Отчёт по тестовому заданию.

Graph-less MLP

Тихомиров С. А.

Задачи:

1 блок. Выполнить четко поставленные технические задачи и ответить на вопросы.

2 блок. Проанализировать статью. Написать описание, ответить на вопросы, предложить свои варианты улучшения подхода.

Написать отчёт, предоставить последнюю версию проекта, команды запусков экспериментов, версии новых пакетов.

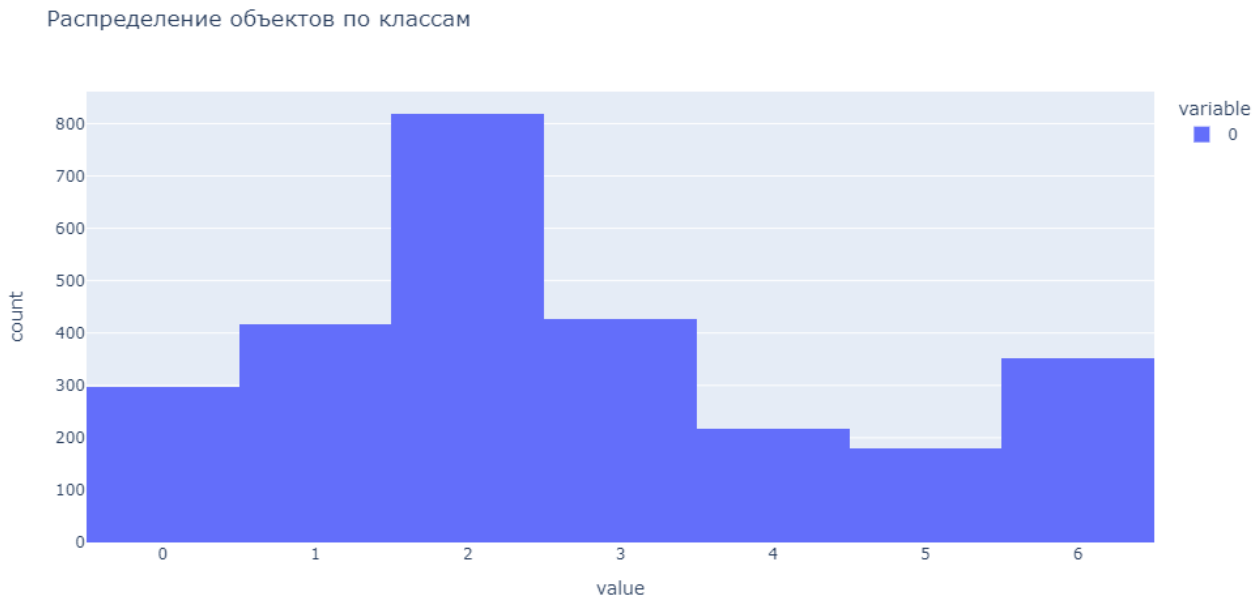
Блок 1: Разработка.

Цель: сравнить качество graph-based, mlp и graph-less_mlp моделей.

1. Metrics

Целесообразно ли использовать Accuracy в данном случае? Почему?

Accuracy – метрика, неприменимая в условиях несбалансированных целевых классов.



В целом есть дисбаланс классов, но для многоклассовой задачи несущественный, поэтому целесообразно использовать accuracy, в следствии интуитивно понятной и очевидной интерпретации.

Реализуйте другую, если видите в этом смысл.

Однако для контроля и дополнительной оценки были реализованы ещё 3 метрики Recall, Precision, F1-score. Реализация подсчёта метрик вынесена в отдельную функцию ?

Реализуйте выбор лучшей модели по статистикам на валидации. Добавьте необходимые зависимости от параметра seed.

Выбор лучшей модели реализован в model_select_experiment.py, в main.py добавлено фиксирование сида для конкретного запуска обучение, сохранения. Выбор модели осуществляется на основе некоторого кол-ва запусков моделей с различными сидами.

Сравните реализованные модели. Какие выводы можно сделать?

	Accuracy (std)	F1-Macro	F1-Micro
GNN	0.7785 (0.0256)	0.5822 (0.0711)	0.1093 (0.0282)
MLP	0.6681 (0.0295)	0.4919 (0.1712)	0.0443 (0.0102)
GLNN	0.6555 (0.0136)	0.4918 (0.0550)	0.0395 (0.0037)

2. Data

Какого размера датасет?

Датасет представляет из себя 2708 объектов - научных публикаций, отнесенных к одному из семи классов.

Какие и сколько фичей у элементов датасета?

В качестве признаков используются словарь вхождений слов и матрицу смежности. Словарь слов описывает каждый объект бинарным вектором, где 1 означает вхождение слова в статью вектором размера 1433. Матрица смежности описывает цитирования из исходной статьи, всего 5429 связи.

3. Class Dataset

Какие объекты и каких размерностей содержатся у выхода graph_collate_fn и от чего это зависит?

Функция graph_collate_fn приводит батч данных в стандартизованный по размерностям внутри батча и преобразует в torch.Tensor.

graph_collate_fn возвращает словарь с 4 элементами:

- labels – Тензор с метками классов размерности **размер батча**
- h – Тензор размерности **размер батча * количество фичей(1433)** с фичами объекта (см. пункт 2)
- h_nn – Тензор размерности **размер батча * количество фичей * максимальное количество связей у объекта в батче** с фичами соседей объектов.

  Каждому объекту ставится в соответствие тензор размерности **количество фичей * максимальное количество связей у объекта в батче**, с фичами других вершин у которых у данного объекта есть ребро.

  Количество связей у объекта в батче – 1 если нет соседей, max_size, если соседей больше чем определенная константа, иначе столько сколько связей.

  По батчу считается значение количества связей и остальным объектам паддится нулями до нужной размерности.
- nn_lenghts - Тензор размерности **размер батча** с количеством связей для каждого объекта.

Реализуйте функцию node_student_collate_fn. Какую информацию о графе из исходных данных использует student_mlp модель?

В данной реализации несмотря на приход soft labels, архитектура та же, что и у MLP, поэтому модель использует только вектор фичей объекта без связи конкретного объекта с другими. Реализация node_student_collate_fn аналогична node_collate_fn, только с прокидыванием ещё и soft_labels.

4. Class MLP

Сколько в сети обучаемых параметров?

Если использовать начальную конфигурацию - num_layers = 1, то

input_dim * output_dim + output_dim (нейроны смещения) =

$= 1433 * 7 + 7 = 10\,038$ обучаемых весов

Если использовать batch_norm, то 10 038 + 2*input_dim(именно input, потому что в реализации MLP сначала BN, потом слой) =

$= 10\,038 + 1433 * 2 = 12\,904$ обучаемых весов

Стоит ли установить num_layers=2? Почему?

В целом на первый взгляд казалось, что такое изменение может принести улучшение, т.к по факту MLP с num_layers = 1 – это просто 7 линейных моделей решающих задачи 1 vs ALL, однако результаты эксперимента приведенного в скрипте mlp_2_layers.py говорят обратное. Качество значительно сильно падает при добавлении одного скрытого слоя.

	Accuracy	F1-Macro	F1-Micro
MLP-1 layer	0.6681 (0.0295)	0.4919 (0.1712)	0.0443 (0.0102)
MLP-2 layer	0.2677 (0.0941)	0.0954 (0.0675)	0.0042 (0.0023)

Я связываю это с отсутствием достаточного количества данных и относительной простотой структуры, из-за чего модель даже с 1 скрытым слоем очень быстро переобучается и дает плохую оценку на валидационной выборке.

Стоит ли установить dropout_p=0.5? Почему?

Качество при добавлении дропаута немного увеличивается, модель становится более устойчивая, так что дропаут добавить стоит, но проблемы MLP с отставанием по качеству это не решает

	Accuracy	F1-Macro	F1-Micro
MLP	0.6681 (0.0295)	0.4919 (0.1712)	0.0443 (0.0102)
MLP + DROPOUT	0.6854 (0.0102)	0.5011 (0.1422)	0.0427 (0.0097)

5. Categorical features.

Возможно ли первую половину численных фичей обрабатывать как категориальные?

В целом возможно, но целесообразно ли, учитывая, что значений каждой категории у любой из фичей 1 или 0 – это вопрос.

Реализуйте подход с обработкой фичей как эмбедингов.

Было решено добавить эмбединг слои, каждая фича преобразуется в отдельный вектор, который в дальнейшем подается на вход моделям.

Решение с добавлением слоя эмбеддингов представлено в файле mlp.py, в виде закомментированного кода. Если убрать кавычки будет работать слой обработки категориальных фичей.

6. Bugs

Комментирование кода в предыдущем блоке задания не баг, но не самый качественный прием, можно было добавлять ещё один аргумент, однако я решил, не перенагружать код, в инициализации моделей и так определены 2 аргумента, которые не используются в одних и используются в других моделях.

Добавил докстринги и кое-где тайпхинты для лучшего чтения кода.

Блок 2: анализ литературы

Познакомьтесь со статьей GRAPH-LESS NEURAL NETWORKS: TEACHING OLD MLPS NEW TRICKS VIA DISTILLATION https://arxiv.org/pdf/2110.08727.pdf

1. Описать основные вклады статьи. Сформулируйте главный вывод 2-3 предложениями.

Статья представляет новый метод машинного обучения под названием Graph-less Neural Network (GLNN), который объединяет преимущества графовых нейронных сетей (GNNs) и нейронных сетей прямого распространения (MLPs), устраняя при этом зависимость от графов в процессе инфференса. Суть заключается в том, чтобы обучать MLP с использованием softmax меток от GNN, при этом на инференсе MLP будет не будет тратить время на прогон GNN. MLP преобразовывает информацию только от фичей вершин, а информация о ребрах, приходит от softmax label GNN. Автор утверждает, что это позволяет значительно повысить качество MLP на многих датасетах и даже достигнуть качества GNN на 6/7.

2. Осознать эксперименты на уровне «могу реализовать»

Есть ли явные отличия в коде от текущего проекта?

- Авторы делают стратифицированный сплит, в то время как в текущем проекте происходит рандомный сплит.
- В текущем проекте нет регуляризационного члена

Если ваши результаты отличаются от представленных в статье, попробуйте объяснить эту разницу

Представленные отличия мне кажутся незначительными для такой большой разницы, в качестве. Смущает качество MLP авторов в статье: 0.58 Accuracy, в то время как текущий проект с MLP, дает 0.668.

3. Сформировать содержательные соображения:

1. Первое чем бы занялся - поменял представление имеющихся фичей. В данный момент исходные данные – корпус документов, закодированных словарем, где в поле документа стоит 1 при каком-то слове, если он встречалось в документе, что в современных реалиях явно не лучший подход. Первым улучшение может быть подход bag of words, где в при наличии слова в документе в поле стоит не 1, а количество вхождений.

Дальше можно перекодировать статьи к TF-IDF представлению. Однако все эти подходы не используют порядок слов и их взаимоотношения, для решения этого можно попробовать формировать эмбеддинги одним из методов CBOW, skip-gram, Glove.

2. Можно найти дополнительную априорную информацию о статьях. Автор публикации, ВУЗ, страна, пол автора, год публикации, число слов, издательство, входит ли в скопус и.т.д

3. Подходы в прошлом пункте подходят только, если у нас имеется тексты или их представления, если остается работать только с тем, что есть, можно попробовать погруппировать фичи, или сделать какие-то статистики для каждого объекта.

4. Т.к данных в целом немного – это звоночек к переобучению, любое усиление архитектуры будет приводить к тому что, модель будет просто запоминать тренировочную выборку. Логично реализовать early-stopping.

5. Как было описано выше, мне кажется изменение архитектуры на таком объеме не будет так эффективно, однако все же можно попробовать изменить архитектуру student-модели. Так можно в качестве студента найти необходимую модель по скорости инференса, но с лучшей решающей способностью

6. Как упоминалось выше, имеет смысл увеличить обучающую выборку аугментацией, можно использовать некоторые подходы из классификации текстов, например: EDA(https://arxiv.org/pdf/1901.11196v2.pdf) или DoubleMix(https://arxiv.org/pdf/2209.05297v1.pdf)

Первый подход EDA использует 4 простые техники:

- Замена синонимами (SR): Случайным образом выбираются n слов из предложения, которые не являются стоп-словами. Каждое из этих

слов заменяется одним из его синонимов, выбранным случайным образом.

- Случайная вставка (RI): Находится случайный синоним случайного слова в предложении, которое не является стоп-словом. Этот синоним вставляется в случайную позицию в предложении. Это делается n раз.
- Случайный обмен (RS): Случайным образом выбираются два слова в предложении и меняются их местами. Это делается n раз.
- Случайное удаление (RD): Случайным образом удаляются каждое слово в предложении с вероятностью p.

Второй подход: Сначала делает простые преобразования генерируя для каждого объекта сэмпл, а потом производит интерполяцию в скрытом состоянии нн.

5. Один из возможных ответов на вопрос в 3 пункте: давайте делать knowledge distillation не через soft labels, а скрытые состояния графовой сети.

Почему это хорошая идея?

Кажется, что это резонное улучшение использовать внутреннее представление GNN, а не её же редуцированную информацию на выходе GNN.

Какие «блоки» потребуется добавить / что изменить?

Необходимо изменить скрипт save_teacher_soft_labels.py, чтобы сохранялись не выходы модели, а например выход предпоследнего слоя, конкретно изменения будут в функции infer_model()

Какой лосс использовать?

Так как выход ненормирован и никак не ограничен предлагаю использовать какой-нибудь из регрессионных лоссов MSE или MAE

Реализуйте

В функции infer_model() добавлен закоментированная строчка кода, она позволит сохранять промежуточные состояния весов

```
# model = torch.nn.Sequential(*(list(model.children())[:-2]))
```