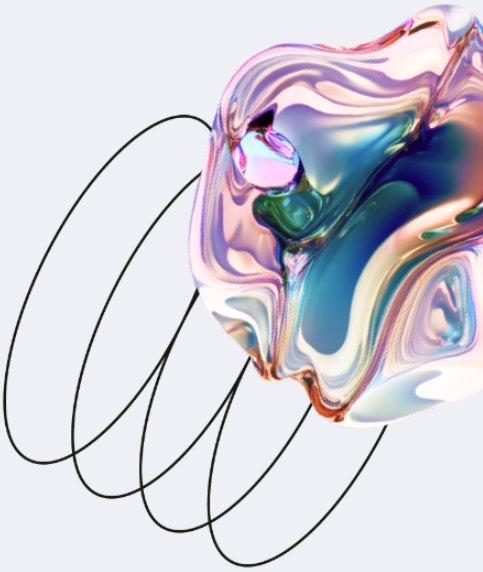
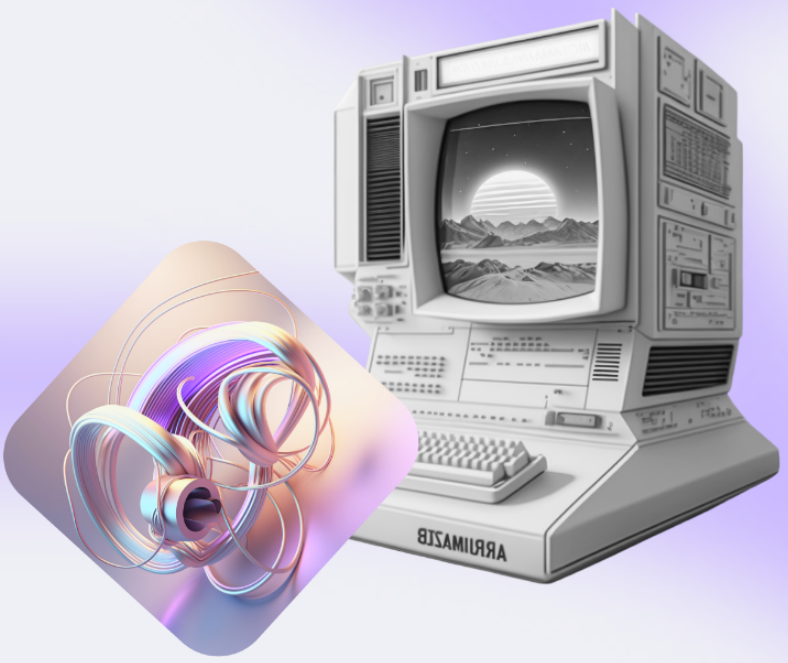
GeekBrains



Современный перенос обучения

Transfer Learning



Оглавление

Введение

Добрый день, дорогие слушатели!

Я рад, что вы пришли, и надеюсь, что вы узнаете много полезного из этого небольшого учебного курса, в который входят три лекции и столько же семинаров. Мы будем учиться применять современные большие нейросети в продуктовых приложениях.

Меня зовут Николай Капырин, я инженер и преподаватель в инженерном центре Экспонента. В 2009 я окончил магистратуру французского вуза ENSIETA, в 2010 – получил диплом специалиста Московского Авиационного Института, отучился в аспирантуре, работал в нескольких компаниях инженерного профиля, продолжая преподавать. Мне повезло с тем, что я не раз занимался курсами по машинному обучению, либо вёл их, либо готовил материалы для онлайн-курсов. Сейчас я снова работаю в преимущественно инженерной команде, и могу вам сказать, что машинное обучение востребовано везде. В этой сфере есть заказчики, специализированные инженеры, и даже инженеры других профилей за год делают несколько проектов с применением нейросетей и прочих ML-алгоритмов. Сокращение ML означает Machine Learning, я попробую его использовать поменьше, но не могу совсем исключить, особенно в семинарах которые у нас будут, где общение более спонтанно.

Скажу пару слов о предварительных знаниях. Тему переноса обучения можно осваивать как до, так и после изучения нейросетей. Для нас эти конкретные предобученные нейросети будут являться лишь одним компоненту продукта. Возможно мы немного залезем в терминологию и детали работы нейросетей. Но у вас останется много сил на внедрение их в продукт. Если же вам уже хочется сформировать глубокую интуицию о работе нейросетей, например, представлять, какое решение заведомо имеет много трудностей и низкую вероятность срабатывания, я рекомендую обратиться к соответствующему курсу, где изучаются глубокие нейросети. Для нашего с вами курса достаточно будет общей математической интуиции, которую можно воспитать в ходе изучения любой другой

дисциплины – информатики и программирования, моделирования систем, информационной безопасности, или просто математики и линейной алгебры. Не помешает курс по классическому машинному обучению, куда входят разные алгоритмы, предшествующие нейросетям.

У нас в курсе будет три лекции и три семинара.

На первой лекции мы поговорим о методе переноса машинного обучения, такой возможности выполнить вычисления заранее и поместить их в контейнер, чтобы использовать потом. Если необходимо, этот контейнер можно будет немного дообучить – как это делать, Вы научитесь на первом занятии в ходе лекции и семинара.

Вторая лекция будет посвящена использованию внутреннего состояния предобученных нейросетей. Эти замечательные алгоритмы можно использовать частично чтобы преобразовывать наши цифровые объекты, делая их более пригодными к автоматизированному анализу. На семинаре мы изучим, как использовать предобученную лингвистическую модель.

На третьей лекции мы пройдём много разных применений переноса обучения, а в финале реализуем задачу переноса стиля, воспользовавшись предубученной нейросетью.

Вы вынесете для себя знание о том, какие популярные придобученные нейросети используются в промышленности и как на их основе построить собственный продукт. Вы сможете идентифицировать нужную вам нейросеть, скачать её и запустить в вашем проекте, а используя навыки программирования вы сможете из выводов этой нейросети сделать ваши собственные, нужные вам выводы, и автоматизировать самые трудные когнитивные задачи.

Словарь терминов

Transfer Learning - это подход в машинном обучении, который предполагает использование знаний, полученных при решении одной задачи, для решения другой, похожей задачи

Файнтюнинг (fine-tuning) - это метод обучения глубоких нейронных сетей (deep neural networks), который заключается в адаптации предварительно обученной модели для решения конкретной задачи. .

Основы и метод файнтюнинга

Это первая лекция в курсе, и она будет посвящена методу переноса обучения, который позволяет извлечь максимум пользы из моделей, которые вы обучаете на основе данных. В некотором смысле, этот метод продуктивизации обученных моделей. Возможно, вы уже проходили машинное обучение и наверняка слышали пару вещей про нейросети. В этой части курса не обязательно помнить математику и знать устройство алгоритмов, достаточно общих идей. Например того, что модели создаются на основе данных, но обычно не представляют из себя каталогов по этим данным, а извлекают из них правила принятия решений по практическим задачам. вА по нейросетям – достаточно будет помнить, что они обычно организованы в несколько слоёв. Очень пригодится способность воплощать решение задач в программах, то есть ваши навыки программирования, и склонность к работе с данными.

Перенос обучения, как и машинное обучение в принципе, преследует задачу воплотить человеческий интеллект в программе. Мы пока не так хорошо оперируем другими формами мышления, зато нам хорошо понятен такой ход обучения: возможность посмотреть на несколько объектов, держа в уме конкретную задачу, потом извлечь из ряда объектов какое-то абстрактное правило и проверить его на других объектах. И люди ничему не учатся совсем с нуля. Перед тем, как научиться

говорить на иностранном языке, мы обычно осваиваем грамматику родного языка, а перед тем, как научиться классифицировать породы собак или собак, мы учимся прежде всего различать эти виды животных, проводя некоторую мысленную линию. Уже пройденный опыт складывается в две вещи:

- Во-первых, готовые правила принятия решений, благодаря которым мы можем мыслить дедуктивно, то есть принимать решения
- Во-вторых, наблюдательность, мы учимся обращать внимание на вещи, на которые раньше бы не посмотрели, использовать более сложные признаки объектов.

В общем, при обсуждении переноса обучения мы довольно близки к человеческому мышлению. Даже если модель машинного обучения будет для нас полностью чёрным ящиком, то есть мы не будем знать, что внутри, всё равно при таком интуитивном взгляде на интеллектуальные алгоритмы, вы будете лучше представлять, что в них поменять и как их использовать, чтобы с их помощью аппроксимировать человеческое мышление.

В этом модуле вы научитесь использовать предобученные, уже готовые нейросети, для преобразования информации и принятия решений. Задачи будут требовать от вас довольно стандартных навыков программирования и подготовки рабочего пространства – загрузки файлов из интернета, открытия файлов, вывода графиков.

По плану занятий, на первой лекции мы обсудим метод переноса обучения, которым сегодня решается очень много прикладных задач. Его историю и развитие. Посмотрим, что это за задачи, которые он так хорошо решает, а также – какие манипуляции с нейросетями нужно проделать чтобы адаптировать обученную сеть на собственную задачу. Потом будет семинар для закрепления навыков, мы будем скачивать нейросети и добиваться от них решения наших задач. На втором занятии мы поговорим о вышеупомянутой "наблюдательности", то есть о способности нейросетей преобразовывать саму задачу в более удобную форму, обращать внимание на более глубокие признаки. Это – Feature Engineering. И в третьей лекции мы будем много говорить обо всех применениях этого метода, развёртке

готовых нейросетей, применении нейросетей как продуктов. В конце решим сложную задачу по использованию нейросети по частям.

Мы хотим, чтобы вы как слушатели научились конкретным вещам:

- Использованию современного глубокого обучения в ваших проектах
- Научить вас собирать короткий код с применением готовых нейросетей
- Показать вам процесс создания таких алгоритмов в надежде сделать из вас квалифицированных пользователей и заказчиков

Мы постараемся указывать исторические источники и документы, где можно больше прочитать о каждой теме лекций, пользуйтесь переводчиком, или читайте в оригинале, как вам удобнее. Тема переноса обучения очень живо развивается, и в каждой популярной статье на эту тему есть какие-то намёки, которые не раз направляли подготовку этого курса в новом, интересном направлении.

Итак, давайте приступим!

Что такое TL

Искусственный интеллект пытаются создать довольно давно, и если в 1950-х проблемы казались небольшими и разрешимыми, спустя несколько циклов упадка финансирования таких разработок (их называют «зимы ИИ») пора было научиться относится более скептически к созданию "думающих машин". Понятие искусственного интеллекта развивалось, от имитаторов, которые можно было программировать вручную, через так называемые символьные системы, до современного «data science», включающего методы классического и глубокого машинного обучения.

Но финальная цель, к которой мы стремимся, состоит в том, что можно обучить алгоритм, который будет полезен во множестве когнитивных задачах, например будет подобен человеку, и при этом, поскольку это технический продукт, мы сможем компоновать из него что-то новое, эксплуатировать промежуточные результаты, так сказать – залезать в интуицию машины. В общем, не создавать новый «интеллект» под каждую небольшую задачу.

Правда в том, что пока что часто складывается так, что когда аналитик берётся за новую задачу, данных для её решения ещё пока очень мало, или они есть, но их нужно преобразовывать в форму, удобную для алгоритмов – заниматься их статистическими параметрами, удалять данные которые мы не хотим учитывать, пытаться найти и исключить предрассудки и побочные зависимости, или паразитные зависимости (ещё можно сказать). В общем, для новых задач у нас обычно мало данных.

Перенос обучения призван решать эту проблему.

Основная идея переноса обучения состоит в том, чтобы взять одну, уже обученную нейросеть, которая обучалась для некоторого домена, на некотором одном датасете

– это исходная сеть или backbone, upstream, есть всякие другие названия. И использовать её в другой, целевой задаче, "таргетной" или downstream.

При переходе на новый набор данных, этой предобученной модели может понадобиться адаптация. Если исходная модель, например, тренировалась предсказывать возраст по паспортным фотографиям лиц людей, а вы планируете её использовать для идентификации сотрудников, скажем чтобы система знала, кто резервирует какой-нибудь переговорный кабинет – то ваши портреты не будут так же хорошо освещены и, может быть, даже не будут фронтальными, в отличие то паспортных фотографий. Нужно немного переобучить систему, чтобы она учитывала новое окружение, это называется доменной адаптацией. Сегодня мы поговорим о том, как это сделать.

Как это применяется в компьютерном зрении, нам бы хотелось обучать алгоритмы, которые преобразуют изображения в какие-то абстрактные объекты, на базе которых мы можем делать аналитику. Например, задача определения того, что же именно изображено в кадре (классификация), сегодня по большей части решена, а вот задача детекции (где именно интересующий нас объект находится в кадре?, и находится ли вообще?) или распознавания текста - всё это требует комбинации из нескольких предобученных нейросетей или других алгоритмов. А что если мы хотели бы обучить модель общим принципам распознавания изображения на огромном датасете, например просто на разных фотографиях из интернета, а потом научить её же определять возраст человека на фотографии, или то, насколько нашей аудитории понравится та или иная фотография? Последняя задача очень субъективна, но на ней отлично демонстрируется перенос обучения между схожими задачами.

В обработке текстов нам бы хотелось, чтобы нейросеть выучила взаимосвязи концепции языка, независимо от того, какой это язык. В нашем десятилетии, линвистические модели в чат-ботах уже не удивляют, но уже в 2016-2018 годах были вполне удобные нейронные модели, правда гораздо проще, чем текущие варианты GPT. Например, нейросеть BERT. Её можно было использовать для грамматического анализа, для перевода с языка на язык, и для той же классификации текстов, и сейчас она не потеряла интереса в небольших задачах анализа, хотя прогресс GPT показывает, что для создания чат-бота нужно было значительно доработать принятый подход.

Итак, изображения и текст. Эти две области не резюмируют машинное обучение, есть ещё очень много задач продуктовой аналитики, работы с таблицами, графами, рекомендательные системы – здесь очень помогают те самые абстрактные промежуточные представления, которые нейросети могут сделать из картинок или текстов. Так рождаются чат-боты или классификаторы компаний на честные или мошеннические, на основе анализа графа взаимосвязей и текста их сообщений.

Пример задачи по TL

Для приложений, в которых у вас не так много данных, transfer learning – это замечательная техника, позволяющая использовать данные из задачи в том, чтобы помочь вам решить вашу задачу. Это одна из тех техник, очень часто используются в продуктовом машинном обучении. Давайте посмотрим, как работает перенос обучения.

Допустим, вы хотите распознать рукописные цифры от нуля до девяти, но у вас нет большого количества размеченных данных об этих рукописных цифрах. Для обучения эффективной модели такого количества данных может быть совсем недостаточно. Что мы можем предпринять в таком случае? Допустим, вы нашли очень большой набор данных из нескольких миллионов изображений с фотографиями собак машин кошек людей, и так далее – всего 1.000 классов. Что сделает аналитик данных в таком сценарии? Он может сперва выполнить обучение нейросети на том наборе данных, который у него есть, который включает 1.000 различных классов. Он обучит алгоритм принимать на вход изображение размера X на Y и возвращать, в ответ, в качестве результата распознавания один из тысячи различных классов. Точнее говоря, возвращать распределение вероятности наблюдаемости каждого из 1000 классов.

Дальше, наверное вы уже слышали, что самые интересные нейросети – это глубокие нейросети – те, в которых информация проходит много слоёв. Итак, вы обучаете первый слой второй, и так далее, до слоя, предшествующего выходному. В этом процессе вы находите параметры для каждого слоя нейронной сети: первого, второго, и так далее, скажем, до 10.

Когда сеть обучена, мы используем её в процессе transfer learning. Применение переноса обучения заключается в том, чтобы взять копию этой нейронной сети, в которой сохраняются все слои вплоть до девятого. Последний слой мы исключим, отрежем, и заменим его на гораздо меньший выходной слой с десятью элементами вместо 1000. То есть будем искать уже другое вероятностное распределение и по его максимуму судить, какой класс, какое число преобладает.

Почему это будет работать?

В нейросетях, которые обычно приводят в пример по этим задачам, данные проходят от входов к выходам, в одну сторону. Каждый следующий слой служит для очередного этапа преобразования данных. То есть у каждого слоя собственная отдельная функция, их работа зависит от предыдущих слоёв, но в обратную сторону это не работает – предыдущие слои не зависят от последних. Значит, если последние слои убрать, оставшаяся часть сети всё равно будет решать какую-то свою задачу.

После такой модификации, то есть убрав один или несколько слоёв нейросети, мы можем создать на выходе нейросети другую, нужную нам структуру. Несколько полносвязанных слоёв и софтмакс после них во многом аналогичны классификатору "метод ближайших соседей". Если мы уберём этот классификатор на выходе, и добавим на его место другой, например с другим количеством выходных слоёв, или с ещё б'ольшим количеством слоёв, или – почему бы и нет – подмешивающий в данные какой-нибудь выход какой-нибудь другой нейросети, мы можем получить на выходе что-то интересное.

Но возникнет проблема: мы взяли большую нейросеть, которую кто-то обучал возможно – недели, возможно – месяцы, поменяли в ней что-то, и... теперь нам тоже нужно ждать несколько недель пока она обучится? А что, если наступит проблема переобучения? Ведь, по условию, у нас нет огромного и чистого датасета под нашу целевую задачу, значит у нейросети будут сотни, может даже десятки примеров для обучения. Естественно, она переобучится – запомнит все эти

примеры наизусть, а когда встретит новые – будет принимать случайные решения. К тому же, она забудет предыдущую задачу, которой её обучали. Это не очень хорошо.

Чтобы такого не было, слои старой сети "замораживают", просят их не обращать внимания на процедуру обучения и заодно экономят на вычислениях. Сперва можно обучать только последнюю добавленную конструкцию из пары слоёв, затем можно "разморозить" сеть и на очень маленькой скорости дать ей немного дообучиться. Это один из способов увеличения качества модели, но если он не сработает – ничего страшного.

Машинное обучение vs Перенос обучения

Поговорим о том, чем перенос обучения отличается от машинного обучения.

Перенос обучения нам нужен для того, чтобы создавать модели приемлемой точности, имея не так уж много данных в датасете. В классическом машинном обучении такая задача обычно считалась баезнадёжной. Но в случае с переносом обучения это возможно.

Простыми словами, перенос обучения – это всё ещё метод машинного обучения. При работе с предобученными моделями, как я уже говорил, можно разделить модель на две части: предобученный преобразователь и дополнительный алгоритм. Иногда он называется "головой модели", а предобученная часть – хвостом, что – соглашусь – немного странно, но можно считать что именно голова принимает решения и выдаёт результаты интеллектуальной задачи. Отправная модели позволяет очень быстро найти следующую, оптимальную для вашей задачи модель, на совсем небольшом количестве имеющихся у вас данных.

Бывает ли перенос обучения не на нейросетях, а на моделях классического машинного обучения? Мне такое не встречалось. Модели классического машинного обучения – линейная и логистическая регрессия и классификация, опорные векторы, ближайшие соседи байесовские модели или деревья – обычно не так уж

долго обучаются, их предобучение занимает десятки минут, может быть часы, но я никогда не слышал о практически применяющихся моделях, которые бы обучались недели или месяцы. То есть аспект "заморозки вычислений" здесь не очень применим, всегда выгодно обучить модель заново.

Что касается дообучения классических алгоритмов – давайте изучим очень простой пример. Часто у таких алгоритмов даже нет возможности дообучиться, а если она есть, то мы используем

Предположим, у нас есть один датасет, который расположен слева, он относится к исходной задаче. И ещё один датасет, размером поменьше, который находится справа – это целевая задача. На самом деле, вы можете придумывать несколько геометрических способов, как перенести обучение в такой задаче, но рассмотрим этот пример умозрительно. Допустим, мы обучили классификатор на левом датасете, и если ничто не намекает на то, что разделительная линия будет поворачивать и продолжится так чтобы разделить и правый датасет, то наш разделительная линия для этого классификатора будет выглядеть примерно так (рисуем линию между закрашенными красными и зелёными точками). Если мы возьмём тот же самый классификатор и обучим только на точках второго датасета (проводим разделяющую линию между незакрашенными точками), то вы видите, что это линия будет довольно плохо разделять первые датасета.

Я проводил численные эксперименты для линейной регрессии. На самом деле, у большинства классических классификаторов нету возможности имитировать перенос обучения, Но если выбрать решатель для линейного классификатора, который работает пошагово. И так, если обучиться только на первом датасете, где группы немного перекрываются, то мы получаем 94% на первом датасете и 80% на втором, что довольно неплохо, надо сказать. Но, к сожалению, бизнесу этого будет недостаточно. Ведь в продуктовом окружении обычно мало данных и очень высокие требования по качеству. Если мы обучим классификатора только на втором датасете, то качество на первом дата свете очень сильно падает. Вы видите, что разделяющая линия практически не работает для элементов первого датасета. Если обучить классификатор на обоих дата сетах сразу, то мы получаем 95% на первом и 90% на втором. Но если взять первый датасет и за один шаг чуть-чуть Да обучить его для восприятия второго дата сета, используя ньютоновский метод оптимизации,

то мы получаем чуть-чуть другой результат, 65% на первом дата сессии и 90 на втором.

Это очень простой пример. Но среди всех интерпретаций того, что вы сейчас видели, важна следующая: если полностью заново повторить обучение для нового датасета, классификатор забудет всё, что он знал про исходную выборку, включая взаимосвязи между элементами, которые вполне могли бы пойти на пользу новому классификатору, если мы будем пытаться экстраполировать его работу за предел старого и нового датасета. Ведь вполне вероятно, что в реальной жизни нам попадутся образцы, находящиеся между этими выборками.

Transfer Learning глазами исследователя данных

Когда мы скачиваем нейросеть и переобучаем её под свою задачу, или просто используем её без дообучения как звено какой-нибудь цепочки, обычно нет времени подумать, какие исследования легли в основу этой сети, и какие ещё ведутся. Но понимая примерную динамику этого процесса, вы сможете лучше предвидеть, что будет дальше, и сможете делать с нейросетями немного более сложные вещи, основываясь на том, что с ними в принципе можно делать.

Возьмём например первую глубокую нейросеть, можно допустить такое обобщение, которая называлась AlexNet. 2012 год – это время когда инструментарий .NET ещё был очень популярным, но в названии нейросети нет дани этой технологии. Хотя...

Итак, посмотрим на архитектуру этой первой глубокой нейросети, которая победила в конкурсе на распознавание датасета ImageNet в 2012 году. Как победила? Выбила ошибку попадания в top-5 категорий примерно в 15%. top-5 – означает, что правильный ответ находился среди пяти ответов, которые нейросеть посчитала наиболее вероятными. В этой сети есть несколько разных слоёв: свёрточные, пулинг, полносвязанные. Примерно во время той публикации появились массивные параллельные архитектуры на базе видеокарт. Все программы для всех этих слоёв нужно было переписать для видеокарт, примерно в качестве шейдеров. И разработчики сети этим и занимались. Кроме того, они испытывали десятки и десятки моделей. В 1989 году Ян ЛеКун опубликовал свои экспериментальные

результаты по нейросети LeNet, так вот – в 2012, в AlexNet использовались другие функции активации внутри сети, ReLU на замену сигмоидам, и вместо одного свёрточного слоя сразу шло три слоя подряд. Это была ещё одна уловка, которая позволяла обучать сеть более сложным правилам и даже экономила память. Что ещё можно сказать про эту сеть.

Можно перечислить другие улучшения, которые вошли в эту модель, но – что я хотел бы сказать – развитием архитектур занимаются учёные, или дата сайентисты, в узком смысле. Мы не знаем, какие улучшения и по какой причине сработают. Следующая нейросеть для компьютерного зрения, которая сейчас является золотым стандартом – ResNet или Residual Network, была создана в Microsoft Research в 2015, и изначально с её помощью пытались обеспечить возможность обучать сеть частично, только до некоторого внутреннего слоя, постепенно наращивая сеть при помощи новых "необученных" слоёв. А получилось решение проблемы затухания градиента, из-за которого долгое время никто не мог автоматически обучать модели, которые были бы глубже пяти слоёв.

Ещё одна тема из науки, но которая вам очень пригодится – перенос обучения добавляет нам в инструментарий ещё один алгоритм – feature extraction. О нем мы обширно поговорим на следующей лекции. Дело в том, что внутри нейросети происходит много всего интересного, это очень интересный способ организации вычислений, и даже промежуточные процессы, которые происходят внутри сети, тоже можно использовать в своем решении. Каждый слой нейросети преобразует задачу и передает на следующий слой вектор выходных параметров. Этот вектор, который кодирует параметры входного объекта в некотором промежуточном, латентном пространстве, мы будем называть эмбеддингом, или латентным представлением.

TL глазами разработчика

Но вернёмся к нашей инженерной реальности. В этом курсе мы будем использовать перенос обучения, будучи в роли разработчиков. Или аналитиков, если хотите.

До 2012 года нейросети были такими же алгоритмами как остальные, одним среди прочих. Мало кто представлял, что они окажутся настолько полезными для вычислений и развлечений, как мы сейчас видим, когда названия архитектур звучат в глянцевых журналах.

Современную эру переноса обучения отсчитывают от нейросети AlexNet, хотя подобными архитектурами занимались с 1980-х годов. Что изменилось? Резко выросла доступность вычислений, стало много качественных данных (собранных для конкурса ImageNet) и интерес талантливых исследователей.

Нейросети, как вы знаете, состоят из слоёв. Тут ничего сложного, информация преобразуется слой за слоем. Наверное, чаще вы слышите про количество параметров в моделях (семь миллиардов, 13 миллиардов...) про количество слоёв особо не говорят, но, в общем, идея состояла в том, что выходной слой, который ближе всего к решению поставленной задачи, можно заменить на что-то ещё, оставив остальную архитектуру и веса, по большей степени, нетронутыми.

Для разработчиков есть вариант совсем не переобучать нейросеть, для такого применения нужно суметь скачать нейросеть и подготовить данные для её работы. Организовав подачу данных в нейросеть вы сможете решать задачу так, словно используйте готовую библиотеку, но только это конкретная библиотека умеет классифицировать или детектировать изображения, придумывать синонимы, различать надёжных и ненадежных контрагентов и т.д

Другое прикладное применение предобученных нейросетей – zero shot learning. Тонкое различие этого метода обычным скачиванием готовой нейросети и использованием её по назначению заключается в том, что объекты, которые мы будем подавать в придабученную нейросеть, относятся к классам которые это нейросеть раньше вообще не видела. Например, исходно нейросеть могла классифицировать животных средней полосы, присваивая им текстовые метки. Задачи zero shot learning такая нейросети получает на вход изображение зебры, и благодаря известному ей распределению классов, присваивает ей правильную метку – например, полосатая лошадь. Вы можете представить, что такие задачи работают с чуть более сложной структурой меток. Но метод появился в 2008 году в области компьютерного зрения, так что он существует довольно давно.

Few-shot learning или onr-shot learning – это самый часто встречающийся метод переноса обучения. Под shot здесь подразумевается показ нейросети объектов нового датсета и проход задачи оптимизации. Как понятно из названия, алгоритм не придётся обучать слишком долго.

В целом, в реальности выборки обычно отличаются от учебных, также отличаются и распределение и вообще встречаемость отдельных признаков. Перенос обучения делает алгоритмы машинного обучения более устойчивыми к смене домена о которой мы уже говорили.

TL для разных задач

Нейросети для обработки изображений (1)

Прежде всего, поговорим о применении нейросетей в обработке изображений. Как я уже говорил, до 2010 или 2012 года нейросети не рассматривались как основное решение задач компьютерного зрения, как ни странно.

К тому времени уже три года лет существовал конкурс под названием ILSVRC. В 2012 году в конкурс входило три задачи: классификация изображений, детекция и точная классификация детектированного изображения. В задаче классификации, по входному изображению поданный на конкурс алгоритм должен был предсказать пять возможных классов, к которым может относиться основной объект на изображении. Датасет конкурса включал объекты, относящиеся к 1.000 разных классов и в каждой группе было 1.200 изображений. Для половины из них были даны также и ограничительные рамки, или bounding box. В конкурсе используется несколько урезанный dataset, потому что в целом, дата сет image.net насчитывает больше 14 млн изображений, размеченных на более чем 21.000 категорий – представьте себе, и создание такого дата сета было огромной работой. Например, его постоянно перепроверяли через службу разметки amazon, которая называется mechanical turk, где люди получали случайные изображения и давали им описание. Но, возможно, эту работу теперь будут делать нейросети.

И вот, в 2012 году AlexNet показывает не просто хорошее решение а побеждает все алгоритмы, которые были основаны не на нейронных сетях. Тогда крупный бизнес активно взялся за это соревнование, топовые it-компании представили свои архитектуры, и с 2015 года, нейросеть от исследователей из компании microsoft создали resnet, который сейчас является золотым стандартом индустрии не только для классификации изображений, ну и для многих задач обработки, которые выстраиваются на эмбеддингах resnet.

С 2018 года в dataset imagine net входят трёхмерные изображения. Ещё интересно, что эта датасет с самого начала сопровождался текстовым датасетом WordNet, на основе которого строилась иерархия понятие в датасете. Это датасет продолжает быть очень релевантным сегодня.

А сам конкурс ILSVRC в последний раз проводился в 2017 году. Если вспомнить, что он был начат в 2009, то приятно, правда, что на решение такой важной задачи ушло меньше 10 лет. Этот конкурс породил множество современных моделей анализа изображений и, конечно, играл важную роль в приближении революции глубокого машинного обучения.

Я бы хотел сказать ещё пару слов про нейросеть AlexNet. Тут вы видите её диаграмму в сравнении с нейросетью 1998 года, которая называется LeNet. Более старую сеть создал очень известный сейчас учёный в области глубокого машинного обучения, канадец Ян Лекун. А сеть 2012 года создал Алекс Крыжевский в рамках своей диссертации в университете Торонто.

Опять же, скорее всего вы знаете, что нейросети обрабатывают информацию слой за слоем. В диаграмму слева поступали картинки размером 28x28, Вы видите что в сети два свёрточных слоя, функционирование которых вам может быть известно из области обработки сигналов или изображений, если нет - ничего страшного, Нам пока не нужно глубоко понимать, как они работают.

Свёртка – это операция, которая производится между изображением побольше и изображением поменьше, которое называется ядром свёртки. Часто производится свёртка картинки сразу с множеством ядер. Ядро содержит, можно сказать, шаблон очень простого объекта, который мы ищем на изображении. В той области, где ядро хорошо совпадает с исходным изображением, результат свёртки будет относительно большим. В остальных местах, результат будет близок к нулю. Однако мы заранее не знаем значение ядер, то есть форму тех самых простых объектов, которые мы ищем на исходном изображении. Роль нейросети – найти и запомнить эти ядра. Второй сверточный слой уже работает с результатами предыдущей свёртки.

Как вы видите, во второй сети уже пять свёрток, хотя автор экспериментировал с разным количеством слоёв и ядер. Говорят, что три свёрточных слоя, расположенные подряд, поставленные здесь без промежуточных слоёв пулинга, потому что эти слои просто не влезали в память. Так или иначе, такое расположение свёрточных слоёв сейчас используют очень часто. В конце обеих нейросетей вы видите несколько полносвязанных слоёв, то есть таких слоёв, где информация передаётся с каждого входа на каждый выход, давая возможность всем признакам перемешаться и составить выходное решение – распределение объектов либо на 10 классов, либо на 1.000.

К разговору об AlexNet и других подобных моделях интересно добавить следующее. Обычно, для переноса обучения, нам понадобятся только свёрточные слои, и пара

технических слоёв, следующих за ними. А поскольку успех свёртки не зависит от размера исходного изображения – ядра свёртки проходят по всем пикселям исходного изображения и создают новое изображение, пропорциональное степени локального совпадения с ядром... В итоге, свёрточная часть модели может принимать на вход картинки любого размера. То есть у вас будет нижний предел, поскольку слой за слоем изображение сжимается, нельзя чтобы оно свернулось в один пиксель. Но верхнего предела не будет, за его возникновение отвечают полносвязанные слои, но и перед ними можно поставить какие-нибудь блоки интерполяции. Я думаю, это важно знать при работе со свёрточными нейросетями, хотя, конечно, на то, чтобы добиться гармоничного соотношения размеров всех слоёв в нейросети уходит львиная доля сил по работы над архитектурой. Но не стоит слишком переживать о ней. Помните, что свёрточная часть может обработать изображение любого размера.

Я уверен, что более глубокие детали реализации этой и других архитектур вы ещё пройдёте в курсе по глубокому обучению. Мир чётко разделился на компьютерное зрение до 2012 года, и после. Раньше алгоритмы машинного обучения опирались на так называемые рукотворные признаки, то есть на такое преобразование входного изображения, которое информатики выдумывали сами. Теперь выходная конструкция предобученной нейросети опирается на преобразования, которые придумала, или выучила нейросеть, исходя из поставленной задачи.

Давайте ещё посмотрим на историю развития больших предобученных сетей. Видно, что качество классификации росло не совсем ровно в соотношении с количеством параметров в нейросети, или времени, затраченного на её обучение. Ещё видно, что размер нейросети не гарантирует ей высокое качество классификации.

Про признаки, которые можно получить из университет, мы поговорим на второй лекции. А остальные задачи, кроме классификации, мы поговорим на последней, третьей лекции этого маленького курса.

А пока посмотрим, что там в области анализа текста.

Нейросети для обработки текста

Анализ текста тоже прошёл длинный путь, от рукотворных признаков типа TF idf, через эру контекстно-независимых алгоритмов, в большей степени нацеленных на работу с небольшими группами слов или с отдельными словами. Сейчас мы находимся в эре-трансформеров.

Давайте немного поговорим про эту архитектуру. Трансформеры принято сравнивать с рекурентными нейросетями, который я способны обрабатывать последовательности любого размера, принимая не всю последовательность целиком, а каждое слово, одно за другим. Но это не совсем правильно. Хотя Трансформеры и использовались изначально для работы с текстом, сегодня на их основе строятся и диффузионные модели, один из современных генераторов изображений. Не имеют рекурентной структуры, они просто умеют обрабатывать длинные последовательности за один проход. Можно обобщённо сказать, что трансформер, внутри себя, выучивает граф взаимосвязей разных слов в своём словаре.

На одном из следующих семинаров мы задействуем языковую модель BERT. Это нейросеть, по своей организации, умеет переводить один набор слов в другой. Это так называемые seq2seq модели. Количество задач, которые может решить это модель, включает машинный перевод, разговорные применения, суммаризацию текста, и так далее. И, хотя на многих прикладных задачах её победили более мощные языковые модели из семейства gpt, она остаётся релевантной в качестве компонента алгоритма для классификации текстов например.

И на этой диаграмме Вы видите, что зоопарк языковых моделей продолжает пополняться. Среди наиболее современных моделей есть и модели размером менее чем 100 млн параметров, и чат gpt с его сотней миллиардов параметров. Модели такого размера уже практически невозможно скачать и запустить на пользовательском устройстве. Вы видите, где находится модель берд, и при скачивании её размер на диске составляет примерно 700 Мб, а обработка одного

предложения занимает порядка половины секунды на обычном ноутбуке с графической видеокартой. Представьте, как долго на той же платформе будет работать чат gpt.

В общем, такие огромные модели предоставляются как сервисы, к ним можно обращаться через API. Например, давайте спросим что-нибудь у языковой модели ruGPT-3 от исследователей из сбербанка. (Показываем примерно, смотрим результат, например спросим – я разрабатываю курс по машинному обучению и создаю лекцию про свёрточные нейросети; план этого лекции следующий: 1.)

В духе нашей сегодняшней лекции нужно сказать, что трансформерные модели вроде берд создавались как для всего языка сразу, или для группы языков, так и для отдельных областей. На диаграмме вы видите нейросети ориентированные на китайский язык или на все языки сразу, а также медицинские, биологические, научные, юридические и другие вариации bert. Есть модели, прошедшие компрессию, то есть либо избавленные от неиспользованных нейронов для экономии места, либо дистиллированные, либо ещё как-нибудь оптимизированные. Это карта методов должна составить у вас представление о крайнем разнообразии архитектур и применений трансформеров.

Остальные типы данных

Конечно, промышленные и бизнесовые задачи далеко не всегда сводится к анализу картинок или текста. Посмотрим на перечень задач, для которых есть модели на репозитории hugingface.

Кроме компьютерного зрения и обработки естественного текста, где вы уже видите много интересных приложений, есть ещё нейросети для обработки звука – классификация звуковых отрезков, преобразование аудио последовательностей в другие аудио последовательности, распознавание речи или перевод текста в речь. Все эти модели можно использовать в своих проектах, полностью, или в рамках переноса обучения.

Чаще всего бизнесу нужно работать с табличными данными. За идеями и статьями можно зайти в соответствующий раздел репозитория huggingface, или же просто поискать статьи, где решается похожая на вашу задача.

Ещё вы видите здесь мультимодальные нейросети и прочие модели. Раньше я уже говорил, что эмбединги разных модальностей, например текста и изображений, можно объединять или производить над ними математические операции, например брать среднее арифметическое значение, и это один из базовых приёмов для создания таких нейросетей, которые, например, учатся писать подписи для изображений. Или, шире – преобразовывать изображение в текст, а текст в изображение. Видите, мы снова вернулись к современным генераторам картинок.

И довольно свежее применение глубокого машинного обучения сосредоточено на графах. Например, глубокие нейросети предсказывают совпадение интересов людей в социальных сетях, и на основе такой модели можно создать рекомендательную рекламную систему.

Для эмбеддингов есть специальные базы данных. По сути, эмбединги – векторы. есть, поиск в векторной базе данных вполне может быть похож на поиск вектора, наиболее похожего на искомый, входной вектор. Это, в свою очередь, напоминает нам о методе ближайших соседей, но стоит помнить, что эмбеддинги модели берт составляют несколько сотен параметров для маленьких вариантов модели.

Описание процесса файнтюнинга

Как выполняется TL

Под конец этой лекции мы немного обсудим основной метод организации переноса обучения – это файнтюнинг.

Обучение серьёзных моделей требует большого объёма данных, и их у нас совсем не всегда будет достаточно. Но, как я уже сказал, можно предобучить наш алгоритм каким-то общим концепциям, а потом, если задача более-менее схожая с базовой задачей, дообучить алгоритм, либо просто использовать его не целиком, а по кусочкам.

Одним из самых популярных, простых и доступных методов переноса обучения является файнтюнинг.

1. Загружаем предобученную сеть

Мы уже пару раз подходили к описанию этого метода в ходе сегодняшней лекции. Первым делом для fine тюнинга нужно предобученную модель и изучить её архитектуру. Для наших практических задач, нужно, чтобы кто-нибудь уже попробовал использовать эту модель в качестве backbone архитектуры. Где можно найти такие модели?

Среди библиографических ссылок в этом курсе вы можете увидеть сайт huggingface, а также коллекция статей paperswithcode, и если вы знаете, как называется ваша задача по-английски, или просто ищете сеть для какого-то конкретного вида данных, там вы получите доступ к обширному набору предобученных моделей.

Репозитории русскоязычных лингвистических моделей, например, есть у активных исследовательских коллективов, например у сбера, физтеха, и называется он deep pavlov.

Предобученные модели могут быть встроены в ваш framework, например в pytorch или tensorflow, они могут даже быть упакованы в ваш инженерный пакет, например в matlab есть deep learning toolbox, куда входят многие популярные предобученные

модели, которые можно файнтюнить, и даже генерировать из них код на низкоуровневом языке.

Напоследок, поискать модели, детали реализации и алгоритмы до обучения стоит через поисковые системы с префиксами "jupyter notebook" или "google colab". Эти слова всегда подсказывают поисковику, что Вы ищете что-то наглядное, как учебные материалы и одновременно готовые к использованию в прикладных задачах.

2. Заменяем последние слои

Следующий этап - работа с выходными слоями вашей модели.

Важно узнать, сколько слоёв можно и нужно переобучить для того чтобы увидеть прогресс по вашей конкретной задаче. Есть ли такой информации не находятся, нужно начать отсоединять от сети выходные слои, один за другим.

Довольно сложно определить, какие слои нужно файнтюнить, а какие нет. После нашего разговора о свёрточных архитектурах вы уже имеете представление о том, какие слои обычно нужно переделать, а какие нужно оставить, как есть. И я уже говорил, что для того, чтобы исключить забывание нейросетью взаимосвязей, выученных на исходном датасете, нужно заморозить всю входную часть модели, хотя бы на время первичного обучения.

В курсе по глубокому машинному обучению вы пройдёте в деталях процедуру оптимизации нейросети. Пока имейте в виду, что новые, подсоединённые к нейросети слои, пока не имеют устоявшихся параметров, обычно их инициализируют случайными числами. Эти случайные числа будут, первое время, предсказывать абсолютно случайные выходы сети, демонстрируя очень большую ошибку на целевом датасете. Что будет, если не заморозить как можно большую часть исходной сети? Ошибка из-за случайной инициализации новый выходной структуры будет передаваться по всей нейросети и существенно менять веса всех слоёв, вплоть до самых ранних.

Ещё Вы помните, что каждый следующий слой свёрточной нейросети работает с результатами предыдущего слоя. То есть, признаки, найденные на ранних слоях, сочетаются в более сложные признаки на последующих слоях. Слои, которые ближе к входу нейросети, различают более простые признаки: градиенты яркости, точки и линии... Дальше идут слои, которые реагируют на текстуры, потом – слои, которые научились активнее реагировать на более высокоуровневые элементы дата сета, например, глаза, фары, колёса, светофоры – те детали изображения, по наличию которых уже можно принимать некоторые решение по целевой задаче. Так что незачем переобучать входные слои, если всё, что они делают, это обнаружение наиболее простых признаков, которые есть и в исходном, и в целевом датасете. А когда веса выходных слоёв немного устояться, можно начать размораживать и последние свёрточные слои, установив небольшую скорость обучения. Признаки, которые не встречаются у объектов целевого датасета, могут снизить своё влияние, и тогда при сжатии нейросети, они могут быть отброшены.

Но размораживать больше слоёв лучше тогда, когда у ваш датасет не очень маленький и насколько исходная задача похожа на финальную. В случае маленького датасета, нейросеть легко переобучится, лучше разморозить меньше параметров. А ещё, если задачи похожи, то исходная сеть содержит много информации, которая будет полезна для финальной задачи, можно заморозить ещё несколько слоёв.

3. Обучаем сеть заново и оцениваем качество

Обучение нейросети при fine тюнинге ничем не отличается от обучения в обычном режиме. Вы подаёте в нейросеть объекты целевого датасета и смотрите за качеством на обучающей и на тестовой или валидационный выборке.

Оценка качества при переносе обучения тоже ничем существенно не отличается. В надежде на то, что выученные ранее взаимосвязи между признаками и метками объектов, пригодятся при работе с объектами целевого датасета, Вы можете не проверять качество модели на исходном датасете, только на целевом. Если выученные знания не скажутся на качестве целевой модели, вы это увидите при тестировании на целевом датасете. Точно так же, как при обычном машинном

обучении, следует опасаться переобучения. Оно проявляется в форме резкого роста ошибки на валидации при сохранении хорошей динамики ошибки на учебных примерах. То есть, ошибка при обучении может продолжать снижаться, или колебаться вокруг некоторого уровня, но ошибка модели на примерах, которых она раньше не видела, начинает очень существенно расти. Имеет смысл сохранять копию модели на каждой эпохе обучения, Так что когда вы заметите, что наступает переобучение, вы сможете вернуться к одной из сохранённых копий модели, которая ещё не переобучилась.

Выводы

Когда применять перенос обучения?

Под завершение лекции давайте поговорим, когда нужно применять метод переноса обучения.

Если вы собираетесь взять одну задачу и попробуете перенести знание, собранное в модели, для выполнения другой задачи, то, конечно, нужно чтобы входные объекты обеих задач были одинаковыми. Система распознавания речи должна принимать на вход какие-то речевые признаки, хотя, как мы ещё обсудим на других лекциях, данные можно преобразовывать между разными видами описания. В случае со звуком, например, один из вариантов – взять двухмерный спектр звукового сигнала, например при помощи вейвлет-преобразования, и классифицировать его, словно это изображение. Если вы придумаете задачу на стыке анализа изображений и анализа текста, например – поиск картинок по текстовому запросу, вам понадобится мультимодальная сеть, которая училась на эмбеддингах и – текста, и – изображений. Эмбеддинги можно просто поставить один за другим, получится один большой такой большой вектор, а можно преобразовать, или, например, взять среднее арифметическое от них. Или, как раньше делали поисковые движки – искать не по картинке, а по всему тексту, который расположен на веб-страницах вокруг этой картинки.

Перенос обучения обычно используется, когда для вашей задачи накоплено мало данных, но есть схожие задачи, для которых накоплено много данных. То есть – есть большие качественные датасеты. Например, в медицинском исследовании может быть сотня рентгеновских снимков, размеченных опытными врачами, это уже большая коллекция. Можно добавить много снимков здоровых людей, но это не поднимет качество классификации между разными патологиями. Тогда нужно, в качестве backbone модели, взять ResNet50, которая обучалась на ImageNet, и дообучить пару финальных слоёв классифицировать именно вашу коллекцию снимков.

Второй, более промышленный вариант – это когда вам нужно обучать много моделей под много разных задач, и вы хотите ускорить этот процесс, обучая

крупные модели под какую-нибудь достаточно стандартную задачу, чтобы потом использовать отдельные её элементы в других моделях. Например, банки так предварительно рассчитывают признаковое описание своих клиентов, чтобы потом, в закрытом безопасном контуре, делать расчёты на этих латентных представлениях. Интересно, что нужно тщательно следить за "сроком годности" таких моделей, ведь информация о клиентах всё время меняется, клиент даже может потребовать изъятия своей информации из датасета, например, при уходе из банка. Тут помогает производить дообучение исходной модели под небольшие изменения в исходном датасете.

Когда не применять?

Когда не стоит применять перенос обучения, а стоит обучить модель с нуля? Есть пара таких сценариев, когда применение переноса обучения ничего не даст или скорее всего обернётся тратой времени.

Первый, самый неинтересный – это если никто не применяет перенос обучения в вашей конкретной задаче. Поиском оптимальных моделей занимаются устойчивые научные коллективы и исследовательские отделы компаний. Если они находят такие архитектуры, которые эффективно обучаются и не теряют свои свойства при переходе на другую задачу, то тем лучше. Не стоит тащить в продукт такую задачу, которую никто успешно не решал.

Второй случай – если задача требует отдельного большого датасета. Настолько большого, что параметры предобученной сети будут полностью переписаны. Тогда имеет смысл очистить нейросеть от найденных в ходе обучения параметров, весов и смещений, и просто начать обучение с нуля, но на той же архитектуре.

Если данные целевой задачи очень отличаются от данных исходной, это похоже на второй случай. Например, если у вас есть классификатор собак, а вы хотите сделать классификатор кошек – эти данные уже достаточно похожи на уровне признаков. Как низкоуровневых – текстура шерсти, цветовая палитра, так и высокоуровневых – расположение глаз, форма морды. И они достаточно похожи на уровне

организации датасета: это тоже изображения примерно соотносимых размеров. Скорее всего, на каком-то внутреннем уровне, нейросеть обучена примерно тем признакам, от которых можно отталкиваться при обработке обоих датасетов, как заложенного в модель учебного, так и вашего целевого. Но искать, сколько именно слоёв нужно заморозить или отрезать – очень трудное занятие с современными нейросетями, которым имеют несколько сотен слоёв.

Вы сами отлично поймёте, если ваш сценарий не требует переноса обучения. Нам важен продуктовый аспект – что можно запустить предобученную нейросеть и очень быстро добиться неплохого качества на вашей задаче. Если на это всё уходит очень много времени, конечно, это не для бизнеса.

Заключение

Давайте подведём итоги этой лекции. Алгоритмы машинного обучения позволяют довольно хорошо находить правила принятия решений в широком перечне задач. А перенос обучения позволяет использовать данные и уже проведённые вычисления, чтобы расширить этот перечень до бесконечных пределов.

На следующем занятии мы углубимся в изучение ещё одного приёма, который выполняют с предобученными нейросетями, а именно – работу с внутренним векторным представлением, которое нейросеть формирует в ходе вычисления своего прогноза. Это представление часто называется эмбеддингом.

Во множестве промышленных проектов с машинным обучением первым делом пытаются использовать либо полностью готовые нейросети, либо перенос обучения (то есть – дообучение). Предобученые нейросети позволяют быстро получить бейслайн-решение, то есть хоть какое-то решение задачи с которым можно сравнивать последующие попытки.

И в библиографии вы найдёте ещё несколько отсылок к курсам или материалам о том, где и как используется перенос обучения. Но вы и сами найдёте много дополнительной информации, термин transfer learning сейчас довольно горячо обсуждается, ведь этот вытащил нейросети из лабораторий и привёл их в широкую промышленность.

Надеюсь, вам было интересно, дальше будет небольшой опрос (?) и увидимся на семинаре, где мы будем применять нейросеть ResNet в качестве классификатора.

Библиография

https://www.v7labs.com/blog/transfer-learning-guide – мы до сих пор прошли только гомогенный перенос обучения, но в этой статье перечислено гораздо больше видов и хорошая теоретическая база

https://habr.com/ru/companies/wunderfund/articles/590651/