7. ЛЕКЦИЯ - Теоретические мотивации

Теперь рассмотрим некоторые теоретические основы графовых нейронных сетей. Одним из их наиболее интригующих аспектов является то, что они были независимо разработаны на основе различных теоретических мотиваций. С одной стороны, графовые сети были разработаны на основе теории обработки сигналов графов как обобщение евклидовых сверток на неевклидову область графов. В то же время, подходы к передаче нейронных сообщений, которые составляют основу большинства современных графовых сетей, были предложены по аналогии с алгоритмами передачи сообщений для вероятностного вывода в графических моделях. И, наконец, графовые сети были мотивированы в нескольких работах, основанных на их связи с тестированием Вайсфайлера-Лемана и изоморфизмом графа.

Конвергенция (сближения) трех разных областей в единую структуру алгоритма примечательна. Тем не менее, каждая из теоретических мотиваций имеет свою собственную интуицию и историю, и выбор одной из них существенно влияет на разработку модели. Теперь наша цель – представить ключевые идеи, лежащие в основе различных теоретических мотиваций, чтобы заинтересованный читатель мог свободно исследовать и комбинировать эти интуиции и мотивации по своему усмотрению.

7.1 Графовые нейронные сети и свертки графов

С точки зрения исследовательского интереса и внимания, происхождение графовых нейронных сетей, основанных на связях со свертками графа, является доминирующей теоретической парадигмой. С этой точки зрения, сети возникают из вопроса: как можно обобщить понятие свертки на общие данные, структурированные по графам?

Свертки и преобразование Фурье

Чтобы обобщить понятие свертки на графах, для начала мы должны определить, что мы хотим обобщить, и предоставить краткие справочные сведения. Пусть f и h - две функции. Можно определить общую операцию непрерывной свертки $\star$ как (7.1) :

$$(f \star h)(x) = \int f(y)h(x-y)dy \qquad (7.1)$$

Одним из важнейших аспектов операции свертки является то, что она может быть вычислена путем поэлементного произведения преобразований Фурье двух функций (7.2),

$$(f \star h)(x) = F^{-1}\left(F\big(f(x)\big) \circ F\big(h(x)\big)\right) \tag{7.2}$$

где формула 7.3

$$F\big(f(x)\big) = \hat{f}(s) = \int f(x) e^{-2\pi x^T si} dx \tag{7.3}$$

является преобразованием Фурье $f(x)$, и его обратное преобразование Фурье определяется как (7.4):

$$F^{-1}\left(\hat{f}(s)\right) = \int \hat{f}(x) e^{2\pi x^T si} dx \tag{7.4}$$

В простом случае с одномерными дискретными данными в конечной области $t \in \{0, \dots, N-1\}$ (т. е. ограничиваясь фильтрами с конечной импульсной характеристикой) мы можем упростить эти операции до дискретной круговой свертки (7.5)

$$(f \star_N h)(t) = \sum_{r=0}^{N-1} f(r) h((t-\tau)_{mod} N) \tag{7.5}$$

и дискретного преобразования Фурье (DFT) (7.6 – 7.7) [25],

$$s_k = \frac{1}{\sqrt{N}} \sum_{r=0}^{N-1} f(x_t) e^{-\frac{i2\pi}{N} kt} \tag{7.6}$$

$$= \frac{1}{\sqrt{N}} \sum_{t=0}^{N-1} f(x_t) \left(\cos\left(\frac{2\pi}{N} kt\right) - \mathrm{isin}\left(\frac{2\pi}{N} kt\right)\right) \tag{7.7}$$

где $s_k \in \{s_0 \dots, s_N - 1\}$ – коэффициент Фурье, соответствующий последовательности $(f(x_0),\ f(x_1), \dots, f(x_{N-1}))$. В уравнении (7.5) используем обозначение $\star_N$, чтобы подчеркнуть, что это круговая свертка, определенная в конечной области $\{0, \dots, N-1\}$, но мы часто будем опускать этот индекс для простоты обозначения.

Преобразование Фурье, по сути, говорит нам, как представить входной сигнал в виде взвешенной суммы (комплексных) синусоидальных волн. Если предположим, что входные данные, и их преобразование Фурье являются вещественнозначными, то можно интерпретировать последовательность $[s_0, s_1, \dots, s_{N-1}]$ как коэффициенты ряда Фурье. В этом представлении s_k сообщает нам амплитуду комплексной синусоидальной составляющей $e^{-\frac{i2\pi}{N}k}$, которая имеет частоту $\frac{2\pi k}{N}$ (в радианах). Часто будем обсуждать высокочастотные компоненты, которые имеют

большое k и быстро изменяются, а также низкочастотные компоненты, которые имеют $k << N$ и изменяются медленнее. Это понятие о низкочастотных и высокочастотных составляющих также будет иметь аналог в области графов, где мы будем рассматривать сигналы, распространяющиеся между вершинами.

С точки зрения обработки сигнала, можно рассматривать дискретную свертку $f \star h$, как операцию фильтрации ряда $(f(x_0),\ f(x_1), \ldots, f(x_N))$ с помощью h. Как правило, рассматриваем ряд как соответствующий значениям сигнала во времени, и оператор свертки применяет некоторый фильтр (например, полосовой фильтр) для модуляции этого изменяющегося во времени сигнала.

Одним из важнейших свойств сверток, на которое мы будем опираться ниже, является тот факт, что они являются трансляционными (или сдвиговыми) эквивариантными (7.8):

$$f(t+a) \star g(t) = f(t) \star g(t+a) = (f \star g)(t+a) \tag{7.8}$$

Это свойство означает, что преобразование сигнала и последующая его свертка с помощью фильтра эквивалентно свертке сигнала и последующей трансляции результата. Обратите внимание, что свертки в качестве следствия также эквивариантны разностной операции (7.9):

$$\Delta f(t) \star g(t) = f(t) \star \Delta g(t) = \Delta(f \star g)(t) \tag{7.9}$$

где формула 7.10

$$\Delta f(t) = f(t+1) - f(t) \tag{7.10}$$

является оператором Лапласа (т. е. разностным) для дискретных одномерных сигналов.

Эти понятия фильтрации и эквивалентности трансляции являются центральными для цифровой обработки сигналов (Digital Signal Processin – DSP), а также лежат в основе интуиции сверточных нейронных сетей (Convolutional Neural Network – CNN), которые используют дискретную свертку двумерных данных.

От сигналов времени к сигналам графа

Таким образом, кратко введено понятия фильтрации и свертки применительно к дискретным, изменяющимся во времени, сигналам. Теперь обсудим, как можено связать их с сигналами в графе. Предположим, есть дискретный изменяющийся во времени сигнал $f(t_0),\ f(t_1), \ldots, f(t_{N-1})$. Один из способов представления подобного сигнала – это соответствующий цепочке (или циклу) граф (рис.

7.1), где каждый момент времени t представлен в виде вершины, а каждое значение функции $f(t)$ является значением сигнала в это время. С этой точки зрения удобно представлять сигнал в виде вектора $\boldsymbol{f} \in \mathbb{R}^N$, причем каждое измерение соответствует другой вершине в цепочке графа. Другими словами, имеем, что $f[t] = f(t)$ (как небольшое злоупотребление обозначением). Ребра на графе представляют, как распространяется сигнал, т. е. вперед во времени.

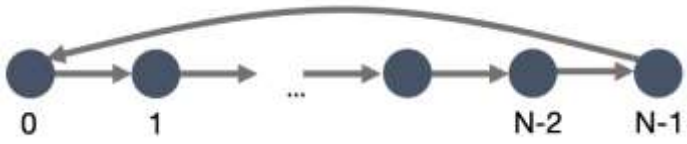

Рис. 7.1: Представление (циклического) временного ряда в виде цепного графа.

Одним из интересных аспектов представления изменяющегося во времени сигнала в виде цепного графа является то, что можно представлять различные операции, такие как временные сдвиги, благодаря матрицам смежности и Лапласа. В частности, матрица смежности для этого графа соответствует циркулянтной матрице A_{c} с (7.11)

$$A_{\mathrm{c}}[i,j] = \begin{cases} 1, \text{если j} = (\mathrm{i}+1)_{\mathrm{mod\ N}} \\ 0, \text{в противном случае} \end{cases} \tag{7.11}$$

и (ненормированный) лапласиан L_{c} для этого графа может быть определен как (7.12)

$$L_{\mathrm{c}} = I - A_{\mathrm{c}} \tag{7.12}$$

Затем можно представить временные сдвиги в виде умножения на матрицу смежности (7.13),

$$(A_{\mathrm{c}} f)[t] = f[(\mathrm{i}+1)_{\mathrm{mod\ N}}] \tag{7.13}$$

и разностная операция путем умножения на лапласиан (7.14),

$$(L_{\mathrm{c}} f)[t] = f[t] - f[(\mathrm{i}+1)_{\mathrm{mod\ N}}] \tag{7.14}$$

Таким образом, можно видеть, что существует тесная связь между матрицами смежности и Лапласа, а также понятиями сдвигов и разностей для сигнала. Умножение сигнала на матрицу смежности приводит к распространению сигналов от вершины к вершине, а умножение на лапласиан вычисляет разницу между сигналом в каждом вершине и его непосредственными соседями.

Учитывая данный вид преобразования сигналов посредством умножения матрицы, мы можем аналогичным образом представить свертку фильтром h как умножение матрицы на вектор f (7.15, 7.16):

$$(f \star h)(t) = \sum_{r=0}^{N-1} f(r)h(t-\tau) \tag{7.15}$$

$$= Q_h f \tag{7.16}$$

где $Q_h \in \mathbb{R}^{N\times N}$ – матричное представление операции свертки с помощью функции фильтра h, а $f = [f(t_0),\ f(t_1), \dots, f(t_{N-1})]^\top$ – векторное представление функции f. Таким образом, в этом представлении рассматриваются свертки, которые могут быть представлены в виде матричного преобразования сигнала в каждой вершине графа. Конечно, чтобы иметь равенство между уравнением (7.15) и уравнением (7.16), матрица Q_h должна обладать некоторыми специфическими свойствами. В частности, чтобы умножение на эту матрицу удовлетворяло эквивариантности трансляции, которая соответствует коммутативности с циркулянтной матрицей смежности A_c, т. е. требуется, чтобы (7.17):

$$A_\mathrm{c} Q_h = Q_h A_\mathrm{c} \tag{7.17}$$

Эквивалентность оператору разности аналогично определяется как (7.18):

$$L_\mathrm{c} Q_h = Q_h L_\mathrm{c} \tag{7.18}$$

Можно показать, что эти требования выполняются для реальной матрицы Q_h, если (7.19)

$$Q_h = p_N(A_\mathrm{c}) = \sum_{r=0}^{N-1} \alpha_i A_c^i \tag{7.19}$$

т. е., Q_h является полиномиальной функцией матрицы смежности A_c. В терминах цифровой обработки сигналов – это эквивалентно идее представления общих фильтров в виде полиномиальных функций оператора сдвига.

Обобщение на общие графы. Таким образом, видно, как сдвиги и свертки изменяющихся во времени дискретных сигналов могут быть представлены на основе матрицы смежности и матрицы Лапласа цепного графа. Учитывая эту точку зрения, можно легко обобщить эти понятия на более общие графы.

В частности, видно, что изменяющийся во времени дискретный сигнал соответствует цепному графу, а также понятие эквивалентности трансляции/разности соответствует свойству коммутативности со смежностью/лапласианом этого цепного графа. Таким образом, можно обобщить эти понятия за пределы цепного графа, рассматривая произвольные матрицы смежности и лапласианы. В то время как в цепном графе сигнал просто распространяется вперед во времени, то в произвольном графе у нас может быть несколько вершин, передающих сигналы друг другу, в зависимости от структуры матрицы смежности. Основываясь на этой идее, мы можем определить сверточные фильтры на общих графах как матрицы Q_h, которые коммутируют с матрицей смежности или лапласианом.

Точнее, для произвольного графа с матрицей смежности A можно представить сверточные фильтры в виде матриц следующего вида (7.20):

$$Q_h = \alpha_0 I + \alpha_1 A + \alpha_2 A^2 + \cdots + \alpha_N A^N \tag{7.20}$$

Интуитивно это дает нам пространственную конструкцию сверточного фильтра на графах. В частности, если умножить вектор свойств вершин $\boldsymbol{x} \in \mathbb{R}^{|\mathcal{V}| \times m}$ на матрицу свертки Q_h, то получим (7.21)

$$Q_h x = \alpha_0 I x + \alpha_1 A x + \alpha_2 A^2 x + \cdots + \alpha_N A^N x \tag{7.21}$$

это означает, что свернутый сигнал $Q_h \boldsymbol{x}[u]$ в каждой вершине $u \in \mathcal{V}$ будет соответствовать некоторой смеси информации в окружении N-го перехода вершины, причем условия $\alpha_0, \ldots, \alpha_N$ контролируют силу информации, поступающей с разных переходов.

Можно легко обобщить данное понятие свертки графа на свойства вершин более высокой размерности. Если есть матрица свойств вершин $\boldsymbol{x} \in \mathbb{R}^{|\mathcal{V}| \times m}$, то можно аналогичным образом применить сверточный фильтр (7.22)

$$Q_h X = \alpha_0 I X + \alpha_1 A X + \alpha_2 A^2 X + \cdots + \alpha_N A^N X \tag{7.22}$$

С точки зрения обработки сигналов можно рассматривать различные аспекты свойств вершины как разные «каналы».

Свертки графов и графовые нейронные сети для передачи сообщений. Уравнение (7.22) также показывает связь между моделью графовой нейронной сети для передачи сообщений и свертками графов. Например, в базовом подходе графовых сетей (см. уравнение 6.5) каждый уровень передачи сообщений, по существу, соответствует применению простого сверточного фильтра (7.23)

$$Q_h = I + A \tag{7.23}$$

в сочетании с некоторыми обучаемыми весовыми матрицами и нелинейностью. В общем, каждый уровень архитектуры графовой сети для передачи сообщений агрегирует информацию из локального окружения вершины и объединяет эту информацию с текущим представлением вершины (см. уравнение 5.4). Мы можем рассматривать эти уровни передачи сообщений как обобщение простого линейного фильтра в уравнении (7.23), где используются более сложные нелинейные функции. Более того, путем укладки нескольких уровней передачи сообщений сети могут неявно оперировать полиномами более высокого порядка матрицы смежности.

Матрица смежности, лапласиан или нормализованный вариант? В уравнении (7.22) определена матрица свертки Q_h для произвольных графов как многочлен матрицы смежности. Определение Q_h таким образом гарантирует, что фильтр коммутирует с матрицей смежности, удовлетворяя обобщенному понятию эквивалентности трансляции. Однако в общем случае коммутативность с матрицей смежности (т. е. эквивалентность трансляции) не обязательно подразумевает коммутативность с лапласианом $L = D - A$ (или любым из его нормализованных вариантов). В этом частном случае цепного графа можно определить матрицы фильтров Q_h, которые одновременно коммутируют как с A, так и с L, но для более общих графов у есть выбор, который нужно сделать в зависимости от того, определяются ли свертки на основе матрицы смежности или какой-либо версии лапласиана. Как правило, в этом случае нет «правильного» решения, и могут быть эмпирические компромиссы в зависимости от сделанного выбора. Понимание теоретических основ этих компромиссов является открытой областью исследований.

На практике для определения сверточных фильтров исследователи часто используют симметричный нормализованный лапласиан $L_{sym} = D^{-\frac{1}{2}}LD^{-\frac{1}{2}}$ или симметричную нормализованную матрицу смежности $A_{sym} = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$. Есть две

причины, по которым эти матрицы желательны. Во-первых, они обе имеют ограниченный спектр, что придает им желаемые свойства численной стабильности. Кроме того эти две матрицы одновременно диагонализуемы, что означает, что они имеют одни и те же собственные векторы. На самом деле, можно легко проверить, что существует простая взаимосвязь между их собственными композициями, поскольку (7.24),

$$L_{sym} = I - A_{sym} \Longrightarrow L_{sym} = U\Lambda U^T \;\; A_{sym} = U(I - \Lambda)U^{\intercal} \tag{7.24}$$

где U – общий набор собственных векторов, а Λ – диагональная матрица, содержащая собственные значения Лапласа. Это означает, что определение фильтров на основе одной из этих матриц подразумевает коммутативность с другой, что является очень удобным и желательным свойством.

Свертки спектрального графа

Теперь понятно, как обобщить понятие сигнала и свертки на область графа. Это можно делать по аналогии с некоторыми важными свойствами дискретных сверток (например, эквивалентностью перевода), и это обсуждение привело к идее представления сверток графа в виде многочленов матрицы смежности (или лапласиана). Однако одним из ключевых свойств сверток, которое было проигнорировано, является взаимосвязь между свертками и преобразованием Фурье. Таким образом, рассмотрим понятие спектральной свертки на графах, где строятся свертки с помощью расширения преобразования Фурье на графы. Спектральная перспектива восстанавливает многие из тех же результатов, которые обсуждалось ранее, а также раскрывает некоторые более общие понятия свертки графа.

Преобразование Фурье и оператор Лапласа. Чтобы мотивировать обобщение преобразования Фурье на графы, полагаемся на связь между преобразованием Фурье и оператором Лапласа (т. е. разностным). Ранее было видено определение оператора Лапласа Δ в случае простого дискретного изменяющегося во времени сигнала (уравнение 7.10), но этот оператор может быть обобщен для применения к произвольным гладким функциям $f: \mathbb{R}^d \rightarrow \mathbb{R}$ как (7.25 – 7.26).

$$\Delta f(x) = \nabla^2 f(x) \tag{7.25}$$

$$= \sum_{i=1}^{n} \frac{\partial^2 f}{\partial x^2} \tag{7.26}$$

Этот оператор вычисляет расходимость ∇ градиента $\nabla f(x)$. Интуитивно оператор Лапласа сообщает нам среднюю разницу между значением функции в точке и значениями функции в соседних областях, окружающих эту точку.

В настройке дискретного времени оператор Лапласа просто соответствует оператору разности (т. е. разнице между последовательными временными точками). В контексте общих дискретных графов это понятие соответствует лапласиану, поскольку по определению (7.27),

$$(Lx)[i] = \sum_{j \in V} A[i,j]\,(x[i] - x[j]) \tag{7.27}$$

который измеряет разницу между значением некоторого сигнала $x[i]$ в вершине i и значениями сигналов всех её соседей. Таким образом, можно рассматривать матрицу Лапласа как дискретный аналог оператора Лапласа, поскольку она позволяет количественно оценить разницу между значением в вершине и значениями у соседей этой вершины.

Теперь, чрезвычайно важным свойством оператора Лапласа является то, что его собственные функции соответствуют комплексным экспонентам. То есть, (7.28)

$$-\Delta\left(e^{2\pi isi}\right) = -\frac{\partial^2\left(e^{2\pi isi}\right)}{\partial t^2} = (2\pi s)^2 e^{2\pi isi} \tag{7.28}$$

таким образом, собственные функции Δ являются теми же комплексными экспонентами, которые составляют моды частотной области в преобразовании Фурье (т. е. синусоидальные плоские волны), с соответствующим собственным значением, указывающим частоту. Фактически, можно даже проверить, что собственные векторы $u_1, \ldots, u_n$ циркулянтного лапласиана $L_c \in \mathbb{R}^{n \times n}$ для цепного графа равны $u_j = \frac{1}{\sqrt{n}}\,[1, \omega_j, \omega_j^2, \ldots, \omega_j^n]$, где $\omega_j = e^{\frac{2\pi j}{n}}$.

Преобразование графа Фурье.

Связь между собственными функциями оператора Лапласа и преобразованием Фурье позволяет обобщить преобразование Фурье на произвольные графы. В частности, можно обобщить понятие преобразования Фурье, рассмотрев собственное разложение общего графа Лапласиана (7.29) [27]:

$$L = U \Lambda U^{\mathsf{T}} \tag{7.29}$$

где определяем собственные векторы U как моды графа Фурье, основываясь понятие мод Фурье в графах. Предполагается, что матрица Λ имеет соответствующие собственные значения вдоль диагонали, и эти собственные значения дают представление о различных значениях частоты на основе графа. Другими словами, поскольку собственные функции общего оператора Лапласа соответствуют модам Фурье, т. е. сложным экспонентам в ряду Фурье, определяем моды Фурье для общего графа на основе собственных векторов лапласиана графа. Другими словами, общий оператор Лапласа соответствует модам Фурье — т.е. комплексным экспонентам в ряду Фурье — мы определяем моды Фурье для общего графа на основе собственных векторов лапласиана графа.

Таким образом, преобразование Фурье сигнала (или функции) $f \in \mathbb{R}^{|V|}$ на графе может быть вычислен как (7.30)

$$s = U^{\intercal} f \tag{7.30}$$

и его обратное преобразование Фурье, вычисленное как (7.31)

$$f = Us \tag{7.31}$$

Свертки графа в спектральной области определяются через точечные произведения в преобразованном пространстве Фурье. Другими словами, учитывая коэффициенты Фурье графа $U^{\intercal} f$ сигнала f, а также коэффициенты Фурье графа $U^{\intercal} h$ некоторого фильтра h, можно вычислить свертку графа с помощью поэлементных произведений как (7.32),

$$f \star_G h = U(U^{\intercal} f \circ U^{\intercal} h) \tag{7.32}$$

где U – матрица собственных векторов лапласиана L и где использовали $\star_G$ для обозначения того, что эта свертка специфична для графа G.

Основываясь на уравнении (7.32), можно представить свертки в спектральной области на основе графических коэффициентов Фурье $\theta_h = \boldsymbol{U}^{\intercal}\boldsymbol{h} \in \mathbb{R}^{|V|}$ функции h. Например, изучая непараметрический фильтр путем прямой оптимизации θ_h и определение свертки (7.33 – 7.34),

$$f \star_G h = U(U^{\intercal} f \circ \theta_h) \tag{7.33}$$

$$= \left(U_{diag}(\theta_h)U^{\intercal}\right) \tag{7.34}$$

где $diag(\theta_h)$ – матрица со значениями θ_h по диагонали. Однако фильтр, определенный таким непараметрическим способом, не имеет реальной зависимости от структуры графа и может не удовлетворять многим свойствам, которые хотим получить от свертки. Например, такие фильтры могут быть произвольно нелокальными.

Чтобы гарантировать, что спектральный фильтр θ_h соответствует значимой свертке на графе, естественным решением является параметризация θ_h на основе собственных значений лапласиана. В частности, можно определить спектральный фильтр как $p_N(\Lambda)$, это многочлен степени N от собственных значений лапласиана. Определение спектральной свертки таким образом гарантирует, что свертка коммутирует с Лапласиан (7.35 – 7.36).

$$f \star_G h = (Up_N(\Lambda)U^{\intercal})f \tag{7.35}$$

$$= p_N(L)f \tag{7.36}$$

Более того, это определение обеспечивает понятие локальности. Если используется полином степени k, то гарантируется, что отфильтрованный сигнал в каждом вершине зависит от информации в его окрестности. Таким образом, в конце концов, выводя свертки графа со спектральной точки зрения, можно восстановить ключевую идею о том, что свертки графа могут быть представлены полиномами Лапласиана (или одним из его нормализованных вариантов). Однако спектральная перспектива также раскрывает более общие стратегии определения сверток на графах.

Интерпретируя собственные векторы Лапласа как частоты в стандартном преобразовании Фурье, можно интерпретировать коэффициенты Фурье как соответствующие различным частотам. В общем случае графа больше не может интерпретировать преобразование Фурье графа таким образом. Однако можно провести аналогии с высокочастотными и низкочастотными компонентами. В частности, можно вспомнить, что собственные векторы $u_i, i = 1, \ldots, |V|$ лапласиана решают задачу минимизации (7.37):

$$min_{u_i \in \mathbb{R}^{|V|}} : u_i \perp u_j \forall j < i \, \frac{u_i^{\intercal} L u_i}{u_i^{\intercal} u_i} \tag{7.37}$$

по теореме Рэлея-Ритца (7.38). Это есть

$$u_i^T L u_i = \frac{1}{2}\sum_{u,v\in V} A[u,v]\,(u_i[u] - u_i[v])^2 \qquad (7.38)$$

по свойствам лапласиана. Вместе эти факты подразумевают, что наименьший собственный вектор лапласиана соответствует сигналу, который изменяется от вершины к вершине на наименьшую величину на графе, второй наименьший собственный вектор соответствует сигналу, который изменяется на вторую наименьшую величину, и так далее. Действительно, использовались эти свойства собственных векторов Лапласа, когда выполнялась спектральная кластеризация. В этом случае показано, что собственные векторы Лапласа можно использовать для назначения вершин сообществам, чтобы минимизировать количество ребер, которые переходите между сообществами. Теперь можно интерпретировать этот результат с точки зрения обработки сигналов: собственные векторы Лапласа определяют сигналы, которые плавно изменяются по всему графу, причем самые плавные сигналы указывают на крупнозернистую структуру сообщества графа.

Графовые нейронные сети, вдохновлённые свёрткой

Предыдущие рассмотрения обобщили понятие свертки на графы. Видно, что основные сверточные фильтры на графах могут быть представлены в виде многочленов (нормализованной) матрицы смежности или лапласиана. Было рассмотрено, как пространственную, так и спектральную мотивацию этого факта, и, как спектральная перспектива может быть использована для определения более общих форм свертки графа на основе преобразования Фурье графа. Теперь кратко рассмотрим, как различные модели GNN были разработаны и вдохновлены на основе этих связей.

Чисто сверточные подходы. Некоторые из самых ранних работ по GNN могут быть непосредственно сопоставлены с определениями свертки графов из предыдущих рассмотрений. Ключевая идея этих подходов заключается в том, что они используют либо уравнение (7.34), либо уравнение (7.35) для определения сверточного слоя, а полная модель определяется путем укладки и объединения нескольких сверточных слоев с нелинейностями. Например, в ранних работах экспериментировали с непараметрическим спектральным фильтром (уравнение 7.34), а также параметрический спектральный фильтр (уравнение 7.35), где они определяли многочлен $p_n(\Lambda)$ с помощью кубического сплайнового подхода. Следуя этим работам, были определины свертки на основе уравнения 7.35 и определили $p_n(L)$ с использованием полиномов Чебышева. Этот подход выигрывает от того факта, что Полиномы Чебышева имеют эффективную рекурсивную форму-

лировку и обладают различными свойствами, которые делают их пригодными для полиномиальной аппроксимации. В рамках аналогичных подходов Ляо и др. изучают многочлены лапласиана на основе алгоритма Ланцоша.

Существуют также подходы, которые выходят за рамки вещественных многочленов Лапласиан (или матрица смежности). Например, рассматривают Полиномы Кэли Лапласиана и Бьянки и др.

Построение графов сверточных сетей и подключений к передаче сообщений. Использование понятие свертки графов для определения одной из самых популярных архитектур GNN, широко известной как сверточная сеть графов (graph convolutional network – GCN). Ключевая идея подхода GCN заключается в том, что можно создавать мощные модели, укладывая очень простые сверточные слои графа. Базовый уровень GCN определен в как (7.39),

$$H^{(k)} = \sigma\left(\tilde{A}H^{(k-1)}W^{(k)}\right) \tag{7.39}$$

где $\tilde{A} = (D+I)^{-\frac{1}{2}}(I+A)(D+I)^{-\frac{1}{2}}$ – нормализованный вариант матрицы смежности (с самоциклами), а $W^{(k)}$ – обучаемая матрица параметров. Эта модель изначально была мотивирована как комбинация простой свертки графа (основанной на многочлене $I + A$), с обучаемой весовой матрицей и нелинейностью.

Как обсуждалось, также можно интерпретировать модель GCN как вариацию базового подхода к передаче сообщений GNN. В общем случае, если рассмотреть объединение простой свертки графа, определенной через многочлен $I + A$, с нелинейности и обучаемые весовые матрицы восстанавливаемые базовую графовую нейронную сеть (7.40):

$$H^{(k)} = \sigma\left(AH^{(k-1)}W_{neigh}^{(k)} + H^{(k-1)}W_{self}^{(k)}\right) \tag{7.40}$$

Другими словами, простая свертка графа, основанная на $I + A$, эквивалентна агрегированию информации от соседей и объединению ее с информацией из самого вершины. Таким образом, мы можем рассматривать понятие передачи сообщений как соответствующее простой форме свертки графа в сочетании с дополнительными обучаемыми весами и нелинейностями.

Чрезмерное сглаживание как сверточный фильтр нижних частот. Интуитивная идея чрезмерного сглаживания заключается в том, что после слишком большого количества циклов передачи сообщений вложения для всех вершин начинают выглядеть одинаково и являются относительно неинформативными.

Основываясь на связи между GNN для передачи сообщений и свертками графа, теперь можно понять чрезмерное сглаживание с точки зрения обработки сигналов графа.

Ключевая интуиция заключается в том, что укладка нескольких раундов передачи сообщений в базовый GNN аналогичен применению фильтра свертки нижних частот, который создает сглаженную версию входного сигнала на графе. В частности, предположим, что упрощена базовое GNN (уравнение 7.40) до следующего уравнения обновления (7.41):

$$H^{(k)} = A_{sym}H^{(k-1)}W^{(k)} \tag{7.41}$$

По сравнению с базовым GNN в уравнении (7.40) упрощена модель, убрана нелинейность и отсутствует добавление «собственных» вложений на каждом этапе передачи сообщений. Для математической простоты и численной стабильности предположено, что используется симметричная нормализованная матрица смежности $A_{sym} = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$, а не нормализованная матрица смежности. Эта модель аналогична простому подходу GCN и по существу сводится к получению среднего значения по соседним вложениям при каждом раунде передачи сообщений.

Теперь легко видеть, что после K раундов передачи сообщений на основе уравнение (7.41), получим представление, которое зависит от K-й степени матрицы смежности (7.42):

$$H^{(k)} = A_{sym}^{K}XW \tag{7.42}$$

где W – некоторый линейный оператор, а X – матрица функций входного вершины. Чтобы понять связь между чрезмерным сглаживанием и сверточными фильтрами, нам просто нужно признать, что умножение $A_{sym}^{K}X$ функций входного вершины на высокую мощность матрицы смежности может быть интерпретировано как сверточный фильтр, основанный на низкочастотных сигналах графа Лапласиана.

Например, предположим, что используется достаточно большое значение K, такое, что достигнута фиксированная точка следующего повторения (7.43):

$$A_{sym}H^{(k)} = H^{(k)} \tag{7.43}$$

Можно проверить, что эта фиксированная точка достижима при использовании нормализованной матрицы смежности, поскольку доминирующее собственное значение A_{sym} равно единице. Можно видеть, что в этой фиксированной точке все функции вершины сойдутся, чтобы быть полностью определенными доминирующим собственным вектором A_{sym}, и, в более общем плане, более высокие степени A_{sym} будут подчеркивать наибольшие собственные значения этой матрицы. Более того, мы знаем, что наибольшие собственные значения A_{sym} соответствуют наименьшим собственным значениям его аналога, симметричного нормированного лапласиана L_{sym} (например, см. уравнение 7.24). Вместе эти факты подразумевают, что умножение сигнала на большие мощности A_{sym} соответствует сверточному фильтру, основанному на самых низких собственных значениях (или частотах) L_{sym}, т. е. он создает фильтр нижних частот.

Таким образом, из этой упрощенной модели можно видеть, что укладка многих раундов передачи сообщений приводит к сверточным фильтрам, которые являются низкочастотными, и – в худшем случае — эти фильтры просто сводят все представления вершин к постоянным значениям внутри связанных компонентов на графе (т. е. «нулевая частота» лапласиан).

Конечно, на практике используются более сложные формы передачи сообщений, и эта проблема частично решается путем включения предыдущего сообщения каждого вершины встраивание на этапе обновления при передаче сообщений. Тем не менее, поучительно понять, как наивное наложение «более глубоких» сверток на графы на самом деле может привести к более простым, а не более сложным сверточным фильтрам.

Графовые нейронные сети без передачи сообщений. Вдохновленные подключениями к сверткам графов, в нескольких недавних работах также было предложено упростить GNN, устранив итеративный процесс передачи сообщений. В этих подходах модели обычно определяются как (7.44),

$$Z = MLP_\theta(f(A)MLP_\phi(X)) \tag{7.44}$$

где $f: \mathbb{R}^{|V|\times|V|} \to \mathbb{R}^{|V|\times|V|}$ – некоторая детерминированная функция матрицы смежности A, MLP обозначает плотную нейронную сеть, $X \in \mathbb{R}^{|V|\times m}$ – матрица характеристик входного вершины, а $Z \in \mathbb{R}^{|V|\times d}$ – матрица представлений изученных вершин. Например, определяют (7.45)

$$f(A) = \tilde{A}^k \tag{7.45}$$

где $\tilde{A} = (D+I)^{-\frac{1}{2}}(I+A)(D+I)^{-\frac{1}{2}}$ – симметричная нормализованная матрица смежности (с добавлением самоциклов). Теперь определяется f по аналогии с алгоритмом персонализированного PageRank как (7.46 – 7.47)

$$f(A) = \alpha\left(I - (1-\alpha)\tilde{A}\right)^{-1} \tag{7.46}$$

$$= \alpha \sum_{k=0}^{\infty}\left(I - \alpha\tilde{A}\right)^{k} \tag{7.47}$$

Интуиция, лежащая в основе этих подходов, заключается в том, что нам часто не нужно чередовать обучаемые нейронные сети со слоями свертки графа. Вместо этого можно просто использовать нейронные сети для изучения преобразований объектов в начале и в конце модели и применить детерминированный уровень свертки для использования структуры графа. Эти простые модели способны превзойти более сильно параметризованные модели передачи сообщений по многим критериям классификации.

Также появляется все больше свидетельств того, что использование симметричной нормализованной матрицы смежности с самоциклами приводит к эффективной свертке графа, особенно в этой упрощенной настройке без передачи сообщений. Обнаружено, что свертки, основанные на A, достигли наилучших эмпирических показателей. Также предоставляется теоретическая поддержка этим результатам. Доказывается, что добавление автоциклов сокращает спектр соответствующего лапласиана графа за счет уменьшения величины доминирующего собственного значения. Интуитивно понятно, что добавление петель уменьшает влияние удаленных узлов и делает отфильтрованный сигнал более зависимым от локальных окрестностей на графе.

7.2 Вероятностные графические модели

Графические нейронные сети хорошо понятны и хорошо мотивированы как расширения сверток к данным, структурированным в виде графов. Однако существуют альтернативные теоретические обоснования для структуры GNN, которые могут обеспечить интересные и новые перспективы.

Одним из ярких примеров является мотивация GNNs, основанная на связях с вариационным выводом в вероятностных графических моделях (probabilistic graphical models – PGM). В этой вероятностной перспективе рассматриваются вложения $z_u, \forall u \in V$ для каждой вершины как скрытые переменные, которые

пытаются вывести. Предполагается, что наблюдаемая структура графа (т. е. матрица смежности, A) и функции входного вершины, X, и цель – вывести лежащие в основе скрытые переменные (т. е. вложения z_u), которые могут объяснить эти наблюдаемые данные. Операция передачи сообщений, лежащая в основе GNNs, затем может рассматриваться как аналог нейронной сети определенных алгоритмов передачи сообщений, которые обычно используются для вариационного вывода распределений по скрытым переменным.

Вложения распределений в гильбертово пространство

Чтобы понять связь между GNN и вероятностным выводом, необходимо сначала (кратко) ввести понятие вложенных распределений в гильбертовы пространства. Пусть $p(x)$ обозначает функцию плотности вероятности, определенную над случайной величиной $x \in \mathbb{R}^m$. Учитывая произвольную (и, возможно, бесконечномерную) карту признаков $\varphi : \mathbb{R}^m \to \mathcal{R}$, можно представить плотность $p(x)$ на основе ее ожидаемого значения под этой картой признаков (7.48) [28]:

$$\mu_x = \int \phi(x)p(x)dx \tag{7.48}$$

Ключевая идея вложений распределений в гильбертово пространство заключается в том, что уравнение (7.48) будет инъективным до тех пор, пока используется подходящая карта объектов φ. Это означает, что μ_x может служить достаточной статистикой для $p(x)$, и любые вычисления, которые необходимо выполнить для $p(x)$, могут быть эквивалентно представлены как функции вложения μ_x. Хорошо известным примером отображения объектов, которое гарантировало бы это инъективное свойство, является отображение объектов, индуцированное ядром радиальной базисной функции Гаусса (RBF).

Изучение вложений распределений в гильбертово пространство является богатой областью статистики. Однако в контексте связи с GNN ключевой вывод заключается просто в том, что можно представлять распределения $p(x)$ как вложения μ_x в некотором пространстве объектов. Будем использовать это понятие для мотивации алгоритма передачи сообщений GNN как способа изучения вложений, которые представляют распределение по скрытым элементам вершины $p(z_v)$.

Графы как графические модели

Рассматривая данные графа с вероятностной точки зрения, можно предположить, что данная структура графа определяет зависимости между различными вершинами. Конечно, обычно интерпретируются графические данные таким образом: вершины, которые соединены в графе, обычно считаются каким-то образом

связанными. Однако в вероятностной среде рассматривается это понятие зависимости между вершинами формальным, вероятностным образом.

Чтобы быть точным, мы говорим, что граф $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ определяет марковское случайное поле (7.49):

$$p(\{x_v\}, \{z_v\}) \propto \prod_{v \in V} \Phi(x_v, z_v) \prod_{(u,v) \in \varepsilon} \Psi(z_u, z_v) \tag{7.49}$$

где Φ и Ψ – неотрицательные потенциальные функции, и где используется $\{x_v\}$ в качестве сокращения для множества $\{xv, \forall v \in V\}$. Уравнение (7.49) говорит о том, что распределение $p(\{x_v\}, \{z_v\})$ над узловыми объектами и вложениями вершин разлагается на множители в соответствии со структурой графа. Интуитивно понятно, что $\Phi(x_v, z_v)$ указывает вероятность вектора признаков вершины x_v с учетом его скрытого вложения в вершина z_v, в то время как Ψ управляет зависимостью между подключенными вершинами. Таким образом, предполагается, что функции вершины являются их скрытыми вложениями, и, что скрытые вложения для подключенных вершин зависят друг от друга (например, подключенные вершины могут иметь аналогичные вложения).

В стандартной настройке вероятностного моделирования Φ и Ψ обычно определяются как параметрические функции, основанные на знаниях предметной области, и, чаще всего, предполагается, что эти функции происходят из семейства экспоненциальных для обеспечения управляемости.

Встраивание вывода среднего поля

Учитывая марковское случайное поле, определяемое уравнением (7.49), где цель состоит в том, чтобы вывести распределение скрытых вложений $p(z_v)$ для всех вершин $v \in V$, а также неявно изучить потенциальные функции Φ и Ψ. В более интуитивных терминах цель состоит в том, чтобы вывести скрытые представления для всех вершин графа, которые могут объяснить зависимости между наблюдаемыми функциями вершины.

Чтобы сделать это, ключевым шагом является вычисление апостериорного $p(\{z_v\}|\{x_v\})$, т.е. вычисление вероятности определенного набора скрытых вложений, заданных наблюдаемые особенности. В общем, вычисление этого апостериора является вычислительно трудноразрешимым — даже если Φ и Ψ известны и четко определены — поэтому необходимо прибегнуть к приближенным методам.

Один популярный подход, который можно использовать, заключается в использовании вариационного вывода среднего поля, где аппроксимируется апостериорное значение с помощью некоторых функций q_v, основанный на предположении (7.50):

$$p(\{x_v\},\{z_v\}) \approx g(\{z_v\}) = \prod_{v\in V} g_v\,(z_v) \qquad (7.50)$$

где каждый q_v является допустимой плотностью. Ключевая интуиция в выводе среднего поля заключается в том, что апостериорное распределение по скрытым переменным разлагается на V независимые распределения, по одному на вершинt.

Чтобы получить аппроксимирующие функции q_v, которые являются оптимальными в приближении среднего поля, стандартный подход заключается в минимизации расхождения Кулбека–Лейблера (KL) между приблизительным задним и истинным задним (7.51):

$$KL(g(\{z_v\})|\{p(\{z_v\}|\{x_v\})\}) = \int \prod_{v\in V} g(\{z_v\}) \log\left(\frac{\prod_{v\in V} g(\{z_v\})}{p(\{z_v\}|\{x_v\})}\right) \prod_{v\in V} dz_v \quad (7.51)$$

Расхождение KL является одним из канонических способов измерения расстояния между распределениями вероятностей, поэтому нахождение функций q_v, которые минимизируют уравнение (7.51), дает приблизительное заднее значение, максимально близкое к истинному заднему значению в предположении о среднем поле. Конечно, непосредственно минимизирующее уравнение (7.51) невозможно, поскольку оценка расхождения KL требует знания истинного заднего.

Методы вариационного вывода могут быть использованы, чтобы показать, что $q_v(z_v)$, которые минимизируют KL, должны удовлетворять следующим уравнениям с фиксированной точкой (7.52):

$$\log(q_v(z_v)) = c_v + \log\big(\Phi(x_v, z_v)\big) + \sum_{v\in N(v)} \int g(z_v) \log\big(\Psi(z_u, z_v)\big) dz_u \quad (7.52)$$

где c_v – константа, которая не зависит от $q_v(z_v)$ или z_v. На практике можно аппроксимировать это решение с фиксированной точкой, инициализировав некоторые начальные предположения $q_v^{(t)}$ к допустимым распределениям вероятностей и итеративному вычислению (7.53).

$$\log\left(q_v^{(t)}(z_v)\right) = c_v + \log\big(\Phi(x_v, z_v)\big) + \sum_{v\in N(v)} \int g_u^{(t-1)}(z_v) \log\big(\Psi(z_u, z_v)\big) dz_u \quad (7.53)$$

Обоснование уравнения (7.52) выходит за рамки данного изложения. Однако для целей основными идеями являются следующие:

1. Можно аппроксимировать истинное апостериорное $p(\{z_v\}|\{x_v\})$ по скрытым вложениям, используя предположение о среднем поле, где предполагается, что Q апостериорное разложение на $|V|$ независимые распределения $p(\{z_v\}\,|\{x_v\}) \approx \Pi_{v\in V}\, q_v(z_v)$.

2. Оптимальное приближение в предположении о среднем поле задается фиксированной точкой в уравнении (7.52), где приблизительное заднее $q_v(z_v)$ для каждого скрытого вершины вложение является функцией признака вершины z_x и предельных распределений $q_u(\boldsymbol{z})_u, \forall u \in \mathcal{N}(v)$ вложений соседей вершины.

На этом этапе начинает возникать соединение с GNNs. В частности, если рассмотривать итерацию с фиксированной точкой в уравнении (7.53), то увидим, что обновленное предельное распределение $q_v^{(t)}(\boldsymbol{z}_v)$ является функцией характеристик вершины x_v (через потенциальную функцию Φ), а также функцией множества соседних маргиналов $\{q_v^{(t-1)}(\boldsymbol{z}_v), \forall u \in \mathcal{N}(v)$ из предыдущей итерации (через потенциальную функцию Ψ). Эта форма передачи сообщений в значительной степени аналогична передаче сообщений в GNNs. На каждом шаге обновляется значения в каждом вершине на основе набора значений в окрестности вершины. Ключевое различие заключается в том, что уравнения передачи сообщений в среднем поле работают с распределениями, а не с вложениями, которые используются при стандартной передаче сообщений GNN.

Можно установить связь между GNNs и выводом среднего поля даже более плотным за счет использования вложений в гильбертово пространство. Предположим, что есть некоторая инъективная карта объектов φ и можно представить все маргиналы $q_v(\boldsymbol{z}_v)$ в виде вложений (7.54).

$$\mu_v = \int q_v(\boldsymbol{z}_v)\boldsymbol{\phi}\,(\boldsymbol{z}_v)d\boldsymbol{z}_v \in \mathbb{R}^d \tag{7.54}$$

С помощью этих представлений можуj переписать итерацию с фиксированной точкой в уравнении (7.52) как (7.55),

$$\mu_v^{(t)} = c + f\left(\mu_v^{(t-1)}, x_v\right), \{\mu_u, \forall u \in \mathcal{N}(v)\} \tag{7.55}$$

где f – векторнозначная функция. Обратите внимание, что f агрегирует информацию из набора соседних вложений (т. е. $\{\mu_u, \forall u \in \mathcal{N}(v)\}$ и обновляет текущее представление вершины (т.е. $\mu_v^{(t-1)}$) использование этих агрегированных данных. Таким образом, можуj видеть, что встроенный вывод среднего поля в точности соответствует форме нейронного сообщения, проходящего по графу!

Теперь, в обычном сценарии вероятностного моделирования, нужно определить потенциальные функции Φ и Ψ, а также карту объектов φ, используя некоторые знания предметной области. И учитывая некоторые Φ, Ψ и ψ. Теперь можно попытаться аналитически вывести функцию f в уравнении (7.55), что позволило бы работать со встроенной версией вывода среднего поля. Однако, в качестве альтернативы, можно просто попытаться изучить вложения μ_v сквозным способом, используя некоторые контролируемые сигналы, и можно определить f как произвольную нейронную сеть. В других другими словами, вместо того, чтобы указывать конкретную вероятностную модель, можно просто изучить вложения μ_v, которые могли бы соответствовать некоторой вероятностной модели. Основываясь на этой идее, определяют f аналогично базовому GNN как (7.56).

$$\mu_v^{(t)} = \sigma\left(W_{self}^{(t)} x_v + W_{neigh}^{(t)} \sum_{v \in N(v)} \mu_v^{(t-1)}\right) \tag{7.56}$$

Таким образом, на каждой итерации обновленное вложение гильбертова пространства для вершины v является функцией вложений его соседей, а также входных данных его объектов. И, как и в случае с базовым GNN, параметры $W_{self}^{(t)}$ и $W_{neigh}^{(t)}$ процесса обновления могут быть обучены с помощью градиентного спуска при любой произвольной задачи.

GNNS и PGM в более общем плане

И так базовая модель GNN может быть получена как встроенная форма вывода среднего поля. Однако существуют и другие способы подключения PGM и GNNS. Например, различные варианты передачи сообщений могут быть получены на основе различных алгоритмов приблизительного вывода. В целом, связи между GNN и более традиционным статистическим реляционным обучением — это богатая область с огромным потенциалом для новых разработок.

7.3 Графовые нейронные сети и изоморфизм графов

Теперь видно, как GNN может быть мотивировано на основе связей с графической обработкой сигналов и вероятностными графическими моделями. Обратим внимание на мотивацию GNN, основанную на связях с проверкой изоморфизма графов. Как и в предыдущих разделах, здесь снова увидим, как базовый GNN может быть получен как нейросетевая вариация существующего алгоритма – в данном случае алгоритма изоморфизма Вайсфайлера-Лемана (Weisfieler-Lehma – WL). Однако, в дополнение для мотивации подхода GNN подключение к

тестированию на изоморфизм также предоставит инструменты для формального анализа мощности GNN.

Изоморфизм графа

Проверка на изоморфизм графов является одной из наиболее фундаментальных и хорошо изученных задач в теории графов. Учитывая пару графов $\mathcal{G}_1$ и $\mathcal{G}_2$, цель тестирования изоморфизма графов состоит в том, чтобы объявить, являются ли эти два графа изоморфными или нет. В интуитивном смысле изоморфность двух графов означает, что они по существу идентичны. Изоморфные графы представляют собой точно такую же структуру графа, но они могут отличаться только порядком расположения вершин в соответствующих им матрицах смежности. Формально, если есть два графа с матрицами смежности A_1 и A_2, а также узловые функции X_1 и X_2, говорим, что два графа изоморфны тогда и только тогда, когда существует матрица перестановок P такая, что (7.57)

$$PA_1P^\intercal = A_2 \; and \; PX_1 = X_2 \tag{7.57}$$

Важно отметить, что изоморфные графы действительно идентичны с точки зрения их базовой структуры. Порядок вершин в матрице смежности – это произвольное решение, которое необходимо принять, когда представляем граф с использованием алгебраических объектов (например, матриц), но этот порядок не имеет никакого отношения к структуре самого базового графа.

Несмотря на свое простое определение, проверка на изоморфизм графа является принципиально сложной задачей. Например, наивный подход к проверке на изоморфизм включал бы следующую задачу оптимизации (7.58):

$$\min_{P \in \mathcal{P}} \|PA_1P^\intercal - A_2\| + \|PX_1 = X_2\| = 0 \tag{7.58}$$

Эта оптимизация требует поиска по полному набору матриц перестановок $\mathcal{P}$, чтобы оценить, существует ли единственная матрица перестановок P, которая приводит к эквивалентности между двумя графами. Вычислительная сложность этого наивного подхода огромна при $O(|V|!)$, и на самом деле не известно ни одного алгоритма полиномиального времени, который правильно проверял бы изоморфизм для общих графов.

Тестирование изоморфизма графа формально называется NP-неопределенным (NP-indeterminate – NPI). Известно, что он не является NP-полным, но нет общих алгоритмов полиномиального времени известны своей проблемой. Целочисленная факторизация - еще одна хорошо известная проблема, которая, как предполагается, относится к классу NPI. Однако существует множе-

ство практических алгоритмов проверки изоморфизма графов, которые работают на широких классах графов, включая алгоритм WL.

Изоморфизм графа и репрезентативная способность

Теория тестирования изоморфизма графов особенно полезна для изучения представления графов. Это дает возможность количественно оценить репрезентативную силу различных подходов к обучению. Если есть алгоритм – например, GNN – который может генерировать представления $z_{\mathcal{G}} \in \mathbb{R}^d$ для графов, то можно количественно оценить мощность этого алгоритма обучения, спросив, насколько полезными были бы эти представления для проверки изоморфизма графов. В частности, учитывая изученные представления $z_{\mathcal{G}1}$ и $z_{\mathcal{G}2}$ для двух графов, «идеальный» алгоритм обучения имел бы это (7.59)

$$z_{\mathcal{G}1} = z_{\mathcal{G}2} \text{тогда и только тогда, когда } \mathcal{G}_1 \text{ изоморфна } \mathcal{G}_2 \quad (7.59)$$

Идеальный алгоритм обучения генерировал идентичные вложения для двух графов тогда и только тогда, когда эти два графа действительно были изоморфны. Конечно, на практике ни один алгоритм обучения представлению не будет «идеальным» (если только $P = NP$). Тем не менее, количественная оценка мощности алгоритма обучения представлению путем подключения его к тестированию изоморфизма графа очень полезна.

Несмотря на то, что проверка изоморфизма графа в целом неразрешима, тогда делаем несколько мощных и хорошо понятных подходов для тестирования приближенного изоморфизма, и можно получить представление о возможностях GNNs, сравнив их с этими подходами.

Алгоритм Вайсфайлера-Лемана

Наиболее естественный способ подключения GNNs к тестированию изоморфизма графов основан на подключениях к семейству алгоритмов Вайсфайлера-Лемана (WL). Было обсуждение алгоритма WL в контексте ядер графов. Однако, подход WL более широко известен как одна из наиболее успешных и хорошо понятых структур для тестирования приближенного изоморфизма. Простейшая версия алгоритма WL, широко известная как 1—WL, состоит из следующих шагов:

1. Двум графам $\mathcal{G}_1$ и $\mathcal{G}_2$, присваиваем начальную метку $l^{(0)}_{\mathcal{G}_i}(v)$ для каждой вершины на каждом графе. В большинстве графов эта метка является просто степенью вершины, т.е. $l^{(0)}(v) = d_v \; \forall \mathrm{v} \in \mathrm{V}$, но если есть дискретные объекты

(т.е. один горячий объект x_v), связанные с вершинами, можно использовать эти функции для определения начальных меток.

2. Далее итеративно присваиваем новую метку каждому вершине в каждом графе путем хэширования множества текущих меток в окрестности вершины, а также текущей метки вершины:

$$l_{\mathcal{G}_i}^{(i)}(v) = HASH\left(l_{\mathcal{G}_i}^{(i)}(v), \left\{\left\{l_{\mathcal{G}_i}^{(i-1)}(v) \forall u \in \mathcal{N}(v)\right\}\right\}\right) \tag{7.60}$$

где двойные фигурные скобки используются для обозначения мультимножества, а хэш-функция сопоставляет каждый уникальный мультимножество с уникальной новой меткой.

3. Повторяем шаг 2 до тех пор, пока метки для всех вершин в обоих графах не сойдутся, т. е. пока мы не достигнем итерации K, где $l_{\mathcal{G}_i}^{(K)}(v) = l_{\mathcal{G}_i}^{(K-1)}(v), \forall v \in V_j, j = 1,2$

4. Наконец, строим мультинаборы $L_{\mathcal{G}_j} = \left\{\left\{l_{\mathcal{G}_j}^{(i)}(v), \forall v \in V_j, i = 0, \dots, K-1\right\}\right\}$ суммируя все метки вершин в каждом графе, объявляем $\mathcal{G}_1$ и $\mathcal{G}_2$ изоморфными, когда мультинаборы для обоих графов идентичны, т. е. тогда и только тогда, когда $L_{\mathcal{G}_1} = L_{\mathcal{G}_2}$.

На рисунке 7.2 иллюстрируется пример процесса маркировки WL на одном графе. На каждой итерации каждый вершина собирает мультимножество меток в своем локальном окружении и обновляет свою собственную метку на основе этого мультимножества. После K итераций этого процесса маркировки у каждого вершины есть метка, которая суммирует структуру его окрестности K-шага, и коллекция этих меток может быть использована для характеристики структуры всего графа или подграфа.

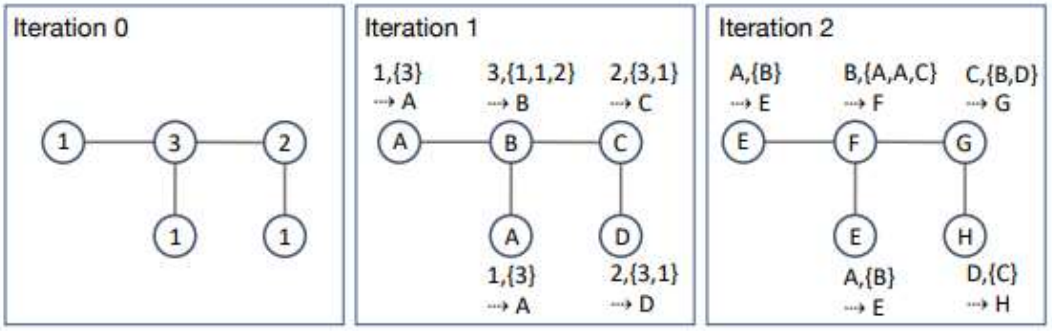

Рис. 7.2. Пример итеративной процедуры маркировки WL на одном графе.

Известно, что алгоритм WL сходится не более чем за $|V|$ итераций и, как известно, успешно проверяет изоморфизм для широкого класса графов. Однако существуют хорошо известные случаи, когда тест завершается неудачей, такие как простой пример, проиллюстрированный на рисунке 7.3.

Рис. 7.3: Пример двух графов, которые нельзя отличить по базовому алгоритму WL.

GNNs и алгоритм WL. Существуют четкие аналогии между алгоритмом WL и подходом GNN с передачей нейронных сообщений. В обоих подходах итеративно агрегируем информацию из окрестностей локальных вершин и используем эту агрегированную информацию для обновления представления каждого вершины. Ключевое различие между двумя подходами заключается в том, что алгоритм WL агрегирует и обновляет дискретные метки (используя хэш-функцию), в то время как модели GNN агрегируют и обновляют вложения вершин с использованием нейронных сетей. Фактически, GNN были мотивированы и выведены как непрерывный и дифференцируемый аналог алгоритма WL.

Взаимосвязь между GNN и алгоритмом WL может быть формализовано в следующей теореме:

Теорема. Определите GNN для передачи сообщений (MP-GNN) как любую GNN, состоящую из K слоев передачи сообщений в следующей форме (7.61):

$$h_u^{(k+1)} = UPDATE^{(k)}\left(h_u^{(k)}, AGGREGATE^{(k)}\left(\left\{h_u^{(k)}, \forall u \in \mathcal{N}(v)\right\}\right)\right) \quad (7.61)$$

где $AGGREGATE$ – дифференцируемая инвариантная функция перестановки, а $UPDATE$ – дифференцируемая функция. Далее, предположим, что только входы дискретных признаков на начальном слое, т. е. $h_u^{(0)} = x_u \in \mathbb{Z}^d, \forall u \in V$. Далее условие $h_u^{(K)} \neq h_v^{(K)}$ выполняется только если вершины u и v имеют разные метки после K итераций алгоритма WL.

Интуитивно говоря, рассматриваемая теорема утверждает, что GNN не более эффективны, чем алгоритм WL, когда у нас есть дискретная информация в качестве признаков вершины. Если алгоритм WL назначает одну и ту же метку двум вершинам, то любая GNN, передающая сообщения, также назначит одно и то же вложение этим двум вершинам. Этот результат по маркировке вершин также распространяется на проверку изоморфизма. Если тест WL не может различить два графа, то MP-GNN также не может различить эти два графа. Также можно показать более положительный результат в другом направлении:

Теорема. Существует MP GNN такая, что $h_u^{(K)} = h_v^{(K)}$ тогда и только тогда, когда две вершины u и v имеют одну и ту же метку после K итераций алгоритма WL.

Эта теорема утверждает, что существуют GNN с передачей сообщений, столь же мощные, как тест WL.

Две приведенные выше теоремы утверждают, что GNN с передачей сообщений не уступают по мощности алгоритму WL и что существуют GNN с передачей сообщений, столь же мощными, как алгоритм WL. Итак, какие GNN фактически получают эту теоретическую верхнюю границу? Интересно, что базовой GNN, достаточно, чтобы удовлетворить эту теорию. В частности, если определим обновления передачи сообщений следующим образом (7.62):

$$h_u^{(k)} = \sigma\left(W_{self}^{(k)} h_u^{(k-1)} + W_{neigh}^{(k)} \sum_{v \in N(v)} h_v^{(k-1)} + b^{(k)}\right) \qquad (7.62)$$

тогда этого GNN достаточно, чтобы соответствовать мощности алгоритма WL.

Однако большинство других моделей GNN, не так эффективны, как алгоритм WL. Формально, чтобы быть такими же мощными, как алгоритм WL, функции $AGGREGATE$ и $UPDATE$ должны быть инъективными. Это означает, что операторы $AGGREGATE$ и $UPDATE$ должны сопоставлять каждый уникальный ввод с уникальным выходным значением, чего нельзя сказать о многих рассмотренных моделях. Например, функции $AGGREGATE$, которые используют (взвешенное) среднее значение вложений соседей, не являются инъективными; если все соседи имеют одинаковое вложение, то (взвешенное) среднее значение не сможет различать входные наборы разных размеров.

Исследователи подробно обсуждают относительную мощность различных архитектур GNN. Они также определяют «минимальную» модель GNN, которая имеет мало параметров, но по-прежнему столь же мощна, как и алгоритм WL.

Они называют эту модель сетью изоморфизма графов (Graph Isomorphism Network – GIN), и она определяется следующим обновлением (7.63):

$$h_u^{(k)} = MLP^{(k)}\left(\left(1+\varepsilon^{(k)}\right)h_u^{(k-1)} + \sum_{v\in N(v)} h_v^{(k-1)}\right) \quad (7.63)$$

где $\varepsilon^{(k)}$ — обучаемый параметр.

Таким образом, отмечен важный отрицательный результат, касающийся GNN с передачей сообщений (MP-GNN): эти модели не более эффективны, чем алгоритм WL. Однако, несмотря на этот результат, исследование того, как можно сделать GNN, которые будут доказуемо более мощными, чем алгоритм WL, является актуальной темой для исследования.

Реляционный пул

Один из способов мотивировать доказуемо более мощные GNN — рассмотреть случаи отказа алгоритма WL. Например, на рис. 7.3 можно видеть, что алгоритм WL — и, следовательно, все MP-GNN — не может отличить связные 6 вершин от набора из двух треугольников. С точки зрения передачи сообщений это ограничение связано с тем, что операции $AGGREGATE$ и $UPDATE$ не могут определить, когда у двух вершин один и тот же сосед. В примере на рис. 7.3 каждая вершина может сделать вывод из операций передачи сообщений, что у её есть два соседа степени 2, но этой информации недостаточно, чтобы определить, связаны ли соседи вершины друг с другом. Это ограничение – не просто крайний случай, показанный на рис. 7.3. Подходы к передаче сообщений обычно не могут идентифицировать замкнутые треугольники на графе, что является критическим ограничением.

Чтобы устранить это ограничение, рассматривается вопрос о дополнении MP-GNN уникальными функциями идентификатора вершины. Если используется $MP-GNN(A,X)$ для обозначения произвольного MP-GNN на входной матрице смежности A и вершинах X, то добавление идентификаторов вершин эквивалентно изменению MP-GNN следующим образом (7.64):

$$MP-GNN(A, X\oplus I) \quad (7.64)$$

где I – единичная матрица размера $d\times d$, а $\oplus$ обозначает конкатенацию матриц по столбцам. Другими словами, просто добавляем уникальную функцию одноразового индикатора для каждой вершины. В случае с рис. 7.3 добавление уникаль-

ных идентификаторов вершин позволило бы MP-GNN идентифицировать, когда две вершины имеют общего соседа, что сделало бы два графа различимыми.

Однако, к сожалению, эта идея добавления идентификаторов вершин не решает проблему. На самом деле, добавляя уникальные идентификаторы вершин, вводится новая и столь же проблематичная проблема: MP-GNN больше не является эквивариантным по перестановкам. Для стандартного MP-GNN имеем это (7.65)

$$P\big(\mathrm{MP}-\mathrm{GNN}(A,X)\big)=\mathrm{MP}-\mathrm{GNN}(PAP^{\mathsf{T}},PX) \tag{7.65}$$

где $P\in\mathcal{P}$ – произвольная матрица перестановок. Это означает, что стандартные MP-GNN эквивариантны по перестановкам. Если переставить матрицу смежности и признаки вершин, то результирующие вложения вершин будут просто переставлены эквивалентным образом. Однако MP-GNN с идентификаторами вершин не инвариантны к перестановкам, поскольку в общем случае (7.66)

$$P\big(\mathrm{MP}-\mathrm{GNN}(A,X\oplus I)\big)\neq\mathrm{MP}-\mathrm{GNN}(PAP^{\mathsf{T}},(PX)\oplus I) \tag{7.66}$$

Ключевая проблема заключается в том, что назначение уникального идентификатора каждой вершине фиксирует определенный порядок вершин для графа, что нарушает эквивариантность перестановок. Чтобы решить эту проблему, предлагается подход реляционный пулинг Relational Pooling (Relational Pooling – RP), который включает в себя маргинализацию всех возможных перестановок вершин. Для любого MP-GNN расширение RP этого GNN задается выражением (7.67)

$$RP-\ GNN(A,X)=\sum_{P\in\mathcal{P}}\mathrm{MP}-\mathrm{GNN}(PAP^{\mathsf{T}},(PX)\oplus I) \tag{7.67}$$

Суммирование всех возможных матриц перестановок $P\in\mathcal{P}$ восстанавливает инвариантность перестановок, и сохраняencz дополнительная репрезентативная способность добавления уникальных идентификаторов вершин. Фактически доказывается, что RP-расширение MPGNN может различать графы, неразличимые для алгоритма WL.

Ограничение подхода RP заключается в его вычислительной сложности. Оценка уравнения (7.67) имеет временную сложность $O(|V|!)$, что на практике невозможно. Однако, несмотря на это ограничение, показывается, что подход RP может достигать хороших результатов, используя различные приближения для

снижения стоимости вычислений (например, выборка подмножества перестановок).

Тест k-WL и k-GNN

Рассмотренный подход реляционного объединения (RP) может создавать модели GNN, которые доказуемо более эффективны, чем алгоритм WL. Однако подход RP имеет два ключевых ограничения [29]:

1. Полный алгоритм трудновыполним в вычислительном отношении.
2. Известно, что RP-GNN более мощны, чем тест WL, но у нас нет возможности охарактеризовать, насколько они мощнее.

Чтобы устранить эти ограничения, в нескольких подходах рассматривалось улучшение GNN путем адаптации обобщений алгоритма WL.

Алгоритм WL, на самом деле является простейшим из того, что известно как семейство алгоритмов k-WL. Следовательно, алгоритм WL, который представлен ранее, часто называют алгоритмом 1-WL, и его можно обобщить до алгоритма k-WL для $k > 1$. Ключевая идея алгоритмов k-WL заключается в том, что помечаем подграфы размера k, а не отдельные вершины. Алгоритм k-WL генерирует представление графа G посредством следующих шагов:

1. Пусть $s = (u_1, u_2, \dots, u_k) \in V^k$ – набор, определяющий подграф размера k, где $u_1 \neq u_2 \neq \dots \neq u_k$. Определим начальную метку $l_{\mathcal{G}}^{(0)}(s)$ для каждого подграфа классом изоморфизма этого подграфа (т. е. два подграфа получают одну и ту же метку тогда и только тогда, когда они изоморфны).
2. Затем итеративно присваивается новая метка каждому подграфу, хешируя мультимножество текущих меток в окрестности этого подграфа: $l_{\mathcal{G}}^{(i)}(s) = HASH\left(\left\{\left\{l_{\mathcal{G}}^{(i-1)}(s'), \forall s' \in N_j(s), j = 1, \dots k\right\}\right\}, l_{\mathcal{G}}^{(i-1)}(s)\right)$

где j-я окрестность подграфа определяется как (7.68) [29]:

$$N_j(s) = \left\{\left\{\left(u_1, \dots, u_{j-1}, v, u_{j+1}, \dots, u_k\right), \forall v \in V\right\}\right\} \tag{7.68}$$

3. Повторяем шаг 2 до тех пор, пока метки для всех подграфов не сойдутся, т. е. пока не будет достигнута итерация K, где $l_{\mathcal{G}}^{(K)}(s) = l_{\mathcal{G}}^{(K-1)}(s)$ для каждого k-набора вершин $s \in V^k$.
4. Наконец, мы строим мультимножество, $L_{\mathcal{G}} = \left\{\left\{l_{\mathcal{G}}^{(i)}(v), \forall s \in V^k, i = 0, \dots, K-1\right\}\right\}$ суммирующее все метки подграфов в графе.

Как и в случае алгоритма 1-WL, суммарное мультимножество $L_{\mathcal{G}}$, сгенерированное алгоритмом k-WL, можно использовать для проверки изоморфизма гра-

фов путем сравнения мультимножеств двух графов. Существуют также методы ядра графа, основанные на тесте k-WL, которые аналогичны WL-ядру.

Важным фактом в алгоритме k-WL является то, что он вводит иерархию репрезентативной способности. Для любого $k \geq 2$ имеем, что тест $(k+1)$-WL строго более мощный, чем тест k-WL. (Однако обратите внимание, что запуск k-WL требует решения изоморфизма графов для графов размера k, поскольку шаг 1 в алгоритме k-WL требует маркировки графов в соответствии с их типом изоморфизма. Таким образом, выполнение k-WL для $k > 3$, как правило, сложно с вычислительной точки зрения.) Поэтому, возникает естественный вопрос: можно ли спроектировать GNN, которые столь же эффективны, как k-WL тест для $k > 2$? И, конечно, очевидным было бы проектирование GNN по аналогии с алгоритмом k-WL.

Разработан подход k-GNN, который является дифференцируемым и непрерывным аналогом алгоритма k-WL. Тогда k-GNN изучают вложения, связанные с подграфами, а не с вершинами, и передача сообщений происходит в соответствии с окрестностями подграфа (например, как определено в уравнении 7.68). Доказывается, что k-GNN могут быть такими же выразительными, как алгоритм k-WL. Однако существуют также серьезные вычислительные проблемы как для теста k-WL, так и для k-GNN, поскольку временная сложность передачи сообщений комбинаторно возрастает по мере увеличения k. Эти вычислительные проблемы требуют различных приближений, чтобы сделать k-GNN пригодными для использования на практике.

Инвариантные и эквивариантные GNN k-го порядка

Еще одно направление работы, мотивированное идеей создания GNN, столь же мощных, как тест k-WL, − это инвариантные и эквивариантные GNN. Важнейшим аспектом GNN с передачей сообщений (MP-GNN, как определено в теореме) является то, что они эквивалентны перестановкам вершин, что означает, что (7.69) [29]:

$$P\big(\mathrm{MP}-\mathrm{GNN}(A,X)\big) = \mathrm{MP}-\mathrm{GNN}(PAP^{\mathsf{T}}, PX) \qquad (7.69)$$

Это справедливо для любой матрицы перестановок $P \in \mathcal{P}$. Это равенство говорит о том, что перестановка входных данных в MP-GNN просто приводит к тому, что матрица вложений выходных вершин переставляется аналогичным образом/

В дополнение к понятию эквивариантности также можуj определить аналогичное понятие инвариантности перестановок для MP-GNN на уровне графа. В

частности, MP-GNN можно расширить с помощью функции $POOL : R^{|V|\times d} \rightarrow R$, которая отображает матрицу вложений изученных вершин $Z \in R^{|V|\times d}$ во вложение $z_{\mathcal{G}} \in R^{d}$ весь граф. В этом варианте уровня графа видно, что MP-GNN инвариантны к перестановкам, т.е. (7.70)

$$POOL\big(\text{MP}-\text{GNN}(PAP^{\intercal}, PX)\big) = POOL\big(\text{MP}-\text{GNN}(A, X)\big) \tag{7.70}$$

это означает, что объединенное вложение на уровне графа не изменяется при использовании другого порядка вершин.

Основываясь на этой идее инвариантности и эквивариантности, предлагается общую форму моделей, подобных GNN, на основе перестановочных эквивариантных/инвариантных тензорных операций. Предположим, есть тензор $x \in \mathbb{R}^{|V|^{k}\times d}$порядка $(k+1)$, где предполагается, что первые k каналов/мод этого тензора пронумерованы вершинами графа. Используется обозначение $P \star X$ для обозначения операции перестановки первых k каналов этого тензора в соответствии с матрицей перестановки вершин P. Затем можно определить линейный эквивариантный слой как линейный оператор (т. е. тензор) $L\colon \mathbb{R}^{|V|^{k_1}\times d_1} \rightarrow \mathbb{R}^{|V|^{k_2}\times d_2}$ (7.71):

$$L \times (P \star X) = P \star (L \times X), \forall P \in \mathcal{P} \tag{7.71}$$

где используется $\times$ для обозначения обобщенного тензорного произведения. Аналогичным образом можно определить инвариантные линейные операторы как тензоры L, удовлетворяющие следующему равенству:

$$L \times (P \star X) = L \times X, \forall P \in \mathcal{P} \tag{7.72}$$

Обратите внимание, что как эквивариантные, так и инвариантные линейные операторы могут быть представлены в виде тензоров, но они имеют разную структуру. В частности, эквивариантный оператор $L\colon \mathbb{R}^{|V|^{k}\times d_1} \rightarrow \mathbb{R}^{|V|^{k}\times d_2}$соответствует тензору $L \in \mathbb{R}^{|V|^{2k}\times d_1\times d_2}$, имеющему $2k$ каналов, пронумерованных вершинами (т. е. вершин каналов в два раза больше, чем входных). С другой стороны, инвариантный оператор $L\colon \mathbb{R}^{|V|^{k}\times d_1} \rightarrow \mathbb{R}^{|V|^{d_2}}$соответствует тензору $L\colon \mathbb{R}^{|V|^{k_i}\times d_1\times d_2}$, имеющему k каналов, пронумерованных вершинами (т. е. столько же, сколько ввод). Интересно, что при таком тензорном взгляде на линейные операторы эквивариантные (уравнение

7.71) и инвариантные (уравнение 7.72) свойства можно объединить в одно требование, чтобы тензор L был фиксированной точкой при перестановках вершин (7.73):

$$P \star L = L, \forall P \in \mathcal{P} \tag{7.73}$$

Другими словами, для заданного входа $x \in \mathbb{R}^{|V|^{k_i} \times d_2}$, как эквивариантные, так и инвариантные линейные операторы будут соответствовать тензорам, которые удовлетворяют фиксированной точке в уравнении (7.73), но количество каналов в тензоре будет различаться в зависимости от того, является ли он эквивариантным или инвариантным оператором.

Тензоры, удовлетворяющие фиксированной точке в уравнении (7.73), могут быть построены как линейная комбинация набора фиксированных базисных элементов. В частности, любой тензор L порядка l, удовлетворяющий уравнению (7.73), может быть записан как (7.74):

$$L = \beta_1 B_1 + \beta_2 + \cdots + \beta_{b(l)} B_{b(l)} \tag{7.74}$$

где $B_i -$ набор фиксированных базисных тензоров, $\beta_i \in R$ – действительные веса, а $b(l)$ – l-ое число Белла. Построение и вывод этих базисных тензоров связаны с математикой и тесно связаны с теорией чисел Белла из комбинаторики. Однако ключевым фактом и проблемой является то, что количество необходимых базисных тензоров растет с l-ым числом Белла, которое представляет собой экспоненциально возрастающий ряд.

Число Белла B_n равно количеству разбиений множества из n элементов на произвольное количество непустых подмножеств, которые не пересекаются. Очевидно, что $B_0 = 1$, так как существует только одно разбиение пустого множества. Например$B_3 = 5$, так как существует 5 возможных разбиений множества $\{a, b, c\}$ из трех элементов: $\{\{a\}, \{b\}, \{c\}\}, \{\{a, b\}, \{c\}\}, \{\{a, c\}, \{b\}\}, \{\{a\}, \{b, c\}\}, \{\{a, b, c\}\}$.

Используя эти линейные эквивариантные и инвариантные слои, определяется инвариантная модель GNN k-порядка на основе следующего состава функций (7.75):

$$MLP \circ L_0 \circ \sigma \circ L_1 \circ \sigma L_2 \ldots \sigma \circ L_m \times X \tag{7.75}$$

В этой композиции применено m эквивариантных линейных слоев $L_1, \ldots, L_m$, где $L_i: L: \mathbb{R}^{|V|^{k_i} \times d_1} \to \mathbb{R}^{|V|^{k_{i+1}} \times d_2}$ с $max_i\, k_i = k$ и k_1=2. Между каждым из этих линейных эквивариантных слоев применяется поэлементная нелинейность, обозначаемая σ. Предпоследней функцией в композиции является инвариантный линейный слой L_0, за которым следует многослойный персептрон (MLP) в качестве последней функции в композиции. Входными данными для инварианта GNN k-го порядка является тензор $x \in \mathbb{R}^{|V|^2 \times d}$, где первые два канала соответствуют матрице смежности, а остальные каналы кодируют признаки/метки начальной вершины.

Этот подход называется k-порядком, потому что эквивариантные линейные слои включают тензоры, которые имеют до k различных каналов. Однако наиболее важно то, что модели порядка k, следующие уравнению 7.75, столь же эффективны, как и алгоритм k-WL. Однако, как и в случае с k-GNN, обсуждавшимися в предыдущем разделе, построение инвариантных моделей k-го порядка для $k > 3$, как правило, трудновыполнимо с вычислительной точки зрения.