I. МАШИННЕ ОБУЧЕНИЕ. ТЕОРИЯ

Зачем обучать машины

Разберём на примере из жизни

Предположим, Олег хочет купить автомобиль и считает сколько денег ему нужно для этого накопить. Он пересмотрел десяток объявлений в интернете и увидел, что новые автомобили стоят около $20 000, годовалые — примерно $19 000, двухлетние — $18 000 и так далее.

В уме Олег-аналитик выводит формулу: *адекватная цена автомобиля начинается от $20 000 и падает на $1000 каждый год, пока не упрётся в $10 000.*

Олег сделал то, что в машинном обучении называют *регрессией* — предсказал цену по известным данным. Люди делают это постоянно, когда считают почём продать старый айфон или сколько шашлыка взять на дачу.

Но кроме пробега у того же автомобиля есть десятки комплектаций, разное техническое состояние, сезонность спроса и еще столько неочевидных факторов, которые Олег, даже при всём желании, не учел бы в голове.

Теперь машина смотрит на наши данные, ищет в них закономерности и учится предсказывать для нас ответ. Самое интересное, что в итоге она стала находить даже такие закономерности, о которых люди не догадывались.

Так родилось машинное обучение.

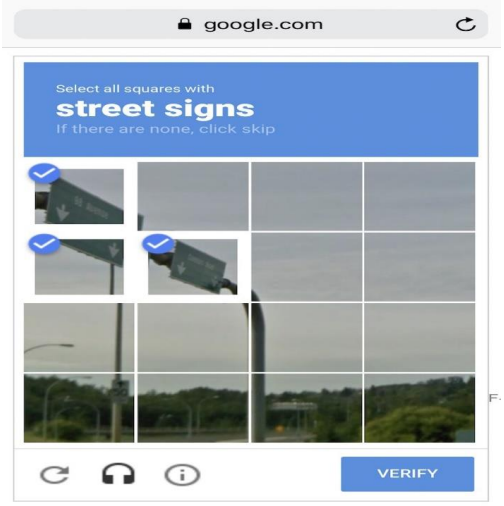
Три составляющие обучения

Цель машинного обучения — предсказать результат по входным данным. Чем разнообразнее входные данные, тем проще машине найти закономерности и тем точнее результат.

Итак, если мы хотим обучить машину, нам нужны три вещи:

Данные

Хотим определять спам — нужны примеры спам-писем, предсказывать курс акций — нужна история цен, узнать интересы пользователя — нужны его лайки или посты. Данных нужно как можно больше. Десятки тысяч примеров



— это самый злой минимум для отчаянных.

Данные собирают как могут. Кто-то вручную — получается дольше, меньше, зато без ошибок. Кто-то автоматически — просто сливает машине всё, что нашлось, и верит в лучшее. Самые хитрые, типа гугла, используют своих же

пользователей для бесплатной разметки. Вспомните ReCaptcha, которая иногда требует «найти на фотографии все дорожные знаки» — это оно и есть.

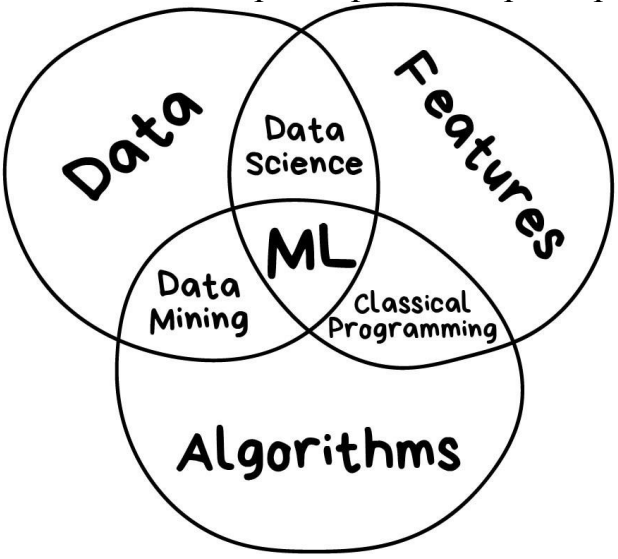
Признаки

Называют их фичами (features). Фичи, свойства, характеристики, признаки — ими могут быть пробег автомобиля, пол пользователя, цена акций, даже счетчик частоты появления слова в тексте может быть фичей.

Машина должна знать, на что ей конкретно смотреть. Хорошо, когда данные просто лежат в табличках — названия их колонок и есть фичи. А если у нас сто гигабайт картинок с котами? Когда признаков много, модель работает медленно и неэффективно. Зачастую отбор правильных фич занимает больше времени, чем всё остальное обучение. Но иногда человек сам занимается разметкой данных и приданием им признаков, хотя получается это не всегда корректно.

Алгоритм

Одну задачу можно решить разными методами примерно всегда. От выбора метода зависит точность, скорость работы и размер готовой модели.



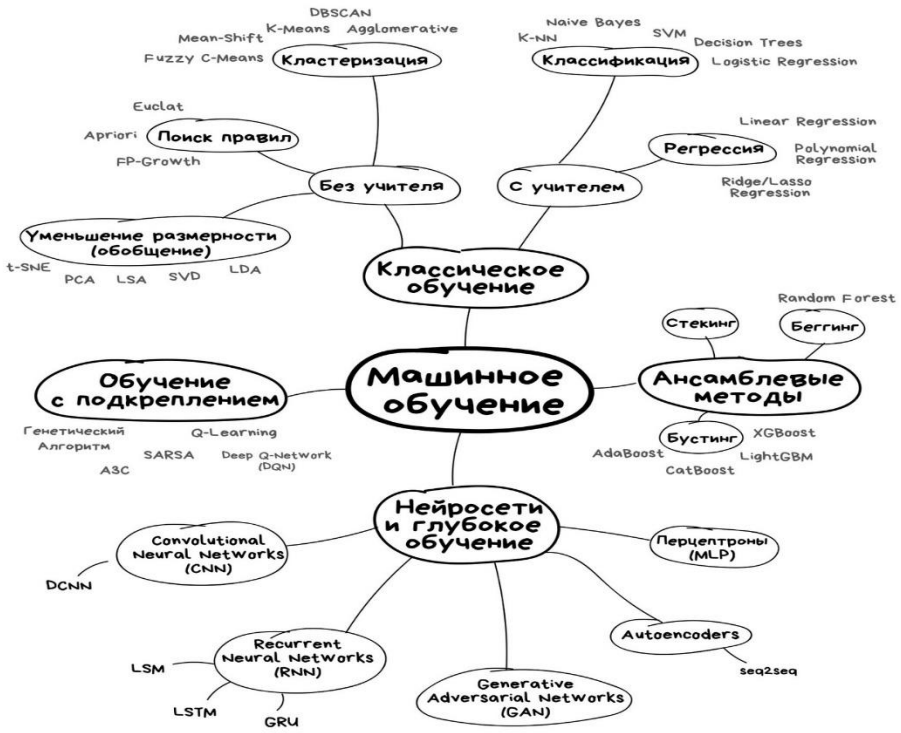
Обучение vs Интеллект

Искусственный интеллект — название всей области, как биология или химия.

Машинное обучение — это раздел искусственного интеллекта. Важный, но не единственный.
Нейросети — один из видов машинного обучения. Популярный, но есть и другие, не хуже.
Глубокое обучение — архитектура нейросетей, один из подходов к их построению и обучению.
Вот что машины сегодня умеют, а что не под силу даже самым обученным.

Машина может	**Машина не может**
Предсказывать	**Создавать новое**
Запоминать	**Резко поумнеть**
Воспроизводить	**Выйти за рамки задачи**
Выбирать лучшее	

Карта мира машинного обучения



Сегодня в машинном обучении есть всего четыре основных направления.

Часть 1. Классическое обучение
Первые алгоритмы пришли к нам из чистой статистики еще в 1950-х. Они решали формальные задачи — искали закономерности в циферках, оценивали близость точек в пространстве и вычисляли направления. Сегодня на классических алгоритмах держится добрая половина интернета. Когда вы встречаете блок «Рекомендованные статьи» на сайте — это почти всегда дело рук одного из этих алгоритмов.

Обучение с учителем

Классическое обучение любят делить на две категории — с учителем и без. Часто можно встретить их английские наименования — Supervised и Unsupervised Learning.
В первом случае у машины есть некий учитель, который говорит ей как правильно. Рассказывает, что на этой картинке кошка, а на этой собака. То есть учитель уже заранее разделил (разметил) все данные на кошек и собак, а машина учится на конкретных примерах.
В обучении без учителя, машине просто предоставляют кучу фотографий животных и говорят «разберись, кто здесь на кого похож». Данные не размечены, у машины нет учителя, и она пытается сама найти любые закономерности. Об этих методах поговорим ниже.
Очевидно, что с учителем машина обучится быстрее и точнее, потому в боевых задачах его используют намного чаще. Эти задачи делятся на два типа: **классификация — предсказание категории объекта, и регрессия — предсказание места на числовой прямой**.
Классификация
«Разделяет объекты по заранее известному признаку. Носки по цветам, документы по языкам, музыку по жанрам»
Сегодня используют для:

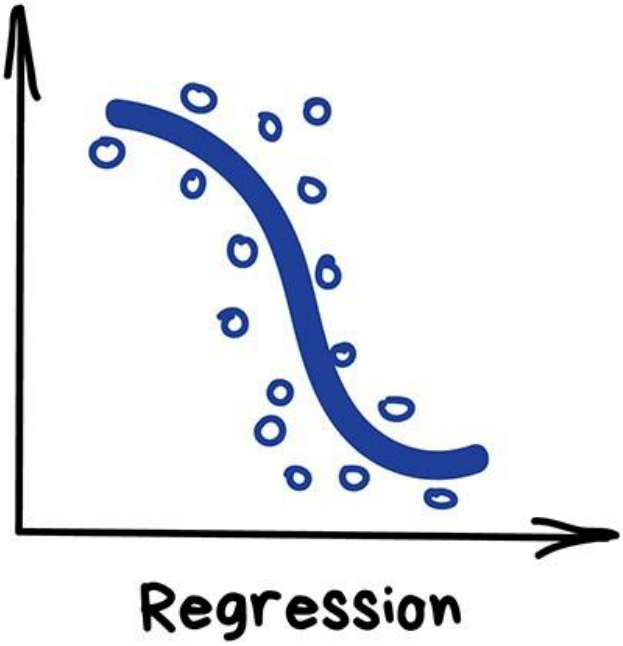
- Спам-фильтры
- Определение языка
- Поиск похожих документов
- Анализ тональности
- Распознавание рукописных букв и цифр
- Определение подозрительных транзакций

Популярные алгоритмы: Наивный Байес, Деревья Решений, Логистическая Регрессия, K-ближайших соседей, Машины Опорных Векторов
Классификация вещей — самая популярная задача во всём машинном обучении. Машина в ней как ребёнок, который учится раскладывать игрушки: роботов в один ящик, танки в другой.
Для классификации всегда нужен учитель — размеченные данные с признаками и категориями, которые машина будет учиться определять по этим признакам. Дальше классифицировать можно что угодно: пользователей по интересам — так делают алгоритмические ленты, статьи по языкам и тематикам — важно для поисковиков, музыку по жанрам — вспомните плейлисты Яндекс.Музыки, даже письма в вашем почтовом ящике.
Раньше все спам-фильтры работали на алгоритме Наивного Байеса. Машина считала сколько раз “плохое" слово встречается в спаме, а сколько раз в нормальных письмах. Перемножала эти вероятности по формуле Байеса и складывала результаты всех
Позже спамеры научились обходить фильтр Байеса, просто вставляя в конец письма много слов с «хорошими» рейтингами. Метод получил ироничное название Отравление Байеса, а фильтровать спам стали другими

алгоритмами. Но метод навсегда остался в учебниках как самый простой, красивый и один из первых практически полезных.
Возьмем другой пример полезной классификации. Вот берёте вы кредит в банке. Как банку удостовериться, вернёте вы его или нет? Точно никак, но у банка есть тысячи профилей других людей, которые уже брали кредит до вас. Там указан их возраст, образование, должность, уровень зарплаты и главное — кто из них вернул кредит, а с кем возникли проблемы.
Для этой задачи придумали Деревья Решений. Машина автоматически разделяет все данные по вопросам, ответы на которые «да» или «нет». Вопросы могут быть не совсем адекватными с точки зрения человека, например *«зарплата заёмщика больше, чем 25934 рубля?»*, но машина придумывает их так, чтобы на каждом шаге разбиение было самым точным. Деревья нашли свою нишу в областях с высокой ответственностью: диагностике, медицине, финансах.

Дерево Решений

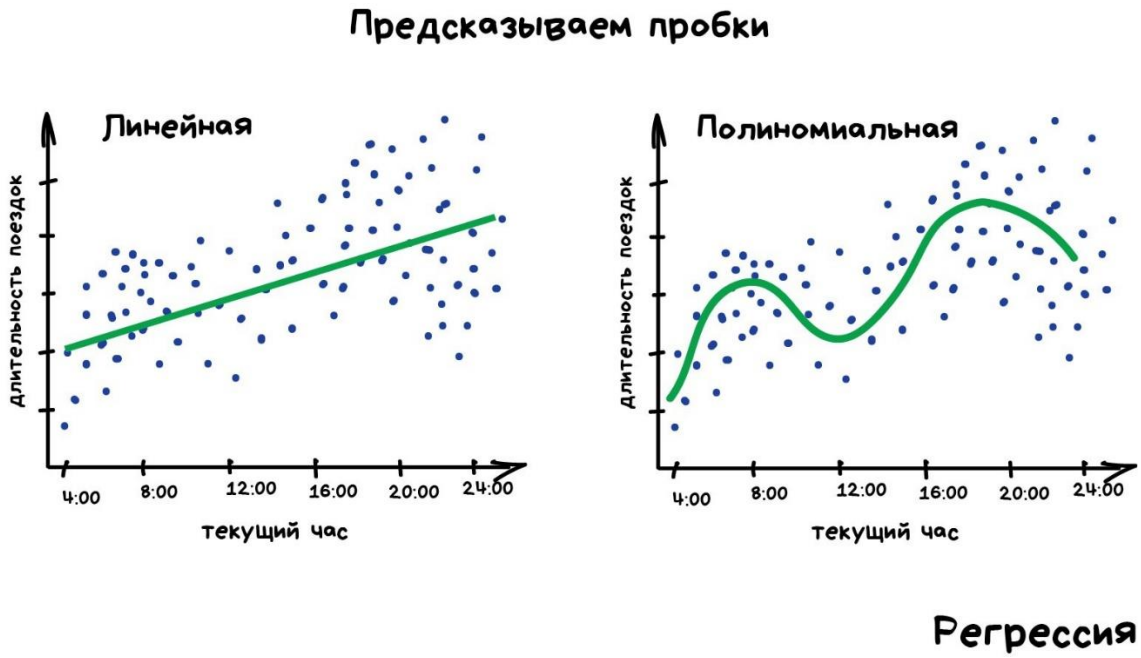
У классификации есть полезная обратная сторона — поиск аномалий. Когда какой-то признак объекта сильно не вписывается в наши классы, мы ярко подсвечиваем его на экране. Сейчас так делают в медицине: компьютер подсвечивает врачу все подозрительные области МРТ или выделяет отклонения в анализах.
Сегодня для классификации всё чаще используют нейросети, ведь по сути их для этого и изобрели.
Правило здесь такое: сложнее данные — сложнее алгоритм. Для текста, цифр, табличек чаще используется классика. Там модели меньше, обучаются быстрее и работают понятнее. Для картинок, видео и тд. — сразу смотрят в сторону нейросетей.
Регрессия



Сегодня используют для:

- Прогноз стоимости ценных бумаг
- Анализ спроса, объема продаж
- Медицинские диагнозы
- Любые зависимости числа от времени

Регрессия — та же классификация, только вместо категории мы предсказываем число. Стоимость автомобиля по его пробегу, количество пробок по времени суток, объем спроса на товар от роста компании и.т.д. На регрессию идеально ложатся любые задачи, где есть зависимость от времени. Регрессию очень любят финансисты и аналитики, она встроена даже в Excel. Внутри всё работает, опять же, банально: машина тупо пытается нарисовать линию, которая в среднем отражает зависимость. Правда, в отличии от человека с фломастером и вайтбордом, делает она это математически точно — считая среднее расстояние до каждой точки и пытаясь всем угодить.



Регрессия

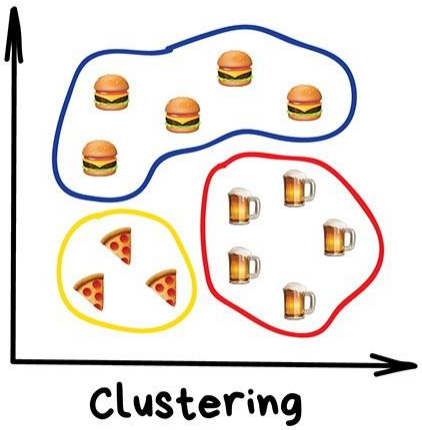
Когда регрессия рисует прямую линию, её называют линейной, когда кривую — полиномиальной.

Обучение без учителя

Обучение без учителя (Unsupervised Learning) было изобретено позже, аж в 90-е, и на практике используется реже. Но бывают задачи, где у нас просто нет выбора.

Обучение без учителя, всё же, чаще используют как метод анализа данных, а не как основной алгоритм.

Кластеризация(кластерный анализ)



«Разделяет объекты по неизвестному признаку. Машина сама решает как лучше»

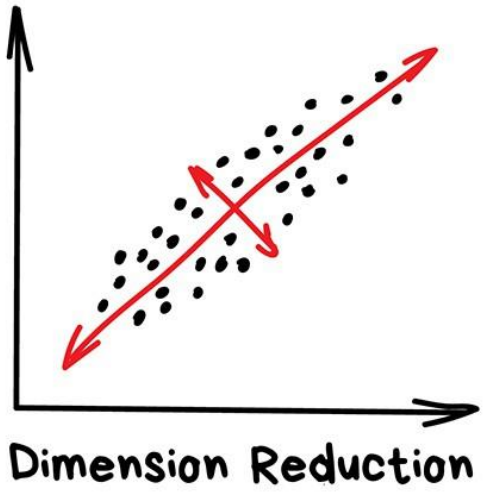
Сегодня используют для:

- Сегментация рынка (типов покупателей, лояльности)
- Объединение близких точек на карте
- Сжатие изображений
- Анализ и разметки новых данных
- Детекторы аномального поведения

Популярные алгоритмы: Метод K-средних, Mean-Shift, DBSCAN

Кластеризация — это классификация, но без заранее известных классов. Она сама ищет похожие объекты и объединяет их в кластеры. Количество кластеров можно задать заранее или доверить это машине
Отличный пример кластеризации — маркеры на картах в вебе
Более сложные примеры кластеризации можно вспомнить в приложениях iPhoto или Google Photos, которые находят лица людей на фотографиях и группируют их в альбомы. Приложение не знает как зовут ваших друзей, но может отличить их по характерным чертам лица. Типичная кластеризация. Правда для начала им приходится найти эти самые «характерные черты», а это уже только с учителем.

Полезная статья по кластеризации: The 5 Clustering Algorithms Data Scientists Need to Know
Как и классификация, кластеризация тоже может использоваться как детектор аномалий. Поведение пользователя после регистрации резко отличается от нормального? Заблокировать его. При этом нам даже не надо знать, что есть «нормальное поведение» — мы просто выгружаем все действия пользователей в модель, и пусть машина сама разбирается кто тут нормальный.
Уменьшение Размерности (Обобщение)

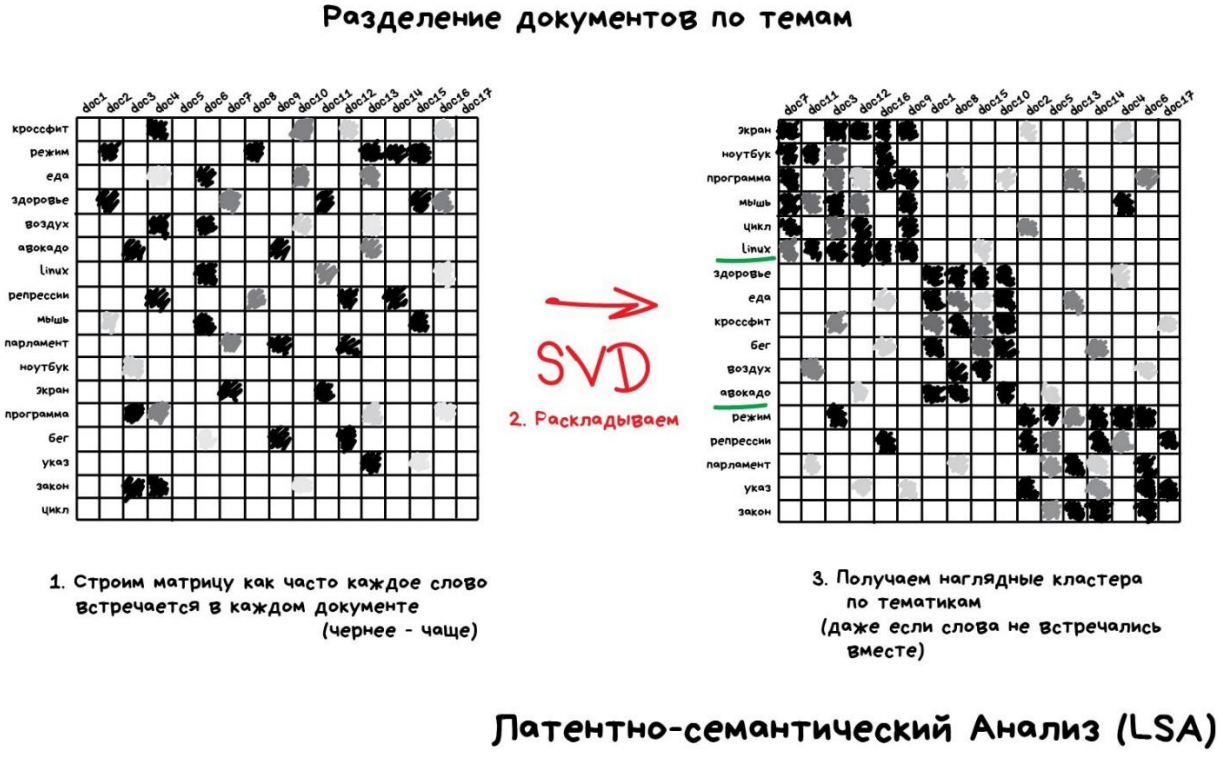


«Собирает конкретные признаки в абстракции более высокого уровня»
Сегодня используют для:

- Рекомендательные Системы (★)
- Красивые визуализации
- Определение тематики и поиска похожих документов
- Анализ фейковых изображений
- Риск-менеджмент

Для нас практическая польза их методов в том, что мы можем объединить несколько признаков в один и получить абстракцию. Например, собаки с

треугольными ушами, длинными носами и большими хвостами соединяются в полезную абстракцию «овчарки». Да, мы теряем информацию о конкретных овчарках, но новая абстракция всяко полезнее этих лишних деталей. Плюс, обучение на меньшем количестве размерностей идёт сильно быстрее.
Инструмент на удивление хорошо подошел для определения тематик текстов (Topic Modelling). Мы смогли абстрагироваться от конкретных слов до уровня смыслов даже без привлечения учителя со списком категорий.
Алгоритм назвали Латентно-семантический анализ (LSA), и его идея была в том, что частота появления слова в тексте зависит от его тематики: в научных статьях больше технических терминов, в новостях о политике — имён политиков. Да, мы могли бы просто взять все слова из статей и кластеризовать, но тогда мы бы потеряли все полезные связи между словами, например, что *батарейка* и *аккумулятор*, означают одно и то же в разных документах.
Точность такой системы очень мала.



Полезные статьи
Как уменьшить количество измерений и извлечь из этого пользу
Алгоритм LSA для поиска похожих документов.

Поиск правил (ассоциация)
«Ищет закономерности в потоке заказов»
Сегодня используют для:

- Прогноз акций и распродаж
- Анализ товаров, покупаемых вместе
- Расстановка товаров на полках
- Анализ паттернов поведения на веб-сайтах

Сюда входят все методы анализа продуктовых корзин, стратегий маркетинга и других последовательностей.
Предположим, покупатель берёт в дальнем углу магазина лимонад и идёт на кассу. Стоит ли ставить на его пути чипсы? Часто ли люди берут их вместе? Наверное да, но какие ещё товары покупают вместе? Когда вы владелец сети гипермаркетов, ответ для вас не всегда очевиден, но одно тактическое улучшение в расстановке товаров может принести хорошую прибыль.
То же касается интернет-магазинов, где задача еще интереснее — за каким товаром покупатель вернётся в следующий раз?

Часть 2. Обучение с подкреплением

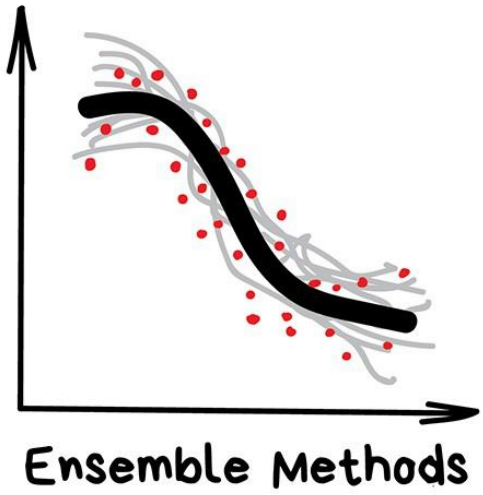
«Брось робота в лабиринт и пусть ищет выход»
Сегодня используют для:

- Самоуправляемых автомобилей
- Роботов пылесосов
- Игр
- Автоматической торговли
- Управления ресурсами предприятий

Обучение с подкреплением используют там, где задачей стоит не анализ данных, а выживание в реальной среде.
Средой может быть даже видеоигра. Роботы, играющие в Марио, были популярны еще лет пять назад. Средой может быть реальный мир. Как пример — автопилот Теслы, который учится не сбивать пешеходов, или роботы-пылесосы.
Знания об окружающем мире такому роботу могут быть полезны, но чисто для справки. Не важно сколько данных он соберёт, у него всё равно не получится предусмотреть все ситуации. Потому его цель
— **минимизировать ошибки, а не рассчитать все ходы**. Робот учится выживать в пространстве с максимальной выгодой: собранными монетками в Марио или временем поездки в Тесле. Выживание в среде и есть идея обучения с подкреплением
Умные модели роботов-пылесосов и самоуправляемые автомобили обучаются именно так: им создают виртуальный город (часто на основе карт настоящих городов), населяют случайными пешеходами и отправляют учиться никого там не убивать. Когда робот начинает хорошо себя

чувствовать в искусственном GTA, его выпускают тестировать на реальные улицы.
Но **машина не запоминает каждое движение, а пытается обобщить ситуации, чтобы выходить из них с максимальной выгодой.**
Машина прогоняет миллионы симуляций в среде, запоминая все сложившиеся ситуации и выходы из них, которые принесли максимальное вознаграждение. Но как понять, когда у нас сложилась известная ситуация, а когда абсолютно новая? Вот самоуправляемый автомобиль стоит у перекрестка и загорается зелёный — значит можно ехать? А если справа мчит скорая помощь с мигалками?
Исследователи постоянно этим занимаются, изобретая свои костыли. Одни прописывают все ситуации руками, что позволяет им обрабатывать исключительные случаи типа проблемы вагонетки. Другие идут глубже и отдают эту работу нейросетям, пусть сами всё найдут. Reinforcement Learning для простого обывателя выглядит как *настоящий интеллект*.

Часть 3. Ансамбли



Сегодня используют для:

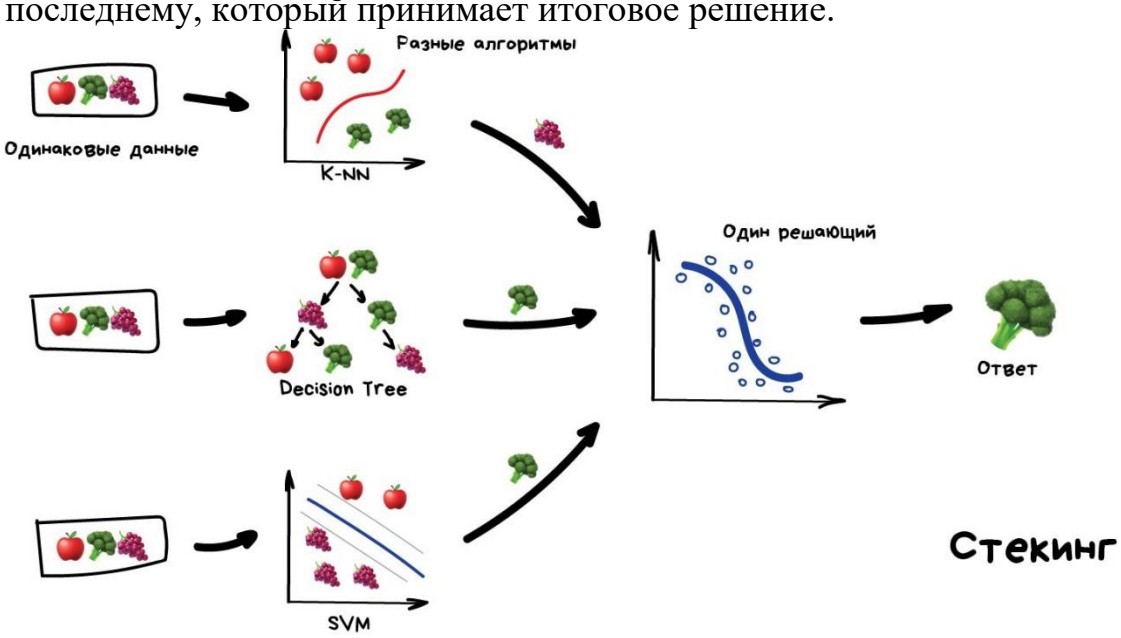
- Всего, где подходят классические алгоритмы (но работают точнее)
- Поисковые системы (★)
- Компьютерное зрение
- Распознавание объектов

Сегодня ансамбли и нейросети дают самые точные результаты и используются всеми крупными компаниями в продакшене.
При всей их эффективности, идея до издевательства проста. Оказывается, если взять несколько не очень эффективных методов обучения и обучить исправлять ошибки друг друга, качество такой системы будет аж сильно выше, чем каждого из методов по отдельности.

Есть три проверенных способа делать ансамбли.

Стекинг

Обучаем несколько разных алгоритмов и передаём их результаты на вход последнему, который принимает итоговое решение.

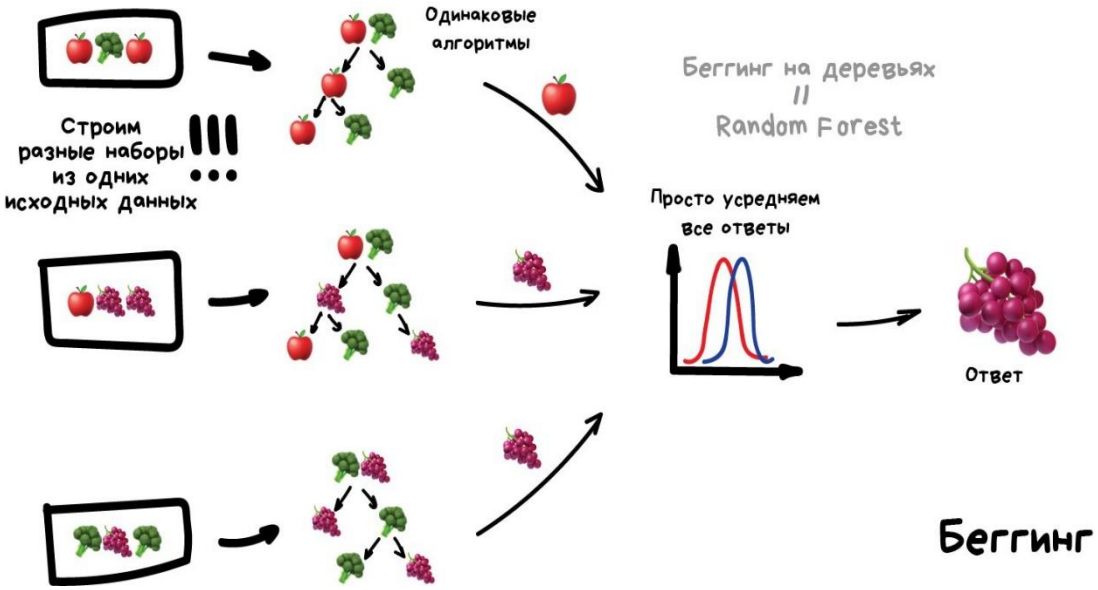


Ключевое слово — разных алгоритмов, ведь один и тот же алгоритм, обученный на одних и тех же данных не имеет смысла. Каких — ваше дело, разве что в качестве решающего алгоритма чаще берут регрессию.

Чисто из опыта — стекинг на практике применяется редко, потому что два других метода обычно точнее.

Беггинг

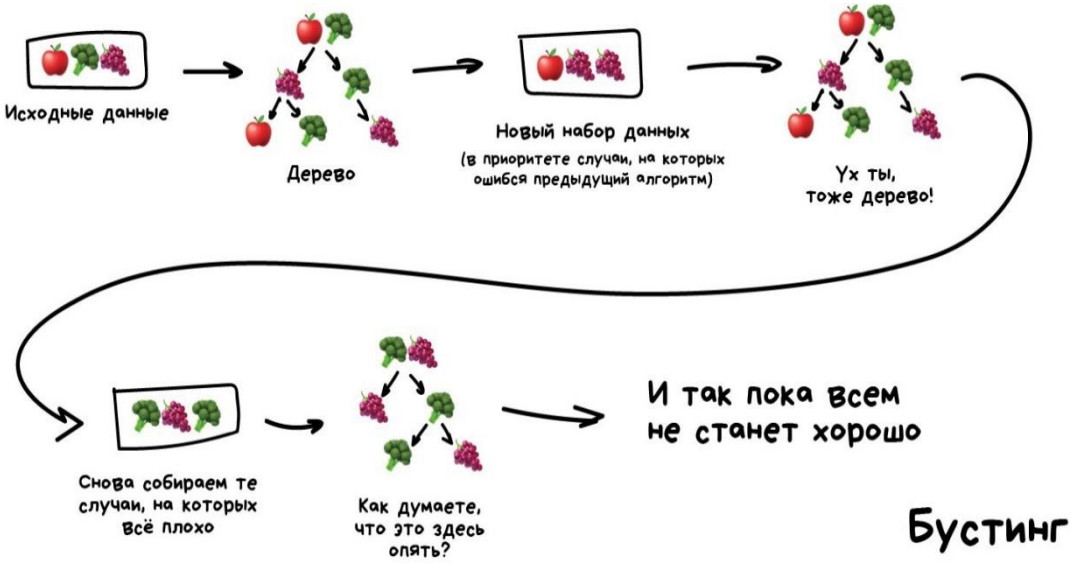
Он же Bootstrap AGGregatING. Обучаем один алгоритм много раз на случайных выборках из исходных данных. В самом конце усредняем ответы. Данные в случайных выборках могут повторяться. То есть из набора 1-2-3 мы можем делать выборки 2-2-3, 1-2-2, 3-1-2 и так пока не надоест. На них мы обучаем один и тот же алгоритм несколько раз, а в конце вычисляем ответ простым голосованием.



Самый популярный пример беггинга — алгоритм Random Forest, беггинг на деревьях, который и нарисован на картинке. Когда вы открываете камеру на телефоне и видите как она очертила лица людей в кадре желтыми прямоугольниками — скорее всего это их работа. Нейросеть будет слишком медлительна в реальном времени, а беггинг идеален, ведь он может считать свои деревья параллельно на всех шейдерах видеокарты.
Дикая способность параллелиться даёт беггингу преимущество даже над следующим методом, который работает точнее, но только в один поток.

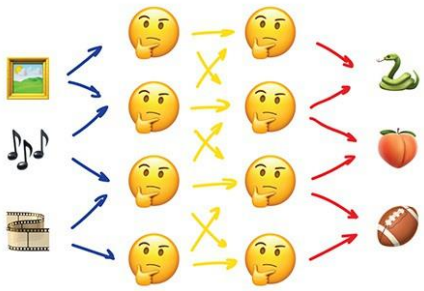
Бустинг

Обучаем алгоритмы последовательно, каждый следующий уделяет особое внимание тем случаям, на которых ошибся предыдущий.
Как в беггинге, мы делаем выборки из исходных данных, но теперь не совсем случайно. В каждую новую выборку мы берём часть тех данных, на которых предыдущий алгоритм отработал неправильно. То есть как бы доучиваем новый алгоритм на ошибках предыдущего.



Плюсы — неистовая, даже нелегальная в некоторых странах, точность классификации, которой позавидуют все бабушки у подъезда. Минусы уже названы — не параллелится. Хотя всё равно работает быстрее нейросетей, которые как гружёные камазы с песком по сравнению с шустрым бустингом. Нужен реальный пример работы бустинга — откройте Яндекс и введите запрос. Слышите, как Матрикснет грохочет деревьями и ранжирует вам результаты? Вот это как раз оно, Яндекс сейчас весь на бустинге.
Сегодня есть три популярных метода бустинга, отличия которых хорошо донесены в статье CatBoost vs. LightGBM vs. XGBoost

Часть 4. Нейросети и глубокое обучение

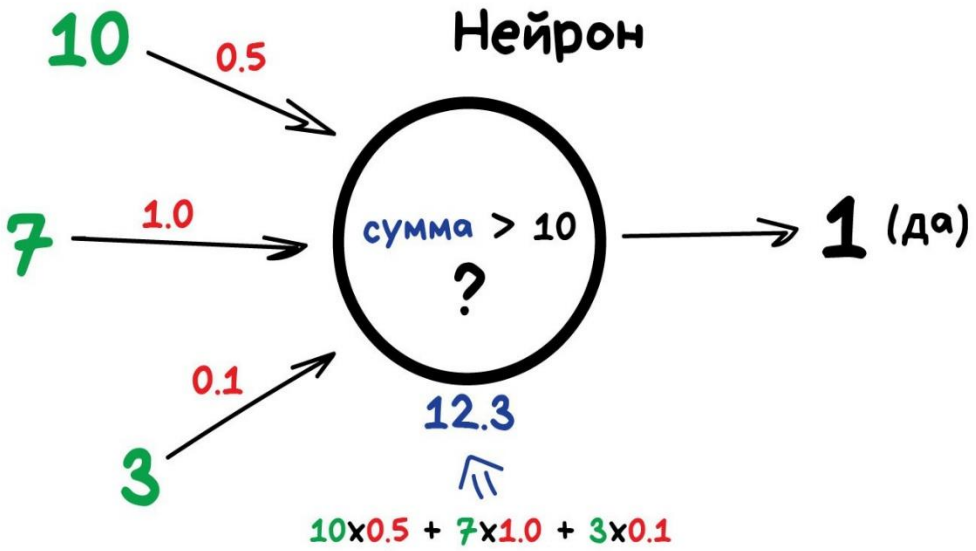


Сегодня используют для:

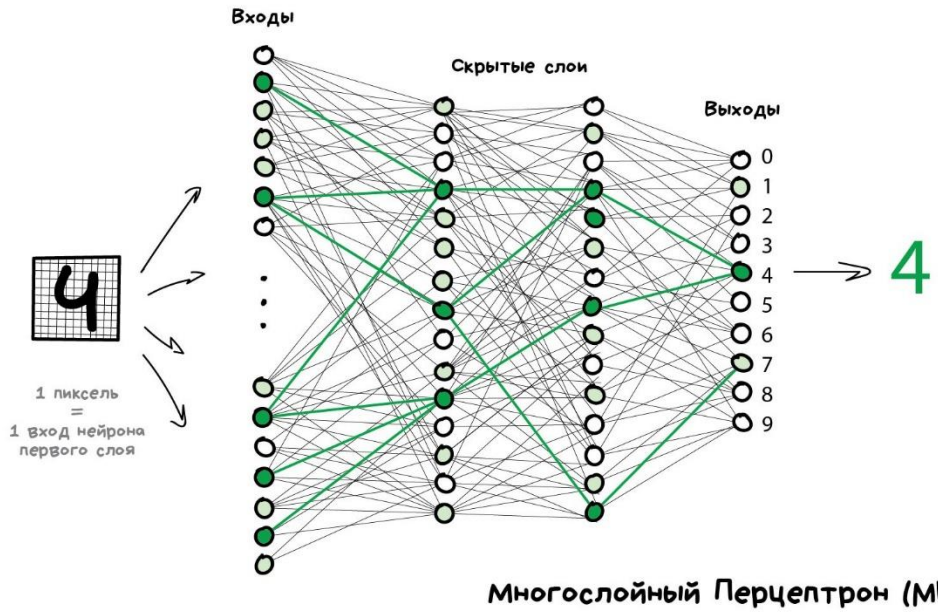
- Вместо всех вышеперечисленных алгоритмов вообще
- Определение объектов на фото и видео
- Распознавание и синтез речи
- Обработка изображений, перенос стиля
- Машинный перевод

Популярные архитектуры: Перцептрон, Свёрточные Сети (CNN), Рекуррентные Сети(RNN), Автоэнкодеры
Любая нейросеть — это набор нейронов и связей между ними. Нейрон лучше всего представлять просто как функцию с кучей входов и одним выходом.

Задача нейрона — взять числа со своих входов, выполнить над ними функцию и отдать результат на выход. Простой пример полезного нейрона: просуммировать все цифры со входов, и если их сумма больше N — выдать на выход единицу, иначе — ноль.
Связи — это каналы, через которые нейроны шлют друг другу циферки. У каждой связи есть свой вес — её единственный параметр, который можно условно представить как прочность связи. Когда через связь с весом 0.5 проходит число 10, оно превращается в 5. Сам нейрон не разбирается, что к нему пришло и суммирует всё подряд — вот веса и нужны, чтобы управлять на какие входы нейрон должен реагировать, а на какие нет.



Чтобы сеть не превратилась в анархию, нейроны решили связывать не как захочется, а по слоям. Внутри одного слоя нейроны никак не связаны, но соединены с нейронами следующего и предыдущего слоя. Данные в такой сети идут строго в одном направлении — от входов первого слоя к выходам последнего.
Если сделать достаточное количество слоёв и правильно расставить веса в такой сети, получается следующее — подав на вход, скажем, изображение написанной от руки цифры 4, чёрные пиксели активируют связанные с ними нейроны, те активируют следующие слои, и так далее и далее, пока в итоге не загорится самый выход, отвечающий за четвёрку. Результат достигнут.



В реальном программировании, естественно, никаких нейронов и связей не пишут, всё представляют матрицами и считают матричными произведениями, потому что нужна скорость. Такая сеть, где несколько слоёв и между ними связаны все нейроны, называется перцептроном (MLP) и считается самой простой архитектурой для новичков.

Когда мы построили сеть, наша задача правильно расставить веса, чтобы нейроны реагировали на нужные сигналы. Тут нужно вспомнить, что у нас же есть данные — примеры «входов» и правильных «выходов». Будем показывать нейросети рисунок той же цифры 4 и говорить «подстрой свои веса так, чтобы на твоём выходе при таком входе всегда загоралась четвёрка».

Сначала все веса просто расставлены случайно, мы показываем сети цифру, она выдаёт какой-то случайный ответ (весов-то нет), а мы сравниваем, насколько результат отличается от нужного нам. Затем идём по сети в обратном направлении, от выходов ко входам, и говорим каждому нейрону — так, ты вот тут зачем-то активировался, из-за тебя всё пошло не так, давай ты будешь чуть меньше реагировать на вот эту связь и чуть больше на вон ту, ок?

Через тысяч сто таких циклов «прогнали-проверили-наказали» есть надежда, что веса в сети откорректируются так, как мы хотели. Научно этот подход называется Backpropagation или «Метод обратного распространения ошибки». Забавно то, что чтобы открыть этот метод понадобилось двадцать лет. До него нейросети обучали как могли.

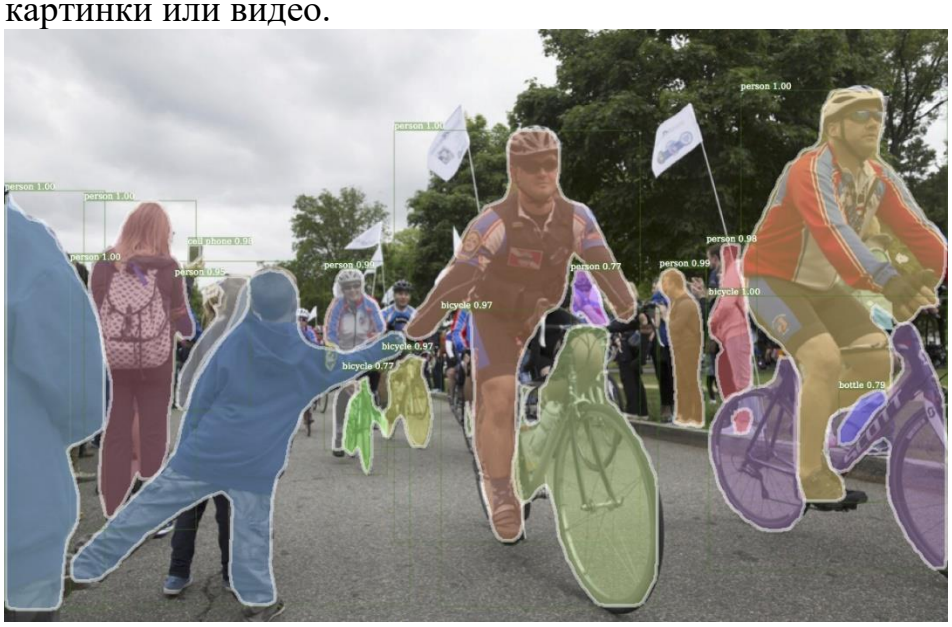
Хорошо обученная нейросеть могла притворяться любым алгоритмом из этой статьи, а зачастую даже работать точнее. Такая универсальность сделала их дико популярными

Отличие глубокого обучения от классических нейросетей было в новых методах обучения, которые справлялись с большими размерами сетей.

Однако сегодня лишь теоретики разделяют, какое обучение можно считать глубоким, а какое не очень

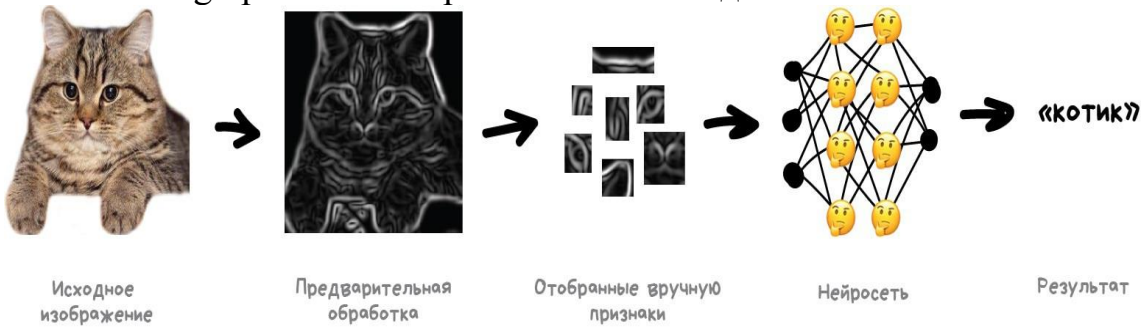
Свёрточные Нейросети (CNN)

Свёрточные сети сейчас на пике популярности. Они используются для поиска объектов на фото и видео, распознавания лиц, переноса стиля, генерации и дорисовки изображений, создания эффектов типа слоу-мо и улучшения качества фотографий. Сегодня CNN применяют везде, где есть картинки или видео.



Картинка выше — результат работы библиотеки Detectron.

Проблема с изображениями всегда была в том, что непонятно, как выделять на них признаки. Текст можно разбить по предложениям, взять свойства слов из словарей. Картинки же приходилось размечать руками, объясняя машине, где у котика на фотографии ушки, а где хвост. Такой подход даже назвали «handcrafting признаков» и раньше все так и делали.



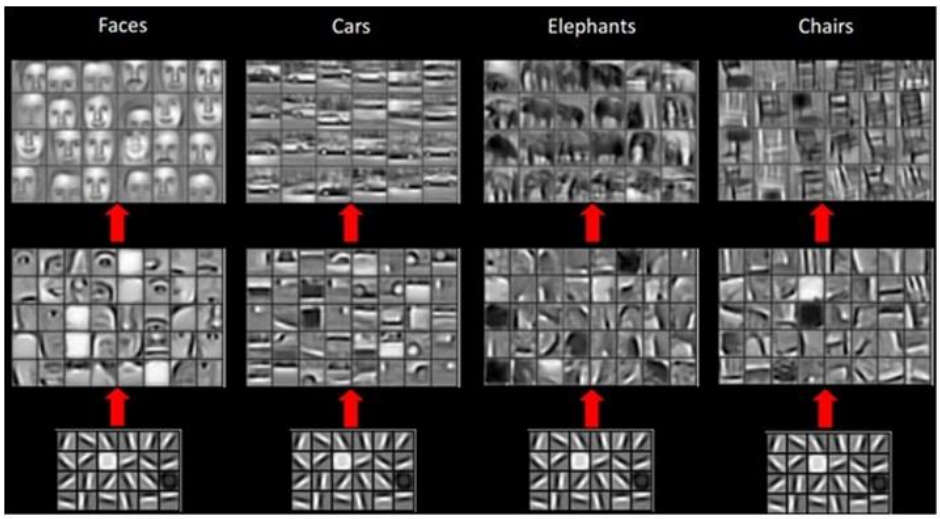
Проблем у ручного крафтинга много.

Во-первых, если котик на фотографии прижал ушки или отвернулся — всё, нейросеть ничего не увидит.

Во-вторых, попробуйте сами сейчас назвать хотя бы десять характерных признаков, отличающих котиков от других животных. Человек не смотрит только на форму ушей и количество лап — он оценивает объект по куче разных признаков, о которых сам даже не задумывается. А значит, не понимает и не может объяснить машине.

Получается, машине надо самой учиться искать эти признаки, составляя из каких-то базовых линий. Будем делать так: для начала разделим изображение на блоки 8x8 пикселей и выберем какая линия доминирует в каждом — горизонтальная [-], вертикальная [|] или одна из диагональных [/]. Могут и две, и три, так тоже бывает, мы не всегда точно уверены.
На выходе мы получим несколько массивов палочек, которые по сути являются простейшими признаками наличия очертаний объектов на картинке. По сути это тоже картинки, просто из палочек. Значит мы можем вновь выбрать блок 8x8 и посмотреть уже, как эти палочки сочетаются друг с другом. А потом еще и еще.
Такая операция называется свёрткой, откуда и пошло название метода. Свёртку можно представить как слой нейросети, ведь нейрон — абсолютно любая функция.

Когда мы прогоняем через нашу нейросеть кучу фотографий котов, она автоматически расставляет большие веса тем сочетаниям из палочек, которые увидела чаще всего. Причём неважно, это прямая линия спины или сложный геометрический объект типа мордочки — что-то обязательно будет ярко активироваться.
На выходе же мы поставим простой перцептрон, который будет смотреть какие сочетания активировались и говорить кому они больше характерны — кошке или собаке.



Красота идеи в том, что у нас получилась нейросеть, которая сама находит характерные признаки объектов. Нам больше не надо отбирать их руками. Мы можем сколько угодно кормить её изображениями любых объектов, просто нагуглив миллион картинок с ними — сеть сама составит карты признаков из палочек и научится определять что угодно.

Рекуррентные Нейросети (RNN)

Вторая по популярности архитектура на сегодняшний день. Благодаря рекуррентным сетям у нас есть такие полезные вещи, как машинный перевод текстов и компьютерный синтез речи. На них решают все задачи, связанные с последовательностями — голосовые, текстовые или музыкальные.

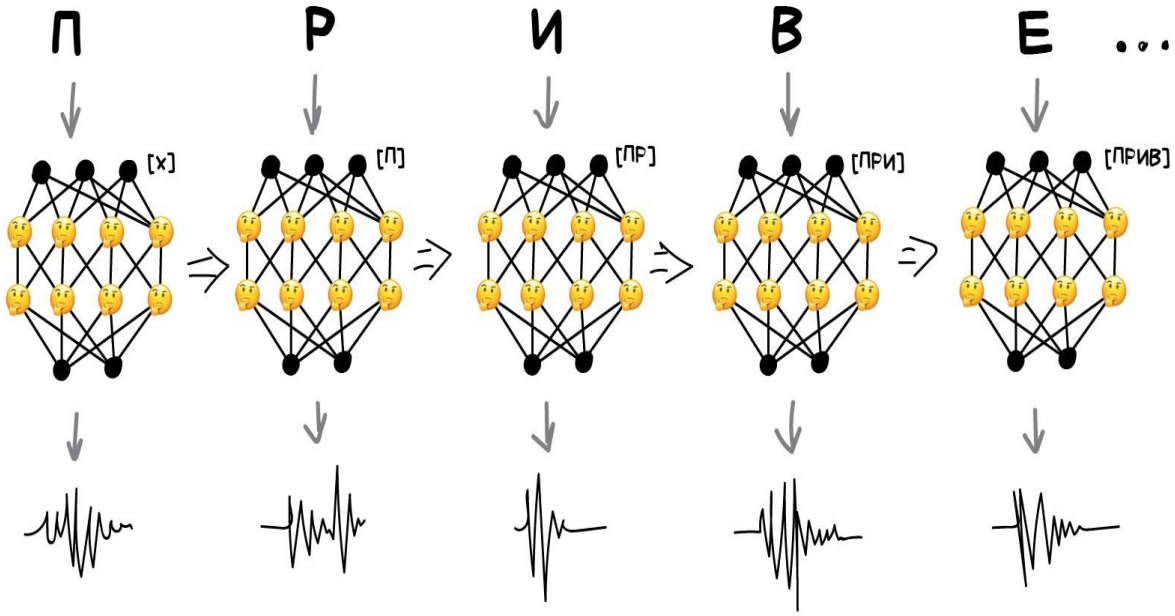
Помните старые голосовые синтезаторы типа Microsoft Sam из Windows XP, который смешно произносил слова по буквам, пытаясь как-то склеить их между собой? А теперь посмотрите на Amazon Alexa или Алису от Яндекса — они сегодня не просто произносят слова без ошибок, они даже расставляют акценты в предложении!

Потому что современные голосовые помощники обучают говорить не буквами, а фразами. Но сразу заставить нейросеть целиком выдавать фразы не выйдет, ведь тогда ей надо будет запомнить все фразы в языке и её размер будет исполинским. Тут на помощь приходит то, что текст, речь или музыка — это последовательности. Каждое слово или звук — как бы самостоятельная единица, но которая зависит от предыдущих.

Достаточно легко обучить сеть произносить отдельные слова или буквы. Берём кучу размеченных на слова аудиофайлов и обучаем по входному слову выдавать нам последовательность сигналов, похожих на его произношение. Сравниваем с оригиналом от диктора и пытаемся максимально приблизиться к идеалу. Для такого подойдёт даже перцептрон.

Вот только с последовательностью опять беда, ведь перцептрон не запоминает что он генерировал ранее. Для него каждый запуск как в первый раз. Появилась идея добавить к каждому нейрону память. Так были придуманы рекуррентные сети, в которых каждый нейрон запоминал все свои предыдущие ответы и при следующем запуске использовал их как

дополнительный вход.



Рекуррентная Нейросеть (RNN)

Была лишь одна проблема — когда каждый нейрон запоминал все прошлые результаты, в сети образовалось такое дикое количество входов, что обучить такое количество связей становилось нереально.

Когда нейросеть не умеет забывать — её нельзя обучить.

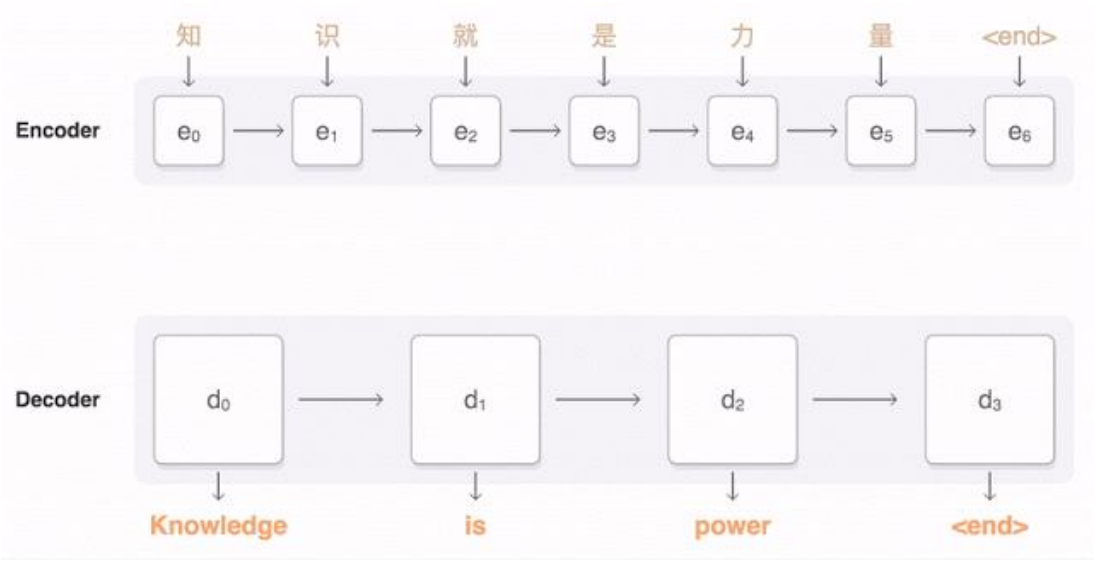
Сначала проблему решили в лоб — обрубили каждому нейрону память. Но потом придумали в качестве этой «памяти» использовать специальные ячейки, похожие на память компьютера или регистры процессора. Каждая ячейка позволяла записать в себя циферку, прочитать или сбросить — их назвали ячейки долгой и краткосрочной памяти (LSTM).

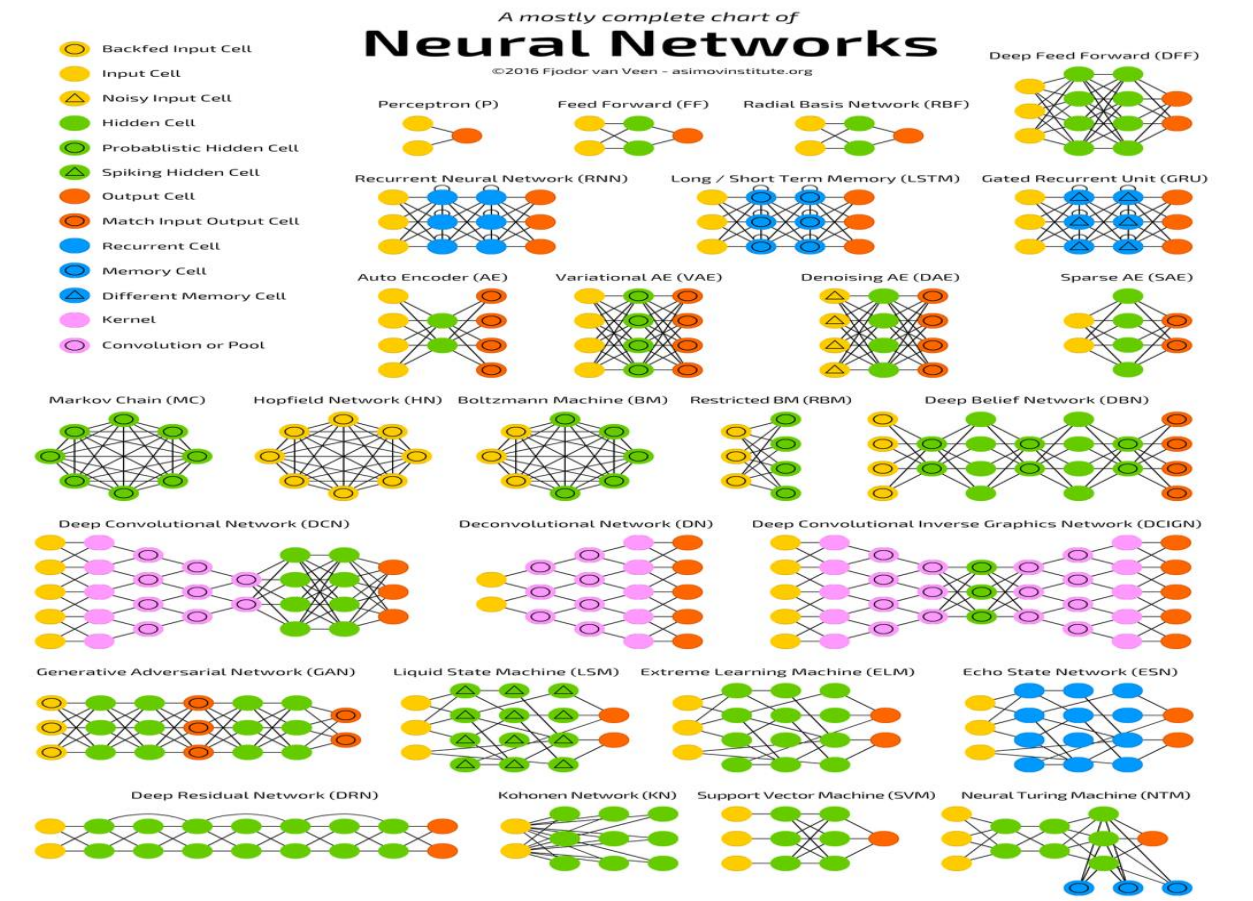
Когда нейрону было нужно поставить себе напоминалку на будущее — он писал это в ячейку, когда наоборот вся история становилась ненужной (предложение, например, закончилось) — ячейки сбрасывались, оставляя только «долгосрочные» связи, как в классическом перцептроне. Другими словами, сеть обучалась не только устанавливать текущие связи, но и ставить напоминалки.

Просто, но работает!

CNN + RNN = фейковый Обама

Озвученные тексты для обучения начали брать откуда угодно. Даже базфид смог выгрузить видеозаписи выступлений Обамы и весьма неплохо научить нейросеть разговаривать его голосом. На этом примере видно, что имитировать голос — достаточно простая задача для сегодняшних машин. С видео посложнее, но это пока.





II.Лабораторная работа

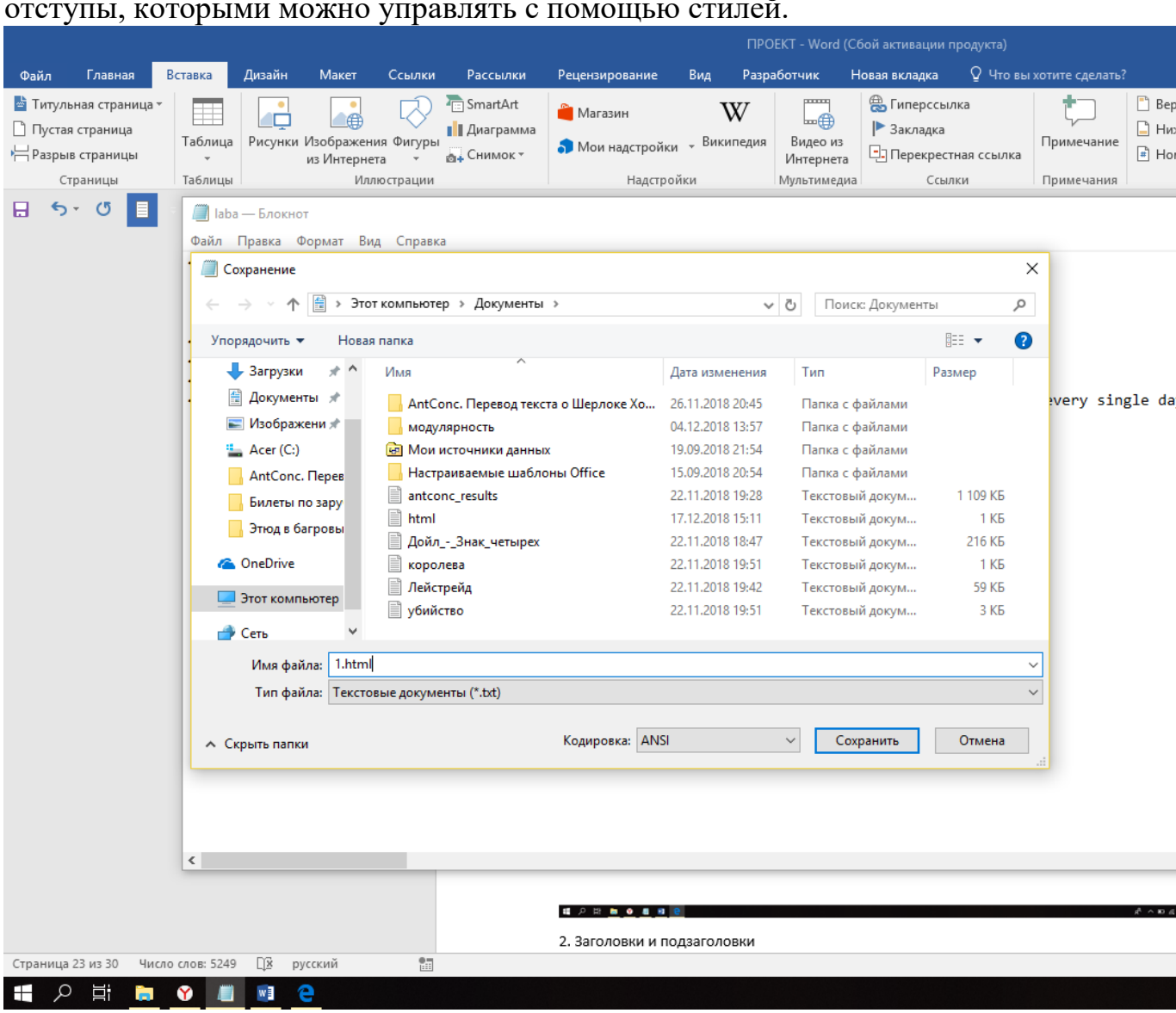
Так как важной составляющей обучения с учителем является обучающая разметка, сегодня ученики попробуют разметить текст в html.

Задание нужно выполнить в Блокноте, используя английский язык. Каждый документ нужно сохранить в формате html. Выбираем “сохранить как” и в названии пишем: ”1/2/3.html.”

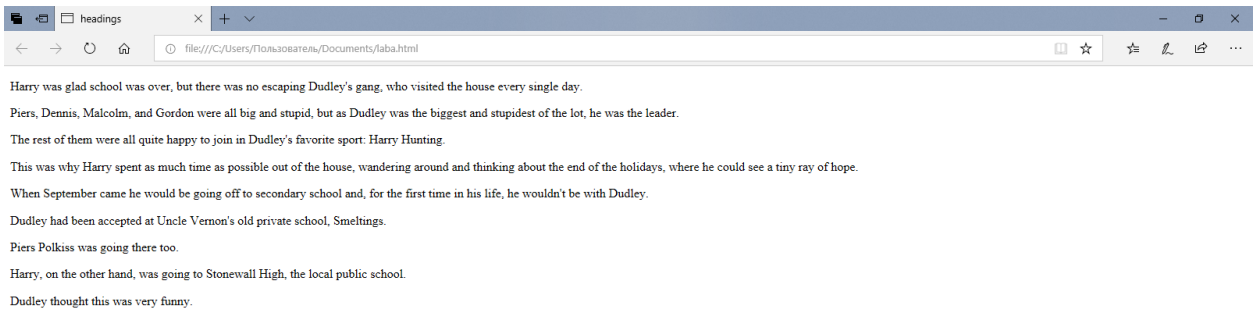
1. Абзацы

В курсе «Структура HTML-документа» вы познакомились с тегами,

необходимыми для создания простейшей HTML-страницы, и с некоторыми служебными тегами, которые не отображаются в браузере.
Начнём с простейшего тега <p>, с помощью которого создаются абзацы. По умолчанию абзацы начинаются с новой строки и имеют вертикальные отступы, которыми можно управлять с помощью стилей.



Разбейте сплошной текст:” Harry was glad school was over, but there was no escaping Dudley's gang, who visited the house every single day. Piers, Dennis, Malcolm, and Gordon were all big and stupid, but as Dudley was the biggest and stupidest of the lot, he was the leader. The rest of them were all quite happy to join in Dudley's favorite sport: Harry Hunting. This was why Harry spent as much time as possible out of the house, wandering around and thinking about the end of the holidays, where he could see a tiny ray of hope. When September came he would be going off to secondary school and, for the first time in his life, he wouldn't be with Dudley. Dudley had been accepted at Uncle Vernon's old private school, Smeltings. Piers Polkiss was going there too. Harry, on the other hand, was going to Stonewall High, the local public school. Dudley thought this was very funny. ” на абзацы (каждое предложение – новый абзац).



2. Заголовки и подзаголовки

Для создания структуры больших текстов обычно используются заголовки. В текстовых редакторах есть возможность выделить часть текста, найти пункт «Заголовок» нужного уровня в меню, и применить его.

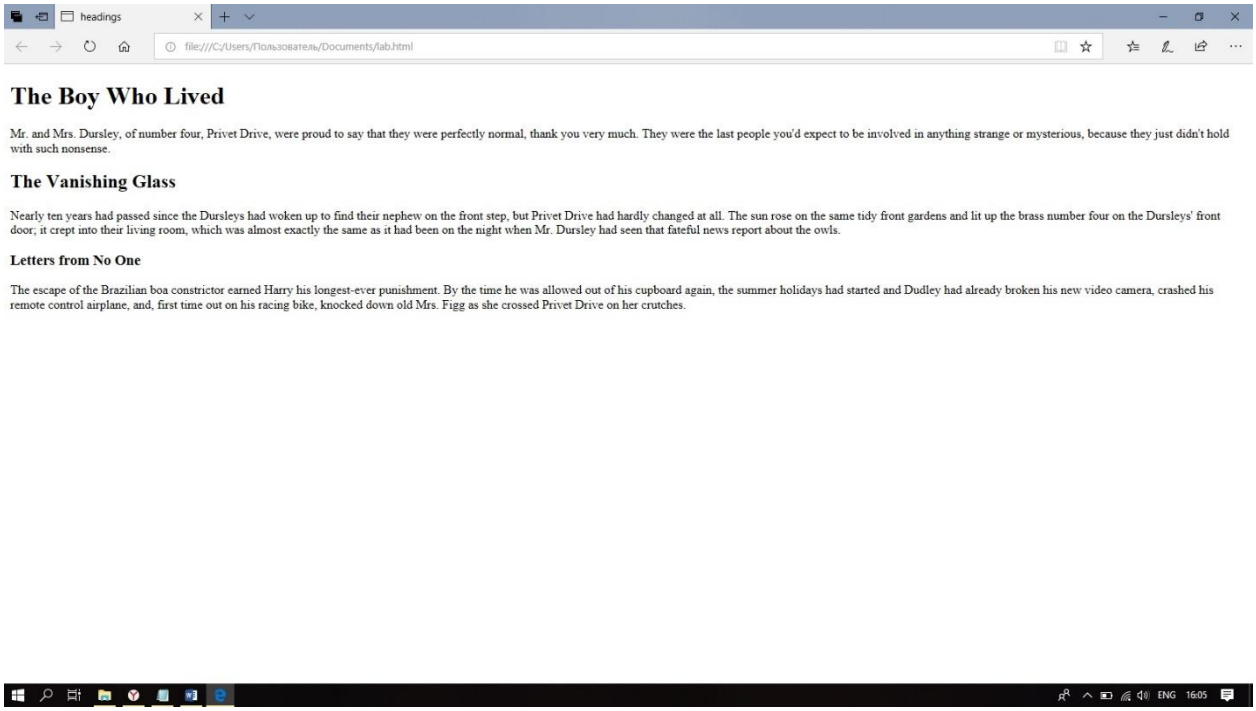
В языке HTML для выделения заголовков предусмотрено целое семейство тегов: от <h1> до <h6>. Тег <h1> обозначает самый важный заголовок (заголовок верхнего уровня), а тег <h6> обозначает подзаголовок самого нижнего уровня.

На практике редко встречаются тексты, в которых встречаются подзаголовки ниже третьего уровня. Поэтому самыми часто используемыми тегами заголовков являются: <h1>, <h2> и <h3>.

Стоит отметить, что поисковые системы придают особое значение заголовкам, поэтому необходимо учиться правильно их использовать. Сделайте по образцу первой главы заголовки 2 и 3 уровня для названий второй и третьей главы.

```
<!DOCTYPE html>
 <html>
  <head>
   <meta charset="utf-8">
<title>headings</title>
</head>
<body>
<h1>The Boy Who Lived</h1>
Mr. and Mrs. Dursley, of number four, Privet Drive, were proud to say that they were perfectly normal, thank you very much. They were the last people you'd
```

expect to be involved in anything strange or mysterious, because they just didn't hold with such nonsense.
The Vanishing Glass
Nearly ten years had passed since the Dursleys had woken up to find their nephew on the front step, but Privet Drive had hardly changed at all. The sun rose on the same tidy front gardens and lit up the brass number four on the Dursleys' front door; it crept into their living room, which was almost exactly the same as it had been on the night when Mr. Dursley had seen that fateful news report about the owls.
Letters from No One
The escape of the Brazilian boa constrictor earned Harry his longest-ever punishment. By the time he was allowed out of his cupboard again, the summer holidays had started and Dudley had already broken his new video camera, crashed his remote control airplane, and, first time out on his racing bike, knocked down old Mrs. Figg as she crossed Privet Drive on her crutches.
</body>
</html>



3.Многоуровневый список
Создать многоуровневый список достаточно просто.

Сначала нужно создать список первого уровня, а затем внутрь любого элемента этого списка, между тегами <li> и , добавить список второго уровня. При этом необходимо аккуратно закрывать все теги.

Пример правильного кода:

1

1.1
1.2

2

Пример кода с ошибкой:

1

1.1
1.2

2

В примере с ошибкой вложенный список вставлен не внутрь элемента списка, а между элементами, что недопустимо.

Количество уровней в списках не ограничено. В многоуровневом списке можно использовать как упорядоченные, так и неупорядоченные списки.

Текст, который должен сожержать файл:
<!DOCTYPE html>
 <html>
  <head>
   <meta charset="utf-8">
<title>multilevel list</title>
</head>
<body>
<h1>любые составляющие списка</h1>

Любые составляющие списка

подпункты
подпункты
подпункты
подпункты

Пункт списка

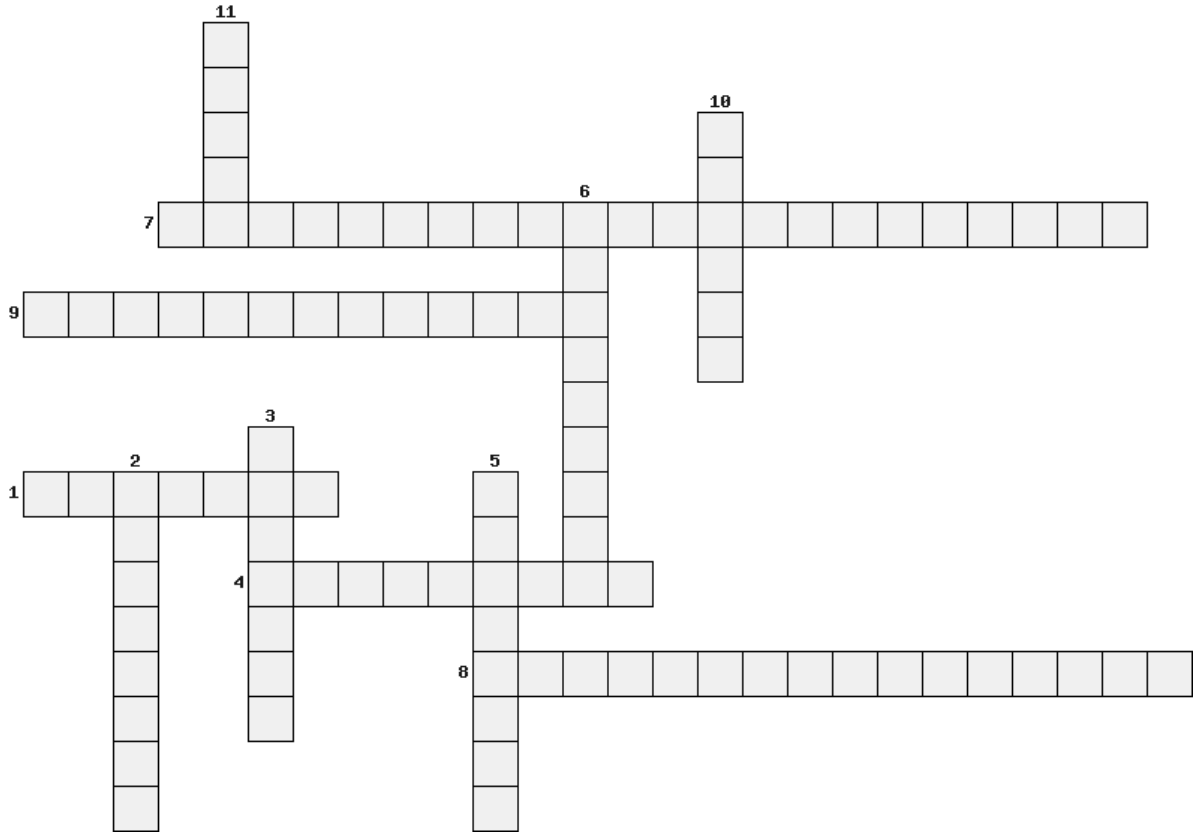

```
</li>
<li>подпункт</li>
  </ol>
    </body>
      </html>
```

Помимо работы с текстом, ученики могут поработать и с картинками. На данном сайте им будут предложены картинки, а они должны помочь машине их разметить, отмечая, что они видят.
Также ученики могут проверить, насколько хорошо нейросети распознавают их рисунки.

III.Задания на закрепление пройденного материала:

1. Разбившись на пары, ученики смогут посоревноваться в усвоении материала тут.

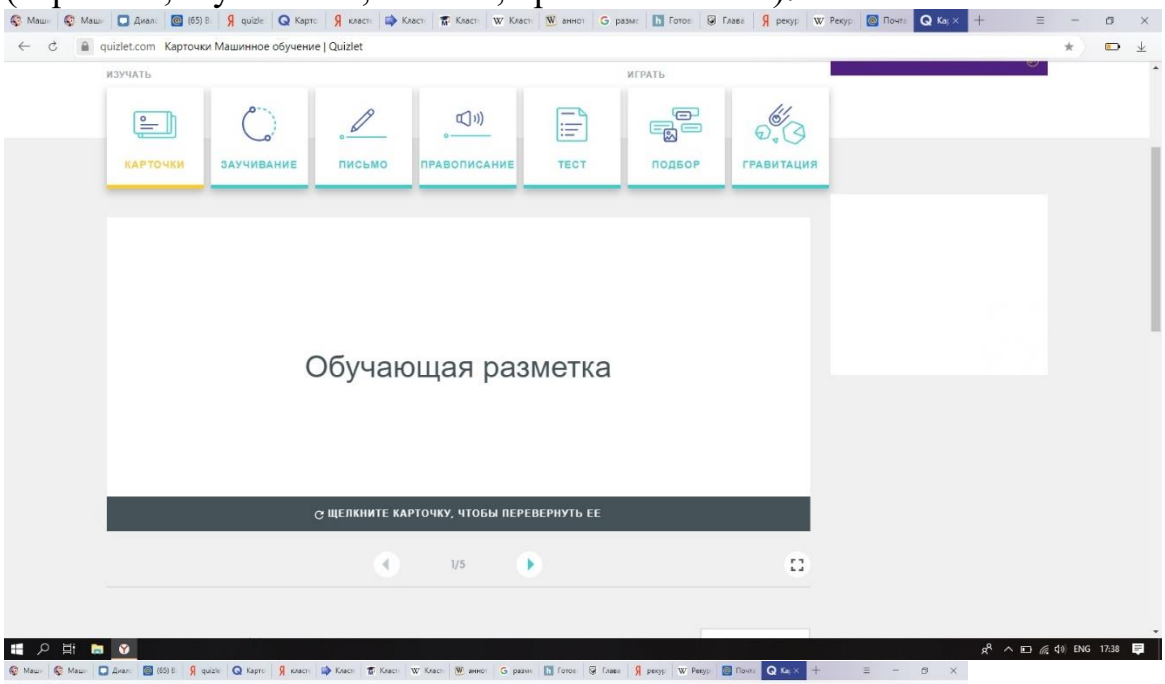
2. Ученикам предлагается решить кроссворд по пройденной теме.

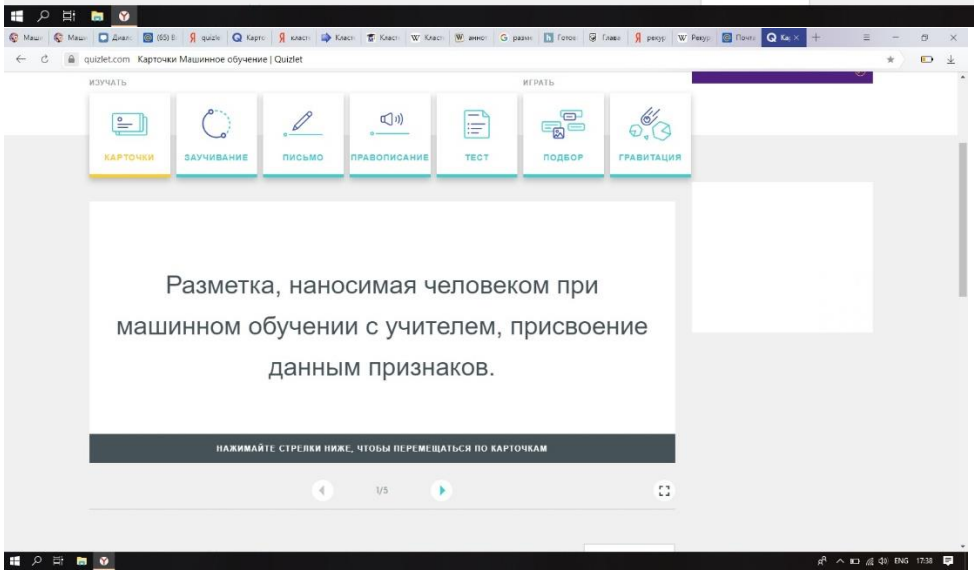


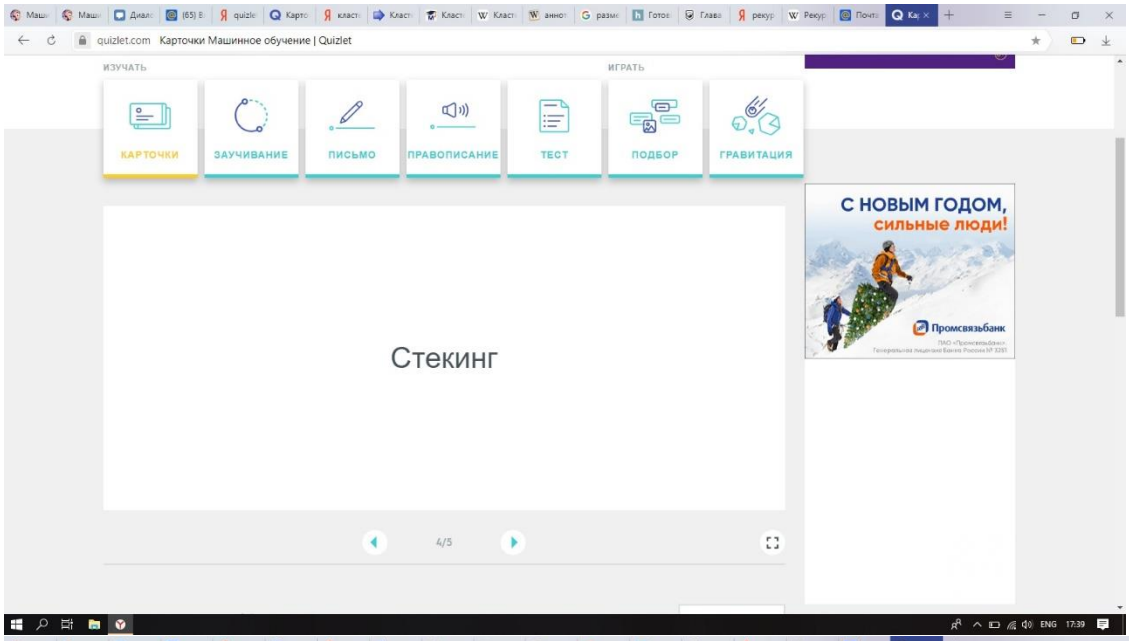
1. Объединение нескольких однородных элементов, которое может рассматриваться как самостоятельная единица, обладающая определёнными свойствами.
2. Набор инструкций, описывающих порядок действий исполнителя для решения некоторой задачи.
3. Мера, позволяющая получить численное значение некоторого свойства программного обеспечения или его спецификаций.
4. Предсказание места на числовой прямой.
5. Несколько методов обучения, исправляющих друг друга.
6. Математическая модель, а также её программное или аппаратное воплощение, построенная по принципу организации и функционирования биологического эквивалента.
7. Наука и технология создания интеллектуальных машин, особенно интеллектуальных компьютерных программ.
8. Класс методов искусственного интеллекта, характерной чертой которых является не прямое решение задачи, а **обучение** в процессе применения решений множества сходных задач
9. Средство поддержки принятия решений, использующееся в машинном обучении, анализе данных и статистике. Его дерева представляет собой «листья» и «ветки»
10. Основная единица слова номер 6 в кроссворде.
11. Алгоритм для спам-фильтров “Наивный …”

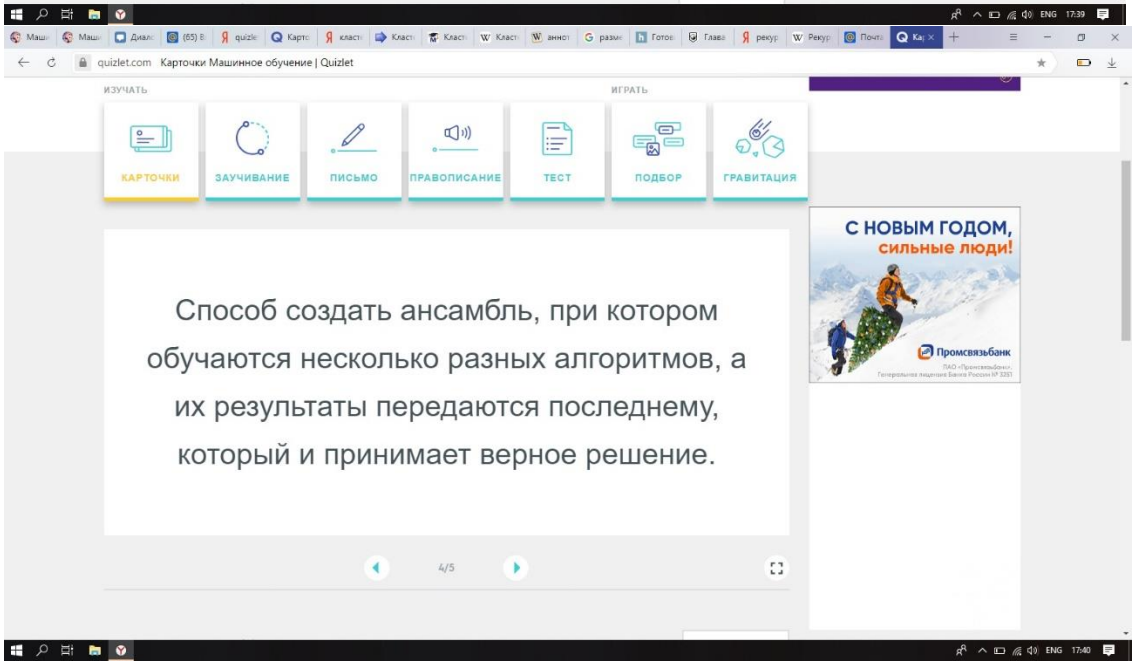
3. Обучающие карточки

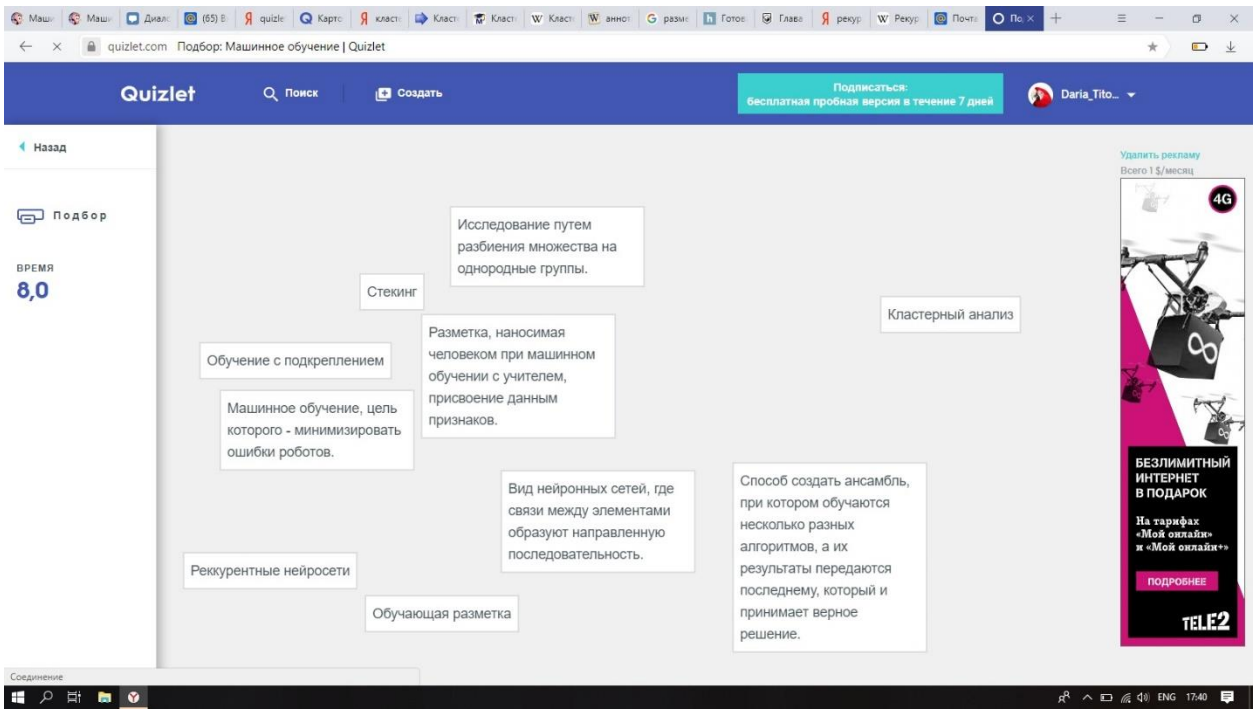
Ученикам предлагаются карточки с терминами. При переходе по этой ссылке ученики могут выбрать одно или несколько заданий и “играть” с терминами (карточки, заучивание, письмо, правописание…).






quizlet.com Карточки Машинное обучение | Quizlet
ИЗУЧАТЬ
ИГРАТЬ
КАРТОЧКИ
ЗАУЧИВАНИЕ
ПИСЬМО
ПРАВОПИСАНИЕ
ТЕСТ
ПОДБОР
ГРАВИТАЦИЯ
Стекинг
4/5
С НОВЫМ ГОДОМ,
сильные люди!
Промсвязьбанк


quizlet.com Карточки Машинное обучение | Quizlet
ИЗУЧАТЬ
ИГРАТЬ
КАРТОЧКИ
ЗАУЧИВАНИЕ
ПИСЬМО
ПРАВОПИСАНИЕ
ТЕСТ
ПОДБОР
ГРАВИТАЦИЯ
Способ создать ансамбль, при котором обучаются несколько разных алгоритмов, а их результаты передаются последнему, который и принимает верное решение.
4/5
С НОВЫМ ГОДОМ,
сильные люди!
Промсвязьбанк



4. Здесь ученики смогут найти загадки по пройденной теме. Загадки можно также и скачать в формате pdf.
5. В конце ученикам предлагается разбиться на 2 команды и поиграть в аналог игры “Alias”. Они могут взять слова из карточек или кроссворда и попытаться объяснить слова противоположной команде.