ВМК

Задание 2. Графы с табличными признаками

Эффективные системы машинного обучения, 2024

Начало выполнения задания: 23 ноября 2024 года, 23:00.
Жесткий дедлайн: **15 декабря 2024 года, 23:59.**

Формулировка задания

Данное задание направлено на ознакомление с основными архитектурами для работы с графами, на обработку табличных признаков с помощью нейронных сетей, а также на автоматический поиск архитектур. В задании необходимо:

1. Обучить базовые модели для работы с графовыми данными (см. первый пункт раздела эксперименты)
2. Сравнить различные подходы к решению задач на графовых данных (см. соответствующие пункта раздела эксперименты)
3. Придумать новую архитектуру нейронной сети с помощью алгоритмов автоматического поиска архитектур (см. соответствующие пункта раздела эксперименты)
4. Оформить закрытый GitHub репозиторий с кодом экспериментов. Репозиторий должен содержать реализации всех использованных методов, ссылки на источники, если код не был написан самостоятельно, а также код/ноутбуки для получения таблиц и графиков, использованных в отчете.
5. Написать отчёт объемом до 5 страниц о проделанной работе (формат PDF). Отчет должен содержать следующие разделы: обзор методов предсказания работы с графами с табличными признаками, введение и постановка задачи, описание основных подходов к обработке графов, раздел с экспериментами, вывод (нужно выбрать, какой подход работает лучше с точки зрения качества работы)

Список экспериментов

Эксперименты этого задания необходимо проводить на любом из датасетов для классификации бенчмарка TabGraphs.

1. **Обучение базовых моделей.** Обучите стандартные архитектуры графовых нейронных сетей: GCN, GAT, GraphSAGE с гиперпараметрами из бенчмарка.
2. **Продвинутые графовые модели.** Воспользуйтесь модификациями классических архитектур графовых нейронных сетей из статьи Classic GNNs are Strong Baselines: Reassessing GNNs for Node Classification. Сделайте перебор параметров моделей. Удалось ли улучшить результат?
3. **Табличные модели.** Обучите три graph agnostic модели: MLP, MLP-PLR, TabM. Добавьте к ним: агрегацию признаков по соседям, DeepWalk эмбеддинги. Насколько качество работы graph agnostic моделей лучше/хуже чем качество GNN? Помогает ли табличным моделям знание графовой составляющей задачи? Насколько? В данном задании *настоятельно рекомендуется* выполнять хотя бы небольшой перебор параметров табличных моделей.
4. **Эмбеддинги для признаков для графовых моделей**. Добавьте PLR эмбеддинги к трем лучшим GNN архитектурам. Удалось ли улучшить модель?
5. **NAS.** Примените *любой* алгоритм NAS (за исключением перебора параметров из репозиториев табличных моделей) для поиска GNN и табличной архитектуры на выбранной задаче. Для этого сперва предложите некоторое пространство для поиска архитектур и объясните, почему вы видите его таким. Затем запустите на нем любимый алгоритм NAS.
6. **Сравнение полученных моделей**. Постройте сводную таблицу, содержащую подробный анализ обученных моделей с точки зрения качества, скорости инференса и потребляемую память.

Требования, советы и замечания

- В работе **разрешается** использовать код из TabGraphs, TabM.
- В работе разрешается использовать библиотеки для NAS. Однако, если вы их используете, а не реализуете алгоритм самостоятельно/пытаетесь прикрутить код из репозиториев, постарайтесь предложить максимально интересное пространство для поиска архитектуры.
- Задание выполняется в командах до 4х человек.
- В отчете должен быть раздел, в котором описан вклад каждого участника команды.
- Текст вклада должен соответствовать истории коммитов на GitHub.
- В задании **разрешается** пользоваться чужим кодом (за исключением пункта ниже), но нужно оставлять ссылки на источники.
- Брать код других команд, выполняющих данное задание **запрещается**.
- **Эксперименты должны быть воспроизводимы**. Поэтому рекомендуется зафиксировать все сиды, а также приложить файл для установки используемого окружения.
- Если какой-то пункт задания сделать не получилось, то об этом следует написать в отчете и сказать почему, это лучше, чем ничего не сделать.
- Генерация отчета с использованием LLM не запрещается, но лучше потом перечитать текст и попытаться скрыть следы использования БЯМ.
- Если берете готовый код, то желательно это делать из официальных источников, а не использовать случайный код из GitHub.
- В отчете должны быть ссылки на статьи, откуда вы взяли метод.
- Преподаватели оставляют за собой право в одностороннем порядке пополнить список запретов при обнаружении вопиющих случаев нечестного выполнения задания.
- Для работы с GNN вы, скорее всего, захотите использовать DGL. Эта библиотека весьма капризна, наиболее удобным способом ее установки является использование (официального докер образа от NVIDIA). Альтернативным способом установки рабочей конфигурации является следующее conda окружение:

```
conda create -n main "python<3.12"

conda install -n main -c pytorch -c nvidia -c dglteam/label/th24_cu121 -y \
    cuda-toolkit=12.1 \
    pytorch=2.4.0 \
    pytorch-cuda=12.1 \
    dgl=2.4 \
    tqdm=4.66 \
    pyyaml=6 \
    pydantic=2.5 \
    numpy=1.26 \
    pandas=2.2 \
    scikit-learn=1.5 \
    torchdata=0.7 \
    torchvision \
    torchaudio
```

Бонусная часть

Бонусные исследования могут сделать так, что проверяющие закроют глаза на проблемы в основной части. Чем больше бонусных исследований *качественно* сделано, тем больше косяков проверяющие готовы простить.

1. Добавьте к GNN моделям сжатие с использованием метода EXACT. Существует его официальная реализация под фреймворд pytorch geometric, поэтому предлагается провести следующий эксперимент: обучите одну и ту же модель с помощью DGL, PyG, PyG+EXACT. Проведите то же сравнение, что и в основной части.

2. Примените любые алгоритмы QAT и PAT к табличным моделям, сделайте вывод о том, как правильно сжимать модели для глубокого обучения на табличных данных.

3. Иногда, как бы это не было парадоксально, добавлением агрегации признаков по соседям/DeepWalk эмбеддингов позволяет лучше работать и GNN. Попробуйте улучшить графовые нейронные сети с помощью описанных приемов.

4. Протестируйте найденную архитектуру ну любом другом датасете из TabGraphs. Насколько она получилась лучше/хуже модели из статьи? Сделайте выводы об устойчивости найденной архитектуры