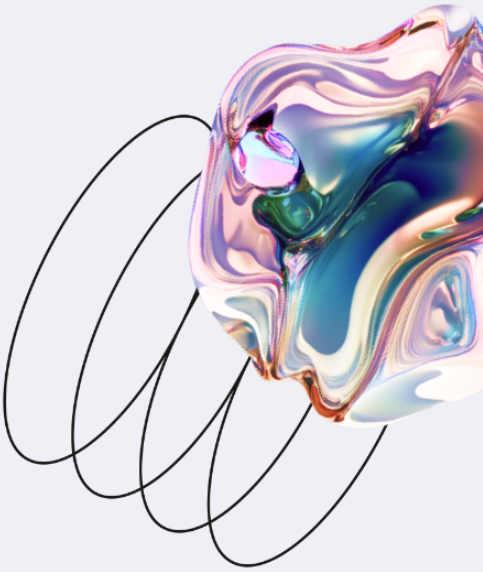
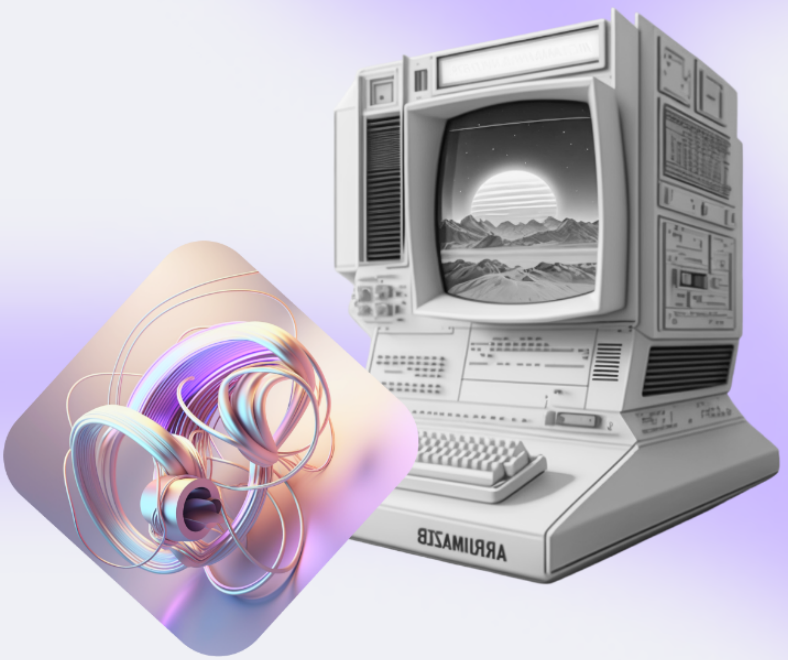
GeekBrains



Representation learning

Transfer learning



Использование моделей как трансформатора признаков

Введение

Всем доброго дня! Это вторая лекция курса по переносу обучения и сегодня мы поговорим про эмбеддинги. Тема довольно наукоемкая, но попробуем не забывать, что наша с вами цель – научиться делать реальные продукты, используя нейросеть в качестве ингредиента.

Рассмотрим задачу. Скажем, есть таблица отзывов о каком-то продукте, и нужно разделить их на положительные и отрицательные, чтобы узнать их соотношение. Потом продукт будет развиваться, публика будет меняться, но через два месяца вы хотите снова провести этот анализ и понять, в какую сторону изменилось соотношение. Как бы вы решили эту задачу? Можно вручную составить словари положительных и отрицательных слов, например. Такие словари в разное время разрабатывались разными коллективами, можно поискать тезаруса эмоциональной окрашенности слов. Предположим, в тексте встречается два положительно окрашенных слова и одно отрицательно окрашенное, и наверное чтобы получить оценку нашего короткого текста, логично суммировать эти оценки и разделить на количество слов.

Если на платформе преобладают простые и лаконичные отзывы, где по одному-двум словам понятно, о чем речь, то такой метод стоит попробовать. Представим, что мы аналитики и пробежимся по датасету. Какой процент ошибок вас устроит? Если отзывы немного сложнее, из целых предложений, вы можете захотеть вытащить из них информацию о том, какой аспект продукта обсуждается. Если это наушники, то отзыв может касаться качества звука, или удобства управления, или формы чехлов...

Но пока просто подумаем о продуктовой стороне дела. Если вы хотите, чтобы нейросеть разметила отзывы о вашем продукте на положительные и отрицательные, то как это должно выглядеть на практике? В этой задаче перед вами длинная таблица, в каждой строке – номер отзыва и его текст. После вашей обработки, в таблице должен появиться ещё один столбец, в котором будет одна

метка – положительный отзыв или нет. Один или ноль. Наверное классов будет больше, ведь все непонятные отзывы попадут в промежуточную категорию.

Фактически, вы создадите новый слой информации, решив некоторую когнитивную задачу, подавая на вход информацию из датасета. Простыми методами можно отделаться, если у вас простого вида отзывы и сравнительно простая задача. Но ведь отзывы могут быть двусмысленными или быть написаны с ошибками. Если текст усложняется, то уже не получится так просто отделить хорошие отзывы от плохих и ваша промежуточная категория будет расти.

Было бы удобно, если бы смысл параграфа текста можно было универсально представить какими-то числами. Крупинки смысла, которые можно выхватить из текста, чтобы затем работать с векторами чисел, а не с параграфами и морфемами...

Такие методы есть, но пока наука дистрибутивной семантики не породили одного универсального представления смысла слов или фраз. Зато есть множество предобученных сетей, которые можно использовать для выражения смысла текста, правда каждый раз это численное представление будет специфично для той задачи, под которую обучали нейросеть. И если модель обучалась на основе новостных текстов, она кое-что будет понимать и про весь язык, но лучше всего будет выражать смысл новостных текстов, и вряд ли сможет качественно классифицировать поэзию, юридические договоры или медицинские диагнозы. Для этого есть другие цели.

Уход от простых методов в сторону нейросетей решил несколько проблем. Например проблему необходимости разработки алгоритмов заново под каждую задачу. Классические методы, такие как, например, словари ключевых слов или вручную составленные правила принятия решений применяются уже лет 50. И пусть они будут работать хуже, чем последние версии GPT, но зато вы будете точно знать, какие конкретно правила лежат в основе алгоритма принимающего решения. Использовать объяснимые алгоритмы – это хорошо, это упрощает проверку сети, то есть валидацию. На самом деле, любая автоматизация принятия решений несет с собой риски, и солидные организации, например банки, когда рассчитывают выгоду и потери от внедрения того или иного решения, учитывают так называемый "модельный риск". Так что банки не боятся использовать нейросети.

Вспомните, о чём мы говорили о прошлой лекции этого курса. Предобученные нейросети позволяют проделать вычисления заранее, как бы законсервировать их, а потом воспользоваться ими как кирпичиками разрабатываемого решения.

Может быть, нейросети позволяют не только упаковать вычисления, но и как-то структурировать смысл, заключенный в объектах набора данных? Может быть можно использовать этот промежуточный процесс без переобучения? И как на основе таких векторов смысла делать полезные продукты? Вот на эти вопроса мне и хочется ответить на сегодняшней лекции.

Преобразование признаков

Давайте поговорим немного о простых методах преобразования признаков. Рассмотрим пример, который показывает, как важно выбрать правильный набор признаков.

Вот перед вами линейного различимые облака точек. Ну, скажем, что красные точки – это исторические здания, а синие точки - это новая застройка, и вам нужно геометрически разграничить два района. Ничего проще, проведём линию. Как мы нашли её параметры? Мы могли выполнить линейную регрессию и найти параметры линии при помощи метода наименьших квадратов. В таком случае мы подставили координаты всех точек в уравнения, и решив его получили координаты линии в форме AX + b. Плюсы и минусы всех методов зависят от поставленной задачи, так что тут начинаются субъективные аргументы. Можно сказать, что минус этого метода, логистической регрессии, состоит в таком факте: точки, далёкие от разделяющей плоскости, тоже влияют на решение. В зависимости от их распределения у нас могут получаться различные линии.

Если бы перед нами стояла бы задача построить дорогу определённой ширины между двумя районами, то лучше было бы выстроить задачу поиска разделяющей линии немного по-другому. Что было бы лучше сделать? Наверное, определить несколько домов, которые наиболее близки к границе между этими двумя группами зданий, и рассчитать форму дороги, опираясь на эти ближайшие точки. Остальные точки нас бы не волновали. Этот метод называется методом опорных векторов. Под векторами здесь понимается векторное описание некоторых объектов, которые мы отбираем в качестве опорных. У этого метода интересный алгоритм решения, я хотел бы чтобы вы в него погрузились когда-нибудь, но что про него сейчас важно

сказать – он исключает точки датасета, одну за другой, до тех пор, пока это не влияет на качество разделения. Таким образом алгоритм отбирает для нас те несколько точек, которых достаточно, чтобы провести между двумя классами разделяющую линию.

А что, если исторические здания слегка перемешаны с новой застройкой? Если мы согласны на некоторое падение качества решения, лишь бы не строить слишком изгибающуюся дорогу, то мы можем сказать себе, что у нас просто линейно неразделимый случай задачи и согласиться с той ошибкой классификации, которую нам даст метод опорных векторов.

И третий случай, который мы рассмотрим в этом примере, всё ещё оставаясь в парадигме метода опорных векторов – вот такой. Исторические здания окружены новой застройкой. Где и как мы проведём разделяющую линию? Понадобится ли изменять алгоритм принятия решений? Может быть классификатор просто решит построить линию где-нибудь за пределами города и не задевать ничьих интересов? Нет, Мы поступим сложнее, уйдём в третье измерение и построим разделяющую поверхность, которое я в трёх измерениях всё равно будет задано линейной зависимости координат. Нашей третьей координатой будет глубина. Смотрите, мы по-прежнему используем линейную разделяющую поверхность, потому что нашли, как нам трансформировать пространство признаков наших домов. Раньше для идентификации каждого дома служили его координаты на плоскости, а потом стали невразумительным, но более адаптированным к задаче набором чисел. Этот приём называется переход от скалярного произведения к произвольным ядрам, или kernel trick, и он позволяет строить нелинейные разделители, не меняя алгоритма принятия решений.

Так, вернёмся к цели этой лекции – научиться делать что-то с предобученными нейросетями. Какие выводы мы можем сделать из представленного примера?

Скажем так: если ваши данные выглядят сложно, то совсем не обязательно усложнять решающее правило. Возможно, есть такой способ трансформации данных, их векторного описания, который позволит решить задачу при помощи очень простой модели. Мы немного говорили об этом на прошлой лекции: большую часть нейронной сети можно представить, как машину для преобразования признаков, или встраивания численного описания входных объектов в какое-нибудь новое пространство.

Обучаемые признаки

Наверное, у вас возник вопрос, как мы выбрали середину каждой группы домов. Ведь можно выбрать любую точку. А как мы нашли центр класса, чтобы рассчитать в этом месте новый признак – глубину? Ведь, если бы мы выбрали другую точку, пример бы совершенно точно не сработал. Координаты этой точки можно найти при помощи оптимизации. Да, наше векторное представление можно называть выученным". Хотя, конечно, если мы зададимся новым распределением домов с новыми координатами, этот метод придется обучать в нуля.

Есть много способов решить этот пример ещё проще. Но, в постановке с опорными векторами, вы можете выбирать разные методы дополнения признаков описания, перебирать их, пока не найдёте оптимальное описание. Да что там, вы можете породить все возможные производные признаки, насытить ими датасет, скажем, теперь, вместо двух признаков у вас будет 2.000. И следующим шагом вы, скорее всего, захотите получить более простое векторное описание, чтобы каждая операция над датасетом из десятка тысяч объектов, каждый из которых представлен тысячами признаков, занимала незаметное время, то есть не приводила к минутным задержкам. Так обычно поступают с табличными данными, или с данными о временных рядах, когда, в отличие от картинок, у вас может не быть внятного представления, как вам вообще обрабатывать эти данные, раз они неструктурированные. Приходится либо изобретать собственный способ обработки, либо перебирать множество универсальных. Тут мы находимся в сфере инженерии признаков, про которую вы больше сможете узнать на лекциях по анализу данных.

Но и сама возможность работать с предобученными нейросетями может потребовать предварительной подготовки данных. Например, в давние времена, когда лингвистические задачи решались с нуля, строки текста порой нужно было избавить от пунктуации, заглавных букв и прочих вещей, которые для решаемой задачи были несущественными. Сегодня, при использовании больших при добученных лингвистических моделей, лучше не делать такой предобработки, она может всё испортить. Если предобученная модель привыкла, что имена людей и названия городов пишутся с большой буквы, а вы предобработали текст, избавившись от всех заглавных букв, сделав весь текст строчным, то качество работы предобученной нейросети на таких данных может сильно упасть по сравнению с данными без предобработки.

Многослойные нейросети

Перенос обучения на классических алгоритмах – векторных машинах, линейной регрессии, деревьях и бустингах – это тема для исследователей. На практике перенос обучения интересно делать только на базе нейросетей. Да и вы скоро поймете, что это не так интересно. Любой из этих алгоритмов обучается довольно быстро, так что проделывать вычисления сильно заранее не имеет большого смысла. Да и публикаций, рассматривающих перенос обучения в практическом плане на классических алгоритмах – очень мало.

Очень много больших достижений по обработке информации выполнены при помощи нейросетей. Попробуем и мы заглянув внутрь нейросети и узнать, какого рода интуиция может сохраниться в этой модели после обучения.

У нейросетей есть большая проблема – сложные нейросети для сложных данных обучаются неделями на мощном промышленном железе. А значит обучать такие сети с нуля можно только, если вы готовы потратить небольшое состояние на аренду мощностей, или уже как-то связаны с лабораторией, которая располагает ими. В отличие от классических алгоритмов, обучение нейросетей на сложных данных – это долго. Естественно, когда этот процесс завершен, хочется чтобы каждый шаг обучения, на которой было потрачено время, приводил бы к полезным результатам во множестве будущих проектов.

Когда вы обучаете сеть классифицировать животных, какая особенность нейросетей позволяет вам переназначить ее для классификации людей, предметов, или погодных явлений на спутниковом снимке? Пожалуй, то, что все внутренние параметры нейросети вам доступны – не только выходное решение, но и все данные, которые нейросеть вырабатывает внутри себя, и которые позволяют ей принимать высокоуровневые решения, преобразуя данные об исходном объекте в выходные данные. Но этот процесс происходит не за один прием, а в течение ряда преобразований, которые происходят с данными на каждом слое нейросети.

Вот, например, нейросеть ResNet50. Она имеет более 25 миллионов параметров. Это прекрасная сеть, индустриальный стандарт для множества задач компьютерного зрения. Сегодня ее обучают на массе разных датасетов, но допустим, кто-то обучал эту сеть на стандартном датасете ImageNet. Задача этого датасета – классифицировать изображение в один из тысячи классов. Итак, на обучение нейросети потрачено много сил и времени, и нас интересует то, что нейросеть научилась понимать про все изображения из датасета. Раз мы

обрабатываем изображение, то первые слои, видимо, имеют дело со светлыми и темными пятнами на картинке. Потом идут слои, которые уже что-то понимают о соотношении этих соотношений... Значит где-то в сети есть промежуточное представление, которое кое-что понимает о том, есть ли в кадре мебель, или представлен ли на нём живой организм, или неодушевленный объект.

Промежуточные представления изображения имеют значительную ценность. И настолько большую, даже, что ResNet часто берут в качестве компонента очень сложных продуктов, которые генерируют новые ракурсы фотографии, или просто отслеживают автомобили, или судят, насколько реалистичное изображение передано в систему... Как работает такой продукт, классифицирует ли он входное изображение на 1000 классов? Конечно нет. Но нейросеть в этом продукте используется для того, чтобы сделать с изображением что-то весьма полезное для последующего анализа.

Сейчас мы поговорим о том, как преобразовывать данные при помощи предобученных нейросетей, а на семинаре вы попробуете поработать с разными внутренними слоями нейросети и преобразовывать стиль изображения. Это приложение называется перенос стиля, и он тоже опирается на перенос обучения.

ResNet, эмбеддинги

Итак, мы уже некоторое время говорим об информации, которая располагается в разных слоях нейросети. Допустим, мы подаем на вход нейросети картинку. Спустя пять слоев мы получаем ряд чисел, который получен в результате цепочки преобразований, свойственных этой нейросети. Этот набор чисел называется эмбеддингом.

Эмбеддинг, или внутреннее представление, или латентное представление объекта, происходит от английское слова to embed, что означает встраивать, осуществлять проекцию.

Все алгоритмы машинного обучения, в большей или меньшей степени, занимаются тем, что переводят задачу в математически разрешимый вид. Глубокие нейросети помогают извлечь интересные признаки, затем много шагов подряд осуществляют их преобразование... Сложно вручную создать алгоритм, который преобразует матрицу пикселей, которой является рисунок или фотография, в вектор

вероятностных распределений, по которому можно судить, какой объект вероятнее всего находится в кадре. Поэтому нам так важная та приятная особенность нейронных сетей, которая заключается в том, что они сами умеют обучаться тому, какие признаки лучше извлекать с точки зрения поставленной задачи.

Если задача поставлена не точно, или размеченных данных не хватает, то можно просто попробовать максимально растащить нужные и ненужные объекты, или объекты всех интересующих нас классов, в пространстве признаков. Если вы изучали различные задачи машинного обучения, такого рода самообучение можно представить как кластеризацию: мы стараемся различить кластеры на какой-то поверхности, но при этом можем изгибать эту поверхность так, чтобы данные можно было разграничить какой-нибудь простой математической функцией.

В этой теме ещё полно интересных ответвлений. Сегодня я надеюсь обсудить с вами, откуда брались векторные описания для всех видов данных, для которых сегодня создаются предобученные нейросети. Изображения, текст, голос – всё это – данные, для которых существуют самые разные признаковые представления. Некоторые использовались и до появления глубоких нейронных сетей, и после – например, спектры для анализа звука. И мы поговорим как о нейронных признаковых описаниях, так и о классических, так называемых рукотворных способах описания объектов.

Эмбеддинги изображений

Что такое эмбеддинг изображения

Итак, векторное представление – это набор чисел, который описывает все важные для вас свойства какого-то объекта из датасета. А эмбеддинг – это результат трансформации векторного описания, который должен сделать ваши данные более полезными. Это очень важная концепция в глубоком обучении. Другое название для эмбеддингов – латентное представление вектора, или представление в латентном пространстве.

Значит, мы представляем объект в виде цифр. Картинки в виде пикселей. Текст в виде индексированного словаря. Что означает это преобразование в другое пространство, что мы пытаемся добиться?

Во-первых, часто эмбеддинги обычно занимают меньше места, чем исходные данные. Тут сложно дать пример, не уходя в архитектуру нейросети, но на предпоследнем уровне перед выходом собраны концепции, наблюдение которых позволяет нейросети решить, что на картинке изображён кот. На этом предпоследнем срезе данных однозначно меньше, чем хранится во всех фотографиях котов, для которых сеть вносит одинаковый вердикт. На предпоследнем срезе лингвистических нейросетей находится несколько сотен чисел информации, в которых закодирована взаимосвязь всех слов предложения со всеми словами известного словаря, так что в небольшой объём эмбеддинга получается втиснуть очень много данных.

А во-вторых иногда данные в форме эмбедингов лучше различаются, чем исходные необработанные данные, потому что они имеют меньшую размерность. Какая размерность пространство, где собраны все возможные картинки размером в 1 мегапиксель? Если взять цветную картинку с тремя каналами, и посчитать, что в одном мегапикселе находится миллион пикселей, то размерность этого пространства – 3 млн. Картинку в этом пространстве можно представить точкой с тремя миллионами координат. Это очень много. Конечно, настоящие картинки, на которых изображено что-то, привычное нам, находится в каких-то областях этого пространства. Вектор чисел на срезе предобученной нейросети обычно имеет размерность в несколько сотен или несколько тысяч элементов. Почему это важно, кроме того что приходится хранить меньше данных? Тут надо рассказать о проклятии размерности. Так получается, что данные в пространстве очень высокой размерности всегда расположены достаточно скученно. Я бы сказал так: имея 3 млн степеней свободы, вы всегда найдёте короткую дорогу между любыми двумя точками, а значит множество всех предметов будет трудно разделить геометрически. Если же, наоборот, удаётся снизить размерность пространства, сохранив связанность важных объектов, то проклятие размерности удаётся победить и важные данные становятся более различимыми.

Это применение появилось давно, и не в мире нейросетей. Аналитики давно пользуются методами снижения размерности многомерных объектов, например чтобы представить эти объекты на графике или чтобы изучить отдельные их взаимосвязи. На примере анализа основных компонентов, или коротко PCA, очень просто рассказать, как они работают, в общих чертах. Этот метод был описан у Карла Пирсона в 1901 году. У вас есть выборка данных, если представить все признаки как координаты, то данные скорее всего будут располагаться в многомерном пространстве в форме облака, может быть с отдельными островками – мы не будем делать кластеризацию, поэтому топология облака не очень важна. Найдем в этом облаке наиболее явное направление, некоторую ось, вдоль которой

ориентированы все данные – это наша главная компонента. Вторая компонента описывает отклонение от этой оси, то есть движение вдоль какой-то из перпендикулярных осей, а помните что у нас многомерное пространство. Это только на графике у нас два направления, основное и второстепенное. В общем, мы выстраиваем новую систему координат, она уже не параллельная существующим осям, но каждая ось, в порядке уменьшения, соответствует направлению, вдоль которого данные наиболее вариативны.

На картинке, кстати, сейчас вы видите сайт paperscape. Это двухмерное представление всех статей на сервере научных препринтов, который называется arXiv. Это очень важный для науки сайт, и вы видите примерно половину статей, мне пришлось немного приблизить, чтобы появились надписи, но вы уже видите, что некоторые статьи заметнее других – размер круга соответствует цитируемости этой статьи. Я люблю изучать этот сайт, а для нас с вами сейчас это просто хороший пример уменьшения размерности.

Иногда эти направления характеризуют класс или еще какие-то признаки объектов. Но их не стоит воспринимать эти эмбеддинги как описание каждого отдельного объекта. Это метод анализа выборки. Эти методы позволяют, скажем, увидеть выборку на плоскости, либо отобрать 50 наиболее вариативных признаков для первичного анализа, но поскольку результат мало что говорит о каждом отдельном объекте, результат снижения размерности – это не эмбеддинг каждого объекта. Добавь мы несколько объектов в выборку, результаты могут кардинально измениться. Мы еще упомянем эти методы в конце лекции, но хотя они и составляли первые попытки автоматически сформировать признаковое описание сложных объектов, это не те эмбеддинги, о которых мы будем говорить сегодня. Современные нейросети позволяют делать гораздо более сложные вещи.

Как создаются нейросети с хорошими эмбеддингами

Как вообще находят структуру, которая обучает удобные эмбеддинги? Есть по крайней мере три способа.

Представим себя на месте проектировщика. Вы создали нейросеть, которая не очень хорошо отбирает признаки, при проверке работы обученной сети

Первый способ: начать перебирать архитектуры, например изменяя количество слоев и количество нейронов в каждом слое. У архитектуры есть параметры, и перебирать их, можно найти направление для улучшения. Это самый затратный

путь, при котором мы находим ту архитектуру, которая показывает наименьшую ошибку на тестовой задаче.

Второй способ. Если у вас нет данных в достаточном количестве, то проектировщик нейросети может попробовать сформулировать цель задачи как самообучение. Базовое представление в самых больших лингвистических моделях было получено на задачах без разметки. Эти модели, как мы посмотрим позже, в своей основе просто учились предсказывать пропущенные слова в тексте. Получается, что из неразмеченного массива текстов, в рамках задачи, мы сделали огромный размеченный датасет. Тоже самое сейчас делают с изображениями, хотя и аккуратнее. Разделяют изображение на небольшие блоки и учат нейросеть, по имеющимся блокам, предсказывать содержимое закрашенных блоков. Получается задача самообучения.

И третий подход осуществляется через кластеризацию и автоэнкодеры. Если у Вас совсем нет разметки для датасета, то вы можете попробовать научить нейросеть сжимать данные до небольшого векторного описания, а потом разжимать их обратно. Такая нейросеть учится восстанавливать входные данные, пропуская их через своего рода бутылочное горлышко нейросети. В самом узком месте такой архитектуры находится потенциально полезное векторное представление, которые очень часто используют как эмбеддинг. Количество признаков в таком векторе будет существенно меньше, чем в исходном объекте, и такие архитектуры обучают.

Я перечислил эти три метода с целью показать вам, что оптимизация нейросети – не лучшее решение, когда нужно выпустить продукт с нейросетью внутри. Лучше начать улучшение продукта с поиска статей и отбора готовых нейросетей, где сделана задача, похожая на вашу. Сегодня таких сетей очень много, и их можно перенацелить на что угодно.

Применения эмбеддингов изображений

Когда вы ищете обходное решение какой-нибудь когнитивной задачи, например хотите проверить, что на всех работниках стройки надеты средства индивидуальной защиты, или дать ответ на свободно сформулированный вопрос, вам нужно представить задачу в абстрактном виде, который обычно решается методами анализа данных. Когда будете искать нейросети под вашу задачу, попробуйте представить ее как одну из популярных обобщенных задач, и вам будет легче найти решение. Какие бывают обобщенные задачи в машинном обучении?

- Поиск, когда нужно найти подходящий результат, соответствующий некоторому запросу.

- Кластеризация, когда нужно объединить объекты в заранее неизвестное количество групп по их схожести

- Рекомендации, в этой задаче нужно определить пожелания субъекта и ранжировать объекты рекомендации по предпочтительности

- Детекция аномалий, когда мы предсказываем желаемый результат, определяем, насколько наблюдение отклоняется от него, и обучаем систему принимать решения, релевантно ли то отклонение, которое мы наблюдаем

- Измерение различий, то есть сравнение распределений признаков разных групп объектов

- Наша обычная классификация – отнесение объекта к одной или нескольким из заранее определённых групп

- Наконец первое, что проходят в машинном обучении – регрессия, сопоставление объекту некоторого числа из непрерывного ряда

Допустим, что вы – торговая площадка, маркетплейс, и вам нужна рекомендательная система. Иногда задачу невозможно решить напрямую, предъявив системе рекомендаций набор картинок и набор предпочтений. Аналитики в таком случае обычно должны предложить какое-то преобразование для картинок, какое-то для табличных данных клиентов, еще одно для текста, для любых объектов, которые есть в нашем распоряжении. Сегодня можно не придумывать такие описания, в выбирать между всеми готовыми. В мире достаточно предобученных сетей, из которых можно взять хоть все внутренние слои в качестве признаков и обучать на них любые классификаторы, или отбирать их в состав численного описания продуктов, на основе которых система будет выдавать рекомендации.

Пиксели

Следующие минут 15 я хочу посвятить разным форматам эмбеддингов для текста, картинок и других видов данных.

Мотивация здесь такая: в промышленности, где нет времени изобретать новые виды анализа данных, нужно пользоваться тем, что уже наработано и доступно. И, я вас уверяю, промышленность с удовольствием пользуется решениями, которые приносят прибыль, и среди них достаточно часто реализуется переход на нейросетевые эмбеддинги. Вам пригодиться навык включать в формулировку ваших задач такой этап преобразования, как выделение релевантных признаков при помощи нейросети.

Кстати, здесь на изображение я поместил перечень предобученных нейросетей, которые встроены в популярную среду технических расчетов как MATLAB. Вы можете увидеть, сколько занимает та или иная сеть, сколько в ней слоев, сколько миллионов параметров нужно было обучить, и какого размера изображения они принимают на вход. Хотя, на самом деле, вы можете подать в них изображение любого размера, но это требует некоторой инженерной сноровки. Так что, как видите, предобученные нейросети нужны и инженерам систем управления, и при обработке сигналов, и в других сферах, где применяется MATLAB.

Сейчас мы проследим небольшую эволюцию признаков для каждого вида структурированных данных, а в конце поговорим про неструктурированные данные, таблицы и графы. И вы сможете сформировать для себя интуицию: куда это всё движется. Не говоря о том что такой разговор подготовит вас к семинару, и вы уже не встретите там слишком непривычных терминов. Поехали, и начнём с признакового описания изображений.

Компоновки, ручные признаки

Лет 50 назад, когда зарождалось компьютерное зрение, инженеры и учёные пользовались для описания своих алгоритмов понятием графических примитивов. Часть используемых сегодня методов анализа изображений, например каскады хаара или преобразование хоафа осуществляют поиск таких примитивов. В одном случае это – поиск на изображении серии контрастных областей, так по сей день работают простейшие детекторы лица. В случае преобразования хафа, к картинке применяется координатная трансформация, то есть все пиксели получают новые координаты, и в местах изображения, где находятся окружности или линии,

пиксели скапливаются в более яркие пятна. Ещё один вид преобразования – переход в частотную область через преобразование фурье или вейвлеты. Для изображения, как и для звука, такое преобразование может решительно изменить качество выполненного решения.

Но всех этих методов было недостаточно, чтобы, например, детектировать на изображении сложную, изменчивую фигуру предмета. Если вы хотели детектировать на изображение футбольный мяч, который можно обнаружить, найдя белую фигуру в форме круга в заранее известной области экрана, то это задача решается просто. Но если вы ищете сложную геометрическую метку или фасад некоторого здания, чтобы локализоваться относительно него, то придётся вносить поправку в алгоритм на то, что здание может быть ближе или дальше, а объектив наблюдателя может быть повёрнут относительно здания и иметь разные геометрические искажения. Для работы с такими задачами были созданы методы SIFT, SURF, ORB и другие, так называемые устойчивые к изменению масштаба локальные дескрипторы. Для их создания учёные и инженеры тщательно отбирали признаки изображения – масштаб соотношения яркости, направленность гистограммы, несколько десятков локальных признаков изображения позволяют определить ключевые точки и задать для каждой точки сотню чисел, которые можно затем поискать на другом изображении и, при нахождении сочетания, найти трансформацию от одного изображения к другому.

Всё это – история компьютерного зрения, так сказать, эпоха рукотворных признаков. Посмотрим, как нейросеть создаёт своё признаковое описание картинки.

Эмбеддинги ResNet

Давайте пройдёмся по нейросети ResNet и посмотрим, как трансформируются данные на каждом этапе обработки. Напомню, что это нейросеть уже предобучена, мы не будем вычислять ошибки прогноза и на их основе, изменять веса нейронов в сети. Для нас они – константы. Перед вами алгоритм, в котором 25 млн констант, и мы сейчас посмотрим, что он делает с картинкой, которую мы подаём ему на вход.

Первым делом обсудим, что же у нас на входе. Наша картинка – это матрица из пикселей, чаще всего каждый пиксель имеет цветовое описание, состоящая из трёх каналов: красный, зелёный, синий, это rgb модель. Один и тот же цветовой оттенок можно описать другой комбинацией чисел, например hsv, в состав которого тоже входит три числа: собственно, оттенок, насыщенность и яркость. Это к слову о

разных признаковых описаниях. Потенциально, каждый пиксель может независимо влиять на интерпретацию всех других пикселей в матрице. Ну, чаще всего, ближайшие соседство каждого пикселя имеет большое значение для интерпретации того, что изображено в каждой порции нашей картинки. И чем ширя наша область интерпретации, тем более общую категорию мы можем присвоить тому, что мы видим на картинке. Если мы делали классификатор, который должен уметь отличать кошек от собак, сперва он обнаруживает локальные признаки, которые скорее говорят в пользу гипотезы о том, что на картинке изображена кошка или собака, а может быть, там вообще ничего не изображено. И потом из этих локальных признаков создаётся глобальное признаковое описание.

В больших графических моделях используются свёрточные слои, построенные на математической операции свёртки. И поэтому на первом слое нашей глубокой придобученной нейросети вы видите изображения, настолько похожие на исходное. Они получены путём применения десятков или сотен различных свёрточных операторов к трём каналам входного изображения. В нашем распоряжении сотня небольших картинок, на некоторых ничего нет, другие словно акцентируют внимание на вертикальных или горизонтальных линиях на изображении. Обычное дело, при помощи свёртки можно искать границы объектов на изображении. Пойдём на несколько слоёв дальше.

Изображение всё ещё отдалённо похожий на входную картинку. Видите, даже несколько свёрточных слоёв не слишком перемешали пиксели на изображении. Они и не могли этого сделать, поскольку каждое свёрточное окно, в неизменном виде, применяется ко всему изображению целиком. Эти операции только подсвечивают некоторые признаки входного изображения, которые потом будут иметь больше веса в финальном решении. Цепочка преобразований привела к тому, что некоторые из сотен маленьких изображений стали чёрными. Очевидно, признаки, которые на них бы отражались, в изображении Не встречаются.

Если вам попадались такие визуализации – изображения того, на что нейросеть обращают внимание на каждом своём уровне, то эти картинки получаются при помощи процедуры оптимизации запятая в которой мы пытаемся так изменить случайное исходное изображение, чтобы некоторые окошко признаков, например окно номер 50 на уровне номер девять, стало максимально светлым. Это тот самый, так называемый нейрона бабушки. Тот нейрон, который включается в вашем мозгу, Когда вы думаете о бабушке. Естественный мозг так не работает, но если нейросеть училась искать только кошек и собак, то в ней, скорее всего, есть такое окошко, которое наполняется белыми пикселями, если на изображении находится кошачий нос.

Если ещё углубиться, то мы увидим, что каждая картинка практически стала отдельным признаком. Дальше нейросеть раскладывает все эти картинки на pixel и просто отбрасывает их взаимное пространственное расположение. Все признаки выстраиваются в единую линию и обрабатываются как большой вектор признаков входного изображения.

В нейросети resnet 50 на выходе находится несколько полносвязанных слоёв. Их входы и выходы можно использовать отдельно от финальной классификации. Выученная нейросетью информация о признаках изображения, которая впоследствии позволяет принять решение о классе животного или характеристики объекта, изображённого на картинке, представляет собой очень богатая, сжатое, закодированное описание этой картинки.

Оно представляет из себя эмбеддинг этой картинки. Естественно, задача может потребовать, чтобы вы хранили эмбеддинги изображений в специальной базе данных и осуществляли по ним поиск. В будущем Мы ещё поговорим о таких векторных или признаковых базах данных, а пока поговорим об истории развития признаков описаний картинок.

Заметим, что мы говорили о глобальных представлениях, то есть анализировали всё, что находится на картинке. Мы находились в рамках задачи классификации. Для других распространённых задач – локализации или детекции, нужно более локальное представление, такое, например, которое даёт нейросеть YOLO.

Как их хранить?

Возможно, у вас уже начали появляться идеи-что можно сделать с признаковым описанием. Пример, так может работать поиск изображений в базе данных. Ведь искать по эмбеддингу проще, чем искать по самой картинке. И, наверное, этот метод будет работать точнее, чем старый мультимодальный поиск, когда картинку из интернета находят по тексту, который её окружает. Поисковиках эти методы используются совместно, а эмбеддинги по-настоящему являются мультимодальными, то есть объединяющими разные модальности данных.

Для хранения эмбидингов существует специальные векторные базы данных, например faiss. Они оптимизированы для хранения и поиска по векторам, а не по реляционным таблицам.

Это ещё один пример того как применить нейросеть на практике: сгенерировать представления всех объектов, которые вам важны, и использовать их как координаты нового пространства, например, для поиска ближайших соседей некоторого нового изображения, или для аутентификации сотрудников при допуске к помещению, хотя для получения эмбеддингов лиц существует нейросеть FaceNet.

Генерация из эмбеддингов

Давайте поговорим об обратной операции – как можно собрать объект на основе его эмбеддинга.

Самое простое, конечно, это поискать ближайший объект в базе данных. Это делается с помощью поиска ближайших соседей для вашего многомерного эмбеддинга. Этот механизм – ровно тот же, который позволяет найти растение в базе данных по какой-нибудь пользовательской фотографии, или находить в интернете картинки, похожие на заданное изображение.

Теперь допустим, что у нас есть не только эмбеддинг, но и нейросеть которая его породила. И мы хотим узнать, какое изображение нужно подать на вход, чтобы получить этот конкретный эмбеддинг. Почему это не имеет смысла делать без коэффициентов конкретной нейросети? Естественно, потому что эмбеддинги ничего не значат в глобальном поле всех возможных численных последовательностей. Эмбеддинги свойственны только конкретной нейросети.

Первая мысль состояла бы в том, что нужно просто пройтись парой тысяч итераций, используя дальность текущего эмбеддинга от желаемого в качестве критерия оптимизации. Происходит такая визуализация через оптимизацию. Так работали ранние генераторы изображений, например DeepDream, с которым с некоторых пор можно было поиграть, населив входное изображение вкраплениями любой природы, например мордами собак или цветами.

Вторым этапом, мы можем запрограммировать такую оптимизацию, которая заставляет наш генератор искать различные изображения. Ведь одно сгенерированное изображение может не отражать всех аспектов эмбеддинга, мы же обсуждали, что в эмбеддинге может быть запутано много концепций. И все равно у таких генераторов было много проблем, например они имели тенденцию создавать абстрактные изображения с высокочастотным шумом, который имел нужный эмбеддинг, но на котором не было ничего различимого.

Как же устроен современный алгоритм генерации картинок, вроде Stable Diffusion? Я вам скажу, что вы приближаетесь к его пониманию. Ведь в нем используется несколько архитектур, для которых доступны предобученные сети, например CLIP – это нейросеть, которая создает эмбеддинги для текста, но обучается совместно на тексте и изображениях. К примеру, как первый этап обработки текстового запроса, нейросеть CLIP передает в следующий алгоритм что-то вроде 77 векторов по 768 параметров в каждом. Очень большой набор эмбеддингов.

В том что касается текста – тут все проще. Задачи которая практически всегда решается через нейросеть-генератор, использующая свое внутреннее состояние чтобы сохранять контекст сказанного, то все что нужно сделать, чтобы сгенерировать текст по эмбеддингу – задать это эмбеддинг в качестве внутреннего состояния нейросети и запусить ее на выполнение. В современных диалоговых нейросетях, это внутреннее состояние может быть заложено разработчиком и доработано какой-нибудь спрятанной от вас строкой-затравкой. Например, разработчики сервиса на основе этой нейросети сперва объяснили ей, каким персонажем она является, что может и что не может говорить... И только после этого вы получаете в свое распоряжение текстовую строку, куда можно добавлять свои слова (ваш запрос) и смотреть, как они меняют внутреннее состояние нейросети, и что она производит в ответ. Так что вот так можно создавать диалоговые агенты.

На следующем занятии поговорим о том, как работают генераторы из текста в изображения и обратно, мультимодальные генераторы.

История признаковых описаний для текста

Словарь, one hot encoding, частота слов, TFIDF и автоэнкодеры

Теперь о том, как анализировать текст. Самые старые диалоговые системы были созданы вручную, а вычленение важных признаков из них осуществлялось регулярными выражениями. Соответственно, если на тексте срабатывает некоторое регулярное выражения, например – наш текст содержит слова хороший или

отличный в разных формах, то мы автоматически классифицируем отзыв как положительный.

Машинное обучение, как всегда, позволило работать с более сложными взаимосвязями, и потребовалось как-то соотносить изучаемый текст с всем известным словарём. Иногда этот словарь составлялся из всех слов датасета. На таком словаре можно построить алгоритмы, оперирующие частотами слов. При организации частотного анализа важно, как часто в тексте встречается какая-то группа слов. Можно разделить встречаемость слов в данном тексте на встречаемость слов во всём обработанном корпусе текстов и получить представление текста в виде TF-IDF. Это сокращение так и читается: частотность термина относительно обратной частотности терминов во всех документах. Чем уникальнее слово, тем лучше оно характеризует данный текст – такое соображение положено в основу частотного анализа при помощи TF-IDF. Напомню, что мы всё ещё говорим о признаках вам описаний текста, то есть о том, как нам преобразовать текст в числа, оставив только самое важное, и начать изучать наш объект математическими методами. TF-IDF позволяет нам ранжировать слова каждого текста и составить из самых важных слов, так сказать, мешок. При представлении в виде мешка слов, мы пытаемся охарактеризовать текст через слова, которые в нём употребляются, не обращая внимания на расположение слов в фразе и на их взаимный контекст.

Грубо говоря, если в тексте о какой-нибудь коммерческой компании фигурирует слово акула, которого нет больше ни в одном описании другой компании, то мы можем смело отнести её к рыболовецким компаниям, или к защитникам прав животных. Даже, если слово акула входила в словосочетание акула шоу-бизнеса, что, определенно, создаёт весьма особенный контекст его употребления. Что же, можем рассматривать неотдельные слова, а их группы – N-граммы. Если построить алгоритм на всех встречаемых в корпусе биграммах, словарь нашего алгоритма очень быстро станет очень большим. Нужно ли хранить так много информации?

word2vec и GloVE

Примерно в то же время, как и в области обработки изображений, в обработке текста случились свои небольшие революции. Появились довольно универсальные методы, которые сохраняют в векторном представлении слова его контекст в предложении. Для этого нужно было научиться обрабатывать крупные датасеты на появившемся тогда оборудовании для параллельных вычислений, и несколько коллективов с этим справились, дав нам word2vec и GloVE.

В основе этих методов лежат нейросети, и эти два метода немного по-разному преобразуют слова предложения в векторы, позволяя сохранять следы окружения каждого слова и не ошибаться в контекстно-зависимых ситуациях. А ещё из этих нейросетей можно вытащить таблицу м бедингов для каждого слова, что позволит делать математические операции над словами. Например, после появления этих сетей, в в блогах и новостях про обработку данных стали демонстрировать пример решения словесной пропорции, когда вы в прямом смысле можете выполнять арифметические операции над словами. С поправкой на пространство, в котором находятся вектора слов, смысловая пропорция подразумевает, что взяв слово король, вычтя из него слово мужчина и прибавив слово женщина вы окажетесь в такой точке латентного пространства, что ближайший точкой в таблице эмбеддингов будет слово королева. Это демонстрация зажгла исследовательское сообщество, и не пришлось долго ждать до дальнейших этапов развития лингвистических нейросетей.

Но для нас сейчас важно, что эти два метода позволяют нам преобразовать текст в числа, с которыми мы можем что-то делать классическими методами машинного обучения, или вовсе арифметики. При ограниченных ресурсах, можно преобразовать каждое слово текста в word2vec или glove, затем взять максимум от всех полученных векторов (это называется операция агрегации) и посмотреть, как соотносится такое представление для разных групп слов, которые вы хотите различить в рамках, скажем, классификатора отзывов о продуктах.

Добавим ещё одну прекрасную особенность: обучение этих нейросетей происходило без разметки, то есть здесь, возможно впервые, получилось обучить нейросеть на тексте всего интернета – запутанном, разнообразном, и уже точно никем не размеченным на правильные и неправильные тексты.

BERT

Следующий этап автоматизации обработки текста – нейросети bert и эмбеддинги, которые из них можно получить.

В октябре 2018 года нейросеть BERT была самой большой на тот момент обученной лингвистической моделью, с ее 3.5 миллионом параметров. Конечно, с некоторой отметки, сложно отличить одну модель от нескольких. Современные огромные модели с миллиардами параметров часто оказываются ансамблями моделей, собранных из нескольких моделей поменьше, и это вполне легитимный способ

усилить работу моделей, улучшить качество, или позволить на ходу выбрать один результат из нескольких предложенных.

С помощью эдмбеддингов BERT стало возможно решать задачи ответа на вопросы, предсказания слов (то есть, создание чат-ботов) и суммаризации текстов, то есть обобщение смысла текста несколькими словами.

Эта нейросеть уже работает не со словами, а с токенами – это важный термин для анализа текстов. Токеном является всё, что подаётся на вход нейросети. Это могут быть группы слов, слова или частички слов, даже отдельные буквы. Так что словарь этой нейросети – не совсем словарь в нашем понимании. В него попали наиболее часто встречающиеся группы символов и отдельные символы, позволяющие экономно представить все существующие и несуществующие слова языка. А значит, при вызове этой нейросети вы получите набор эмбедингов по всем входным токенам, то есть не по словам, а по группам символов. Вот такая небольшая интуиция.

Чтобы получить представление предложения или целого текста, принято, как я уже говорил, выполнять агрегацию эмбедингов. Для этого можно усреднить, взять максимум или минимум от всех эмбидингов, которые нейросеть вернула для входного предложения или текста. Останется ли какой-нибудь смысл в этой компоновке – зависит от того, на каком наборе данных учился конкретный вариант нейросети bert, которая используется в анализе. Это может быть русская или мультиязычная сеть, юридическая или медицинская... взяв базовый вариант такой нейросети и до обучив на вашем домене можно значительно поднять точность алгоритма именно на ваших примерах.

BERT стала очередной сота моделью, при этом получить эмбеддинги этой нейросети предельно просто, сама это нейросеть сравнительно небольшая: порядка гигабайта, и, если подумать, демократизация задач, которые решались с помощью этой нейросети, свидетельствует не только о её отличной способности анализировать естественный текст. Как вы знаете, сегодня гораздо лучшее это делают модели типа GPT. Количество решаемых задач при помощи BERT скорее свидетельствует о том насколько просто и предсказуемо работать с этой моделью. Она открыла путь и предвосхитил развитие следующего витка технологий, но по-прежнему очень много где встречается.

GPT и API

Если размер эмбединга нейросети bert составлял 768 чисел, то только лишь первая версия GPT уже использовала больше десятка тысяч значений. Такие крупные лингвистические модели часто используют через API. Их не только очень трудно дообучить для локальной задачи, но и даже запустить в собственном окружении. При таких больших эмбедингах, хранить их в векторной базе данных очень накладно. При этом, несомненно, такие представления хранят много информации о смысле не только самого текста, но и о его связи со всеми остальными текстами интернета. Но что делать с этой информацией – интересная продуктовая задача. Обычно, всё-таки, такие огромные лингвистические нейросети используются в диалоговом режиме.

Более простые модели, вроде Glove или Word2Vec, работают гораздо быстрее, но учитывают так мало информации о контексте, что на их основе вряд ли можно создать сколь угодно удовлетворительный чат-бот. Но они тоже включаются в соревнования на современных датасетах, и вы можете посмотреть в библиографии одно исследование, которое называется MTEB, небольшую часть которого я показываю здесь, на экране. Это очень обширное исследование, где цель – исследовать качество эмбеддингов разных нейросетей, включая BERT, на самых разных задачах: классификация, суммаризация... Можно выбрать лучшую нейросеть, хотя за последние 20 минут, я думаю, вы получили хорошее представление о самых доступных из них, и сами можете судить об их преимуществах и недостатках.

Аудио и другие виды информации

Спектры, вейвлеты, MFCC и нейронки

Завершая разговор на тему эмбедингов структурированных данных, хочется сказать пару слов про обработку звука. Рукотворными признаками в этой области являются различные спектры, в том числе MFCC. В промышленной практике часто используется такой приём: преобразовать аудиосигнал небольшой длины в двухмерный спектр, например, при помощи вейлетов, а затем классифицировать

звук или слово при помощи нейросети, созданной для анализа изображений. Странно, но часто это даёт прекрасные результаты.

Выводы

Почему tSNE не подходит на роль выученных эмбеддингов

Давайте на секунду вернемся к задаче уменьшения размерности и обсудим один метод, в названии которого есть слово "эмбеддинг", но который нельзя использовать в качестве универсального преобразователя признаков, в отличие от тех нейросетей, которые мы рассмотрели. Этот метод называется стохастическое вложение соседей с t-распределением, или tSNE. По-английски этот метод называется t distributed stochastic neighbor embedding.

Мы встречаем знакомое нам слово эмбединги. И правда, этот метод служит для помещения многомерных данных в пространство меньшей размерности. Часто он используется для визуализации многомерных данных на плоскости. Вот пример: это карта библиотеки статей под названием arviv.org. Связи между статьями устанавливается методами анализа естественного текста, то есть можно представить, что статьи сперва переводятся в векторное представление путём, например, подсчёта редких слов в каждой статье и формировании из этих слов некоторого индекса, который позволяет найти эти слова в едином общем словаре. Таким образом каждый документ может получить длинное векторное представление, благодаря которому документы можно будет сравнивать между собой.

Tsne - это метод снижения размерности. Как и pca, ica, или применение нейросетей с бутылочным горлышком посередине, например автоэнкодеров. Естественно, полученные признаки в уменьшенном пространстве сохраняют какую-то информацию об исходных объектах. На её основе тоже можно рассуждать о взаимосвязи объектов, например, если мы сократим количество размерностей до двух, мы сможем визуально проверить качество кластеризации, или ещё какие-то параметры структуры объектов в датасете.

Однако, эти методы лучше не использовать для создания устойчивых векторных представлений. Ведь, вспомните, модель машинного обучения может оказаться в проблематичном состоянии переобучения. Это когда для всех объектов, которые этой сети попадались раньше, в ходе обучения, результаты находятся с высокой

точностью. Но стоит только подать на вход новые объекты, которые эта сеть раньше не видела, как качество сильно падает. Сеть могла не заниматься обнаружением взаимосвязей и признаков, характеризующих наши объекты с точки зрения задачи. Например, такая сеть не выучила, чем кошки отличаются от собак точка она просто запомнила все фотографии кошек и собак, а на реальных объектах результат классификации будет очень неустойчивым.

Так и методы снижения размерности хорошо работают только с той выборкой, над которыми они проводили свои геометрические операции. Новые объекты могут оказаться в совсем непредсказуемом пространстве признаков. Так что, например, при помощи tsne лучше не браться создавать базу данных по признакам, которые характеризуют фотографии кошек. Эти методы не занимаются обобщением. А при улучшение признаков его описания Мы хотим избавиться от лишней информации, оставить только необходимое, чтобы задачу можно было решить какие-нибудь простым алгоритмом на выходе, например, просто проведя линию между классами.

Заключение

Мы поговорили о распространённых представлениях представлениях для структурированных данных – как они были получены, как развивались, какие сегодня часто применяют.

Что же с неструктурированными данными? Со всеми теми объектами, которые не имеют заранее фиксированного размера, словаря, или которые трудно преобразовать в картинку?

Бизнесу очень интересно анализировать таблицы с отчётами и графы взаимосвязей, которые возникают в ходе анализа данных. Увы, есть много приёмов для работы с табличными данными, есть даже специальные графовые нейросети, то есть, теоретически, бизнес может обучить нейросеть делать что-то полезное с цепочками данных в таблицах или в графах. Но, кажется, что индустрия переиспользуемых расчётов тут только зарождается.

Пример из жизни: сейчас в законодательном поле только начниают пытаться обобщить агрегацию персональных данных, которые позволят работать совокупностью таких данных, не нарушая законные интересы владельцев. Банки и госучреждения пробуют алгоритмы, создают ГОСТы, пишется законодательная база. Всё для того, чтобы обезличенную информацию можно было использовать для анализа, совмещая разные источники информации, при этом обеспечивать низкий

риск утечки сопоставленной персональной информации. Не технической утечки, когда кто-то подслушивает трафик или получает незаконный доступ к базе данных, это одно поле методов. А именно возможность идентифицировать человека по персональным данным на пересечении нескольких таблиц, как бы они ни были обезличены. Здесь могут помочь табличные, графовые и другие эмбеддинги.

Другой пример, организации, располагающие огромными массивами данных про субъекты бизнеса, обучают нейросети представлять информацию о контрагентах и их взаимосвязях при помощи какого-то ряда чисел. Вы уже знаете что речь идёт про эмбеддинги. Такое представление позволяет классифицировать мошеннические действия или различить между сотнями одноимённых организаций или индивидуальных предпринимателей, чтобы предоставлять им более адаптированные услуги.

Пока в этой области перенос обучения возможен только внутри самой организации, и если мы с вами представляем небольшой коллектив, то возможно, что результаты такого предобучения нам с вами слишком дорого достанутся. Но если вы получили или собрали большой релевантный датасет, то создание на его основе предобученной нейросети – неплохой способ продуктивизировать данные, то есть заставить их работать и приносить прибыль.

В следующем семинаре мы поговорим о том, как взять эмбеддинги из предобученной лингвистической нейросети и, без особого машинного обучения, решить несколько интересных задач из области анализа естественного текста. До встречи!