом, вопрос сводится к тому, чтобы подобрать подходящий фильтр.

Слайд 20 - Поиск подходящего фильтра Получаем, что фильтр можно определить произвольной диагональной матрицей. Вспомним о том, что нам хотелось бы получить правило, которое локализовано в пространстве. Также хотелось бы получить правило, которое имеет меньшее количество параметров.

Рассмотрим фильтр $\widetilde{g}_{poly}(\theta) = \sum_k \theta_k \Lambda^k$

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

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

Слайд 21 - Полиномы Чебышева Есть проблема - вычисления требуют $O(N^2)$??? операций.

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

Полиномы Чебышева удовлетворяют следующей рекурсивной формуле :

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

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

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

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

Слайд 22 - Линейное приближение По аналогии с CNN, 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 $K = 1 \rightarrow g_{poly}(\theta) \star x \approx \theta'_0 x + \theta'_1(L - I_N)x = \theta'_0 x - \theta'_1 D^{-0.5}AD^{-0.5}x$

Идея в том, что пореход к линейной модели, на первый взгляд, ограничил сложность итогового правила, однако мы все еще можем восстановить богатый класс функций фильтра, **сложив несколько таких слоев, но мы не ограничены явной параметризацией**, заданной полиномами Чебышева. Также такое правило позволит нам строить более глубоки модели.

Слайд 23 - Линейное приближение **Проблема :** полученное в результате линейного приближения правило изменяет норму векторов, что может привести к так называемому градиентному взрыву. **Нормализуем полученное правило** $I_N + D^{-0.5}AD^{-0.5} \rightarrow \widetilde{D}^{-0.5}\widetilde{A}\widetilde{D}^{-0.5}$

Таким образом, приходим к следующему правилу :

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

Пусть в графе N вершин, имеющих по C признаков В соответствии с вышесказанным, рассмотрим C сигналов длины N. Как задать свертку для данного графа?

Слайд 24 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 $N \times F$. В матричнов виде описанное выглядит так:

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

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

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

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

Линейное время по количеству ребер - хороший результат, учитывая, что зачастую рассматриваемые графы сильно разрежены.

Слайд 25 - Устройство классификатора Перед обучением вычислим $A' = \widetilde{D}^{-0.5}\widetilde{A}\widetilde{D}^{-0.5}$ Таким образом, предлагается взять две свертки, первую из них пропустить через нелинейность в лучших традициях уже существующих архитектур для CNN.

$$Z = f(X, A) = softmax(A'ReLu(A'XW_0)W_1) \tag{2}$$

Слайд 26 - Функция ошибки

$$Z = f(X, A) = softmax(A'ReLu(A'XW_0)W_1)$$

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

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

Слайд 27 - Устройство классификатора Таким образом, когда мы определили функцию активации и loss, можес приступить к самому процессу обучени, который разбивается на несколько этапов :

- Вычисляем метки $z_1, \ldots, z_n$
- Считаем loss для этих меток
- Проводим операцию обратного расспространения ошибки

Эти действия повторяются до сходимости ошибки или до достижения некоторого количества шагов.

Слайд 28 - Расспространение ошибки 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.

Слайд 29 - Спасибо за внимание Сейчас я готов ответить на интересующие вас вопросы.