6. ЛЕКЦИЯ – Графовые нейронные сети на практике

И так было представлено несколько архитектур графовых нейронных сетей (GNN). Однако не обсуждалось, как оптимизируются эти архитектуры и какие виды функций потерь и регуляризации обычно используются. Теперь обратим внимание на некоторые из этих практических аспектов GNN. Обсудим некоторые репрезентативные приложения и то, как GNN обычно оптимизируются на практике, включая обсуждение неконтролируемых методов предварительного обучения, которые могут быть особенно эффективными. Также представим общие методы, используемые для регуляризации и повышения эффективности GNN.

6.1 Приложения и функции потерь

В подавляющем большинстве современных приложений GNN используются для одной из трех задач: классификация вершин, классификация графов или предсказание отношений. Как обсуждалось, эти задачи отражают большое количество реальных приложений, таких как предсказание того, является ли пользователь ботом в социальной сети (классификация вершин), предсказание свойств на основе структур молекулярного графа (классификация графов) и содержание рекомендация в онлайн-платформах (предсказание отношений). Теперь кратко опишем, как эти задачи преобразуются в конкретные функции потерь для GNN, а также обсудим, как можно предварительно обучить GNN без учителя, чтобы повысить производительность этих задач.

Далее будем использовать $z_u \in \mathbb{R}^d$ для обозначения вывода вложения вершины конечным слоем GNN, и будем использовать $z_G \in \mathbb{R}^d$ для обозначения вывода вложения на уровне графа функцией объединения. Любой из подходов GNN, в принципе может быть использован для создания этих вложений. В общем, определим функции потерь на вложениях z_u и z_G и будем считать, что градиент потерь распространяется обратно через параметры GNN с использованием стохастического градиентного спуска или одного из его вариантов.

Графовые нейросети для классификации вершин.

Классификация вершин — одна из самых популярных тестовых задач для GNN. Например, в период с 2017 по 2019 год, когда методы GNN начали приобретать все большее значение в машинном обучении, в исследованиях GNN доминировали датасеты Cora, Citeseer и Pubmed, которые были популяризированы. Эти базовые уровни включали классификацию категории или темы научных статей на основе их положения в сети цитирования, с языковыми особенностями вершин (например, векторами слов) и очень небольшим количеством положительных примеров, приведенных в каждом классе (обычно менее 10% вершин).

Стандартным способом применения GNN к такой задаче классификации вершин является обучение GNN полностью контролируемым образом, где мы определяем потери при помощи функции классификации softmax и потерь с отрицательным логарифмическим правдоподобием (6.1):

$$L = \sum_{u \in V_{train}} -log\big(softmax(z_u, y_u)\big) \tag{6.1}$$

Здесь предполагаecz, что $y_u \in Z^c$ – однократный вектор, указывающий класс обучающей вершины $u \in V_{train}$. Например, в настройках сети цитирования y_u указывает тему статьи u. Используется $softmax(z_u, y_u)$ для обозначения прогнозируемой вероятности того, что вершина принадлежит классу y_u, вычисленной с помощью функции softmax (6.2):

$$softmax(z_u, y_u) = \sum_{i=1}^{c} y_u[i] \frac{e^{z_u^T w_i}}{\sum_{j=1}^{c} e^{z_u^T w_j}} \tag{6.2}$$

где $w_i \in \mathbb{R}^d, i = 1, \ldots, c$ — обучаемые параметры. Существуют и другие варианты контролируемых потерь вершин, но обучение GNN контролируемым образом на основе потерь в уравнении (6.1) является одной из наиболее распространенных стратегий оптимизации для GNN.

Контролируемые, полуконтролируемые, трансдуктивные и индуктивные вершины. Необходимо обратить внимание, что, параметр классификации вершин часто называют как контролируемым, так и полуконтролируемым. Одним из важных факторов при применении этих терминов является то, что используются разные вершины во время обучения GNN. Как правило, можно различать три типа вершин [20]:

1. Имеется набор обучающих вершин V_{train}. Эти вершины включены в операции передачи сообщений GNN, и они также используются для вычисления потерь, например, с помощью уравнения (6.1).

2. В дополнение к обучающим вершинам также могут быть трансдуктивные тестовые вершины, V_{trans}. Эти вершины не помечены и не используются в вычислении потерь, но эти вершины — и их инцидентные ребра — по-прежнему участвуют в операциях передачи сообщений GNN. Другими словами, GNN будет генерировать скрытые представления $h_u^{(k)}$ для вершин в $u \in V_{trans}$ во время операций передачи сообщений GNN. Однако окончательные вложения слоя z_u для этих вершин не будут использоваться при вычислении функции потерь.

3. Есть еще и индуктивные тестовые вершины, V_{ind}. Эти вершины не используются ни в вычислении потерь, ни в операциях передачи сообщений GNN во время обучения, а это означает, что эти вершины – и все их ребра – полностью незаметны, во время обучения GNN.

Термин «полууправляемый» применим в тех случаях, когда GNN тестируется на трансдуктивных тестовых вершинах, так как в этом случае GNN наблюдает за тестовыми вершинами (но не их метками) во время обучения. Термин индуктивная классификация вершин используется для обозначения условий, в которых тестовые вершины и все инцидентные им ребра совершенно не наблюдаются во время обучения. Примером индуктивной классификации вершин может быть обучение GNN на одном подграфе сети цитирования, а затем ее тестирование на полностью непересекающемся подграфе.

Графовые нейросети для классификации графов

Подобно классификации вершин, приложения по классификации на уровне графа популярны в качестве эталонных задач. Ядерные методы были популярны для классификации графов, и в результате некоторые из самых популярных ранних эталонных тестов для классификации графов были адаптированы из литературы по ядрам. Например, задачи связанные с классификацией свойств ферментов на основе представлений на основе графов. В этих задачах часто используется потеря классификации softmax, аналогичная уравнению (6.1), с той ключевой разницей, что потеря вычисляется с помощью вложений z_{G_i} на уровне графа по набору размеченных обучающих графов $T = \{G_1, \ldots, G_n\}$. В последние годы GNN также добились успеха в задачах регрессии с использованием графических данных, особенно в задачах, связанных с предсказанием молекулярных свойств (например, растворимости) на основе графических представлений молекул. В этих случаях стандартно использовать потери квадрата ошибки следующего вида (6.3):

$$L = \sum_{G_i \in \mathrm{T}} \left\| MLP\left(z_{G_i}\right) - y_{G_i} \right\|_2^2 \qquad (6.3)$$

где MLP – плотносвязная нейронная сеть с одномерным выходом, а $y_{G_i} \in \mathbb{R}$ – целевое значение для обучающего графа G_i.

Графовые нейросети для предсказания отношений. В то время как задачи классификации, безусловно, являются наиболее популярным применением GNN, они также используются в задачах прогнозирования отношений, таких как рекомендательные системы и завершение графа знаний. В этих приложениях стандартной практикой является использование функций потерь с встраиванием парных вершин. В принципе, GNN можно комбинировать с любой из парных функ-

ций потерь, обсуждаемых в предыдущем материале, с выходом GNN, заменяющим неглубокие вложения.

Предварительное обучение графовых нейронных сетей. Методы предварительного обучения стали стандартной практикой в глубоком обучении. В случае работы с графовыми нейронными сетями можно предположить, что предварительная подготовка с использованием одного из методов восстановления окружения может быть полезной стратегией для повышения производительности при последующей задаче классификации. Например, можно предварительно обучить сеть, чтобы восстановить недостающие ребра в графе и потом перейти к точной настройке.

Интересно, однако, что этот подход не принес большого успеха в контексте графовых нейронных сетей. Фактически, было обнаружено, что случайно инициализированная сеть равносильна предварительно обученной с реконструкцией окружения. Одна из гипотез, объясняющих это открытие, опирается на тот факт, что сигналы в графовых сетях изначально несут в себе достаточно много информации об окружающих связях. Соседние вершины в графе, как правило, будут иметь аналогичные вложения в сети из-за структуры передачи сигналов, поэтому принудительное восстановление потерянных соседей может быть просто избыточным.

Несмотря на этот отрицательный результат в отношении предварительной подготовки с потерями при реконструкции окружения, были достигнуты положительные результаты при использовании других стратегий предварительного обучения. Например, предлагают метод Deep Graph Infomax (DGI), который включает в себя максимизацию взаимной информации между вложениями вершин z_u и вложениями графов z_G. Формально такой подход оптимизирует следующие потери (6.4):

$$L = -\sum_{u \in V_{train}} \mathbb{E}_G log\big(D(z_u, z_G)\big) + \gamma \mathbb{E}_{\tilde{G}} log\big(1 - D(\tilde{z}_u, z_G)\big) \qquad (6.4)$$

Здесь z_u обозначает вложение вершины u, сгенерированное из сети на основе графа G, в то время как $\tilde{z}_u$ обозначает вложение вершины u, сгенерированное на основе искаженной версии графа G, обозначаемого $\tilde{G}$. Используется D для обозначения дискриминантной функции, которая представляет собой нейронную сеть, обученную предсказывать, произошла ли вершина из реального графа G или из поврежденной версии $\tilde{G}$. Обычно граф искажается путем изменения либо свойств вершин, либо матрицы смежности, либо и того, и другого каким-либо стохастическим образом (например, перетасовкой элементов матрицы свойств). Суть, стоящая за этой функцией потерь, заключается в том, что графовая нейрон-

ная сеть должна научиться генерировать вложения вершин, которые могут отличать реальный граф от его поврежденного аналога. Можно сказать, что эта оптимизация тесно связана с максимизацией взаимной информации между вложениями вершин z_u и вложением z_G на уровне графа.

Функция потерь, используемая в DGI (уравнение 6.4), является лишь одним примером из более широкого класса задач без контроля, которые имеют неоспоримый успех в контексте графовых нейронных сетей. Чтобы максимизировать взаимную информацию между различными уровнями представлений или различать реальные и искаженные пары вложений, неконтролируемые стратегии обучения обычно включают в себя графовые нейронные сети. Концептуально, эти подходы, которые также иногда используются в качестве вспомогательных средств во время контролируемого обучения, имеют сходство с подходами к предварительной подготовке «маскировки контента», которые открыли новые техники в сфере обработки естественного языка. Тем не менее, расширение и совершенствование подходов к предварительному обучению графовых нейронных сетей является открытой и активной областью исследований.

6.2 Проблемы эффективности и выборка вершин

В основном обсуждали графовые нейронные сети с точки зрения уравнений для передачи сообщений на уровне вершин. Однако, непосредственная реализация сети на основе этих уравнений может быть вычислительно неэффективной. Например, реализуя операции передачи сообщений для всех вершин, если несколько из них будут иметь общих соседей, то в итоге это может привести к избыточным вычислениям. Теперь обсудим некоторые стратегии, которые могут быть использованы для эффективной реализации графовых сетей.

Реализации на уровне графа

С точки зрения минимизации количества математических операций, необходимых для выполнения передачи сообщений, наиболее эффективной стратегией является реализация уравнений на графовом уровне. Эти уравнения обсуждали на уровне графа и ключевая идея заключается в реализации операций передачи сообщений на основе немногочисленных матричных умножений. Например, уравнение на уровне графа для базовой нейронной сети задается (6.5),

$$H^{(k)} = \sigma\left(AH^{(k-1)}W_{neigh}^{(k)} + H^{(k-1)}W_{self}^{(k)}\right) \tag{6.5}$$

где $H^{(t)}$ – матрица, содержащая вложения слоя k всех вершин графа. Преимущество использования этих уравнений заключается в том, что нет избыточных вы-

числений – т. е. вычисляем вложение $h_u^{(k)}$ для каждой вершины u ровно один раз при запуске модели. Однако ограничение этого подхода заключается в том, что он требует одновременной работы со всем графом и всеми свойствами вершин, что может быть невозможно из-за ограничений памяти. Кроме того, использование уравнений на уровне графа существенно ограничивает использование полномасштабного (в отличие от мини-пакетного) градиентного спуска.

Подвыборка и мини-пакетирование

Чтобы снизить объем потребляемой памяти и облегчить мини-пакетное обучение, можно работать с подмножеством вершин во время передачи сообщений. Математически можно представить это как выполнение уравнений графовой сети на уровне вершин для подмножества вершин графа в каждом пакете. Избыточных вычислений можно избежать с помощью тщательного проектирования, гарантирующего, что при запуске модели только один раз вычисляем вложение $h_u^{(k)}$ для каждой вершины u в пакете.

Проблема, однако, заключается в том, что нельзя просто запустить передачу сообщений по подмножеству вершин в графе без потери информации. Каждый раз, когда удаляется вершина, также удаляются ее ребра (т. е. изменяем матрицу смежности). Нет никакой гарантии, что выбор случайного подмножества вершин приведет к связному графу, к тому же выбор случайного подмножества вершин для каждого мини-пакета может оказать серьезное негативное влияние на производительность модели.

Предлагают стратегию для преодоления этой проблемы путем подбора окружения вершин. Основная идея состоит в том, чтобы сначала выбрать набор целевых вершин для пакета, а затем рекурсивно выполнить выборку их соседей, чтобы обеспечить сохранение связности графа. Чтобы избежать возможности выборки слишком большого количества вершин для пакета, предлагают создать подвыборку соседей каждой вершины, используя фиксированный размер выборки для повышения эффективности пакетных тензорных операций.

6.3 Совместное использование параметров и регуляризация

Регуляризация является ключевым компонентом любой модели машинного обучения. В контексте графовых нейронных сетей, как известно, хорошо работают многие стандартные подходы к регуляризации, включая L2, отсев и нормализацию слоев. Однако существуют также стратегии регуляризации, которые несколько спецефичны для настройки графовых сетей.

Совместное использование параметров между слоями. Одной из стратегий, которая часто используется в графовых сетях со многими уровнями передачи со-

общений, является совместное использование параметров. Основная идея состоит в том, чтобы использовать одни и те же параметры во всех функциях агрегирования и обновления. Как правило, этот подход наиболее эффективен в сетях с более чем шестью уровнями, и он часто используется в сочетании с функциями закрытого обновления.

Отсев ребер. Другая стратегия, специфичная для графовых сетей, известна как отсев ребер. В этой стратегии регуляризации случайным образом удаляются (или маскируются) ребра в матрице смежности во время обучения, интуитивно понимая, что это сделает сеть менее подверженной чрезмерной подгонке и более устойчивой к шуму в матрице смежности. Этот подход был особенно успешным при применении сетей к графам знаний. Также необходимо обратить внимание, что подходы к созданию подвыборки окружения, приводят к подобной регуляризации в качестве побочного эффекта, что делает ее очень распространенной стратегией в крупномасштабных приложениях графовых нейронных сетей.