ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА

Лекции по курсу «Интеллектуальные системы»

Для студентов направления 090401

Очное обучение 2024-2025 учебный год

Лектор: Солдатова Ольга Петровна, к.т.н., доцент

Искусственный интеллект и машинное обучение

Лекция 1

Искусственный интеллект.

История развития.

Машинное обучение.

Виды, способы и задачи машинного обучения.

Функционалы ошибки и метрики оценки качества.

Формальное определение модели машинного обучения.

Понятие эмпирического риска.

Ошибки I и II рода.

Метрики оценки качества решения задачи классификации.

Построение ROC кривых.

Метрики оценки качества решения задачи регрессии.

Литература по курсу

1. Куприянов А.В. «Искусственный интеллект и машинное обучение» https://do.ssau.ru/moodle/course/view.php?id=1459
2. Осовский С. Нейронные сети для обработки информации / Пер. с пол. И.Д. Рудинского. – М.: Финансы и статистика, 2002. – 344 с.: ил.
3. Горбаченко, В. И. Интеллектуальные системы: нечеткие системы и сети : учебное пособие для вузов / В. И. Горбаченко, Б. С. Ахметов, О. Ю. Кузнецова. — 2-е изд., испр. и доп. — Москва : Издательство Юрайт, 2018. — 105 с. — (Университеты России). — ISBN 978-5-534-08359-0. — Текст : электронный // ЭБС Юрайт [сайт]. — URL: https://urait.ru/bcode/424887 – Режим доступа: https://urait.ru/bcode/424887
4. 4. Гафаров Ф.М. Искусственные нейронные сети и приложения: учеб. пособие / Ф.М. Гафаров, А.Ф. Галимянов. – Казань: Изд-во Казан. ун-та, 2018. – 121 с. – Текст : электронный. – Режим доступа: https://repository.kpfu.ru/?p_id=187099
5. Хайкин С. Нейронные сети: Полный курс: Пер. с англ. - 2-е изд. – М.: Вильямс, 2006. – 1104 с.: ил.
6. Борисов В.В., Круглов В.В., Федулов А.С. Нечёткие модели и сети. – М.: Горячая линия– Телеком, 2007. -284 с.: ил.
7. Рутковская Д., Пилиньский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечёткие системы: Пер. с польск. И.Д.Рудинского, - М.: Горячая линия – Телеком, 2007. – 452 с. ил.
8. Николенко С., Кадурин А., Архангельская Е. Глубокое обучение. — СПб.: Питер, 2018. — 480 с.: ил. — (Серия ≪Библиотека программиста≫).
9. Гудфеллоу Я., Бенджио И., Курвилль А. Глубокое обучение / пер. с анг. А. А. Слинкина. – 2-е изд., испр. – М.: ДМК Пресс, 2018. – 652 с.: цв. ил.

Искусственный интеллект

Почти за сто лет до построения компьютера (Lovelace, 1842) человек задался вопросом, смогут ли они стать разумными.

Сегодня **искусственный интеллект** (ИИ) – бурно развивающаяся дисциплина, имеющая многочисленные приложения. Термин искусственный интеллект (ИИ) является русским переводом английского термина **artifical intelligence**.

ИИ – это область информатики, предметом которой является разработка компьютерных систем, обладающих возможностями, традиционно связываемыми со способностями естественного интеллекта.

Искусственный интеллект:

1. Наука и технология создания интеллектуальных машин, особенно интеллектуальных компьютерных программ.
2. Свойство интеллектуальных систем выполнять творческие функции, которые традиционно считаются прерогативой человека.

Искусственный интеллект
Сильный и слабый ИИ

Теория **сильного ИИ** предполагает, что компьютеры могут приобрести способность мыслить и осознавать себя, хотя и не обязательно их мыслительный процесс будет подобен человеческому.

Теория **слабого ИИ** такую возможность отвергает.

Сильный ИИ обладает следующими свойствами:

- Принятие решений, использование стратегий, решение задач, действие в условиях неопределённости;
- Представление знаний, включая общее представление о реальности;
- Планирование;
- Обучение;
- Общение на естественном языке;
- Объединение всех этих способностей воедино для достижения общих целей.

История развития

Считается, что искусственный интеллект как наука начался с **теста Тьюринга**. Формулировка теста впервые появилась в знаменитой статье **Computing Machinery and Intelligence**, которую Алан Тьюринг выпустил в 1950 году. Следует отметить, что возможность создания ≪мыслящих машин≫ и наличия интеллекта у компьютеров обсуждалась и самим Тьюрингом, и его коллегами к тому времени уже как минимум лет десять; это была частая тема для дискуссий в английском Ratio Club, к которому принадлежал и Тьюринг.

Этот тест заключается в следующем: человек и машина при помощи записок ведут диалог, а судья (человек), находясь в другом месте, должен определить по запискам, кому они принадлежат, человеку или машине. Если ему это не удается, то это будет означать, что машина успешно прошла тест. Тьюринг посчитал, что к 2000 году машины будут способны ввести в заблуждение 30% собеседников при условии длительности беседы не более 5 минут.

7 июня 2014 года тест для определения искусственного интеллекта, придуманный британским математиком Аланом Тьюрингом 64 года назад, был пройден. Чатбот, созданный под руководством российского программиста **Владимира Веселова**, сумел преодолеть 30% барьер, установленный Тьюрингом более полувека назад. В 2012-м году Густман победил в соревнованиях, посвященных 100-летию со дня рождения Алана Тьюринга. В 2012 результат был 29.2 %, в 2014 — 33.3%.

История развития

Программа «Евгений Густман» состоит из базы знаний, которая имеет около трех тысяч шаблонов распознавания фраз пользователя. Были использованы также различные методы управления диалогом, которые позволяют имитировать именно человека, а не поисковую машину. Евгений старается направить беседу в нужное ему русло, стараясь создавать такие ситуации, когда его фразы выглядят человекоподобно.

При короткой продолжительности беседы — 5 минут — такой подход часто срабатывает». Чатбот убедил 33 процента судей (всего в эксперименте участвовало 30 судей и 5 чатботов), в том, что он является 13-летним мальчиком из Одессы.

С 9 по 15 марта 2016 года состоялся матч по игре **«Го»,** проходивший между компьютерной программой **AlphaGo**, разработанной британской компанией **Google Deep Mind,** и корейским профессионалом Ли Седолем. Всего было сыграно 5 партий. Игра велась по китайским правилам. Для работы AlphaGo использовались 1920 процессоров и 280 графических процессоров, работающих в распределённой сети. Матч завершился победой AlphaGo со счётом 4:1.

AlphaGo имеет значительные отличия от программ-предшественников. Она задействует нейронные сети, где эвристические оценки не основываются на конкретных значениях переменных, закодированных людьми, а в значительной степени, извлекаются самой программой, путём десятков миллионов просмотров сыгранных партий и собственных партий с самой собой.

История развития

Даже сама команда разработчиков AlphaGo не в состоянии указать, каким образом AlphaGo оценивает позицию в партии и выбирает свой следующий ход. Метод Монте-Карло также стал одним из основных способов повышения эффективности программы в выборе ходов. При создании программы использовались данные из теории распознавания образов и машинного обучения.

В 1956 году Джон Маккарти (John McCarthy), Марвин Минский (Marvin Minsky), Натаниэль Рочестер (Nathaniel Rochester) и Клод Шеннон (Claude Shannon) — организовали Дартмутский семинар. В заявке на грант для проведения семинара Джон Маккарти писал: ≪Мы предлагаем исследование искусственного интеллекта сроком на два месяца с участием десяти человек летом 1956 года в Дартмутском колледже, Гановер, Нью-Гемпшир. Исследование основано на предположении, что всякий аспект обучения или любое другое свойство интеллекта может в принципе быть столь точно описано, что машина сможет его имитировать. Мы попытаемся понять, как обучить машины использовать естественные языки, формировать абстракции и концепции, решать задачи, сейчас подвластные только людям, и улучшать самих себя. Мы считаем, что существенное продвижение в одной или более из этих проблем вполне возможно, если специально подобранная группа ученых будет работать над этим в течение лета≫.

История развития

Всё началось с исследований, которые продолжали начатую логиками тему **автоматического логического вывода (automated theorem proving**). Первые программы пытались строить выводы в заданных исследователями формальных системах. Например, появившаяся в 1959 году программа **≪Универсальный решатель задач≫ (General Problem Solver, G.P.S.)** могла строить вывод в системах, заданных логическими формулами определенного вида. Вся эта наука тогда называлась **кибернетикой** вслед за книгой Норберта Винера ≪Кибернетика, или Управление и связь в животном и машине ≫. Тогда же, во второй половине 1950-х и начале 1960-х годов, появились и самообучающиеся машины, в частности персептрон Розенблатта.

Первой работой, предлагающей математическую модель нейрона и конструкцию искусственной нейронной сети, была статья **Уоррена Маккаллока (Warren McCulloch) и Уолтера Питтса**, опубликованная в 1943 году. Авторы отмечают, что из-за бинарной природы нейронной активности (нейрон либо≪включен≫, либо ≪выключен≫, практически без промежуточных состояний). Нейроны удобно описывать в терминах пропозициональной логики, а для нейронных сетей разрабатывают целый логический аппарат, формализующий ациклические графы

История развития

Сама конструкция искусственного нейрона, получилась очень современной: линейная комбинация входов, которая затем поступает на вход нелинейности в виде ≪ступеньки≫, сравнивающей результат с некоторым порогом.

Такая ≪нейронная сеть≫ еще не умела обучаться ни в каком смысле слова, и ее непосредственный эффект был в том, чтобы вообще предложить идею формализации нейронных сетей и нейронной активности, показав, что у человека в голове вполне может содержаться собранная из нейронов машина Тьюринга.

Статья Маккаллока и Питтса прошла практически незамеченной среди нейробиологов, но создатель кибернетики **Норберт Винер** сразу понял перспективы искусственных нейронных сетей и идеи о том, как мышление может самопроизвольно возникать из таких простых логических элементов.

Винер, **Маккаллок, Питтс, фон Нейман** и **Джером Леттвин**, стали работать над тем, как адаптировать статистическую механику для моделирования мышления, а потом построить работающий компьютер на основе моделирования нейронов. Память в таком компьютере предполагалось получить из замкнутых контуров активации нейронов, когда последовательная активация превращается в самоподдерживающийся процесс.

Ещё один шаг в развитии искусственных нейронных сетей сделал **Дональд Хебб** в своей книге **The Organization of Behaviour**, вышедшей в 1949 году.

История развития

Основная идея, перешедшая из книги Хебба в современное машинное обучение практически без изменений, — это так называемое **правило Хебба**, которое сам Дональд Хебб в первоисточнике формулировал так: ≪Когда аксон клетки А находится достаточно близко, чтобы возбудить клетку В, и многократно или постоянно участвует в том, чтобы ее активировать, в одной или обеих клетках происходит некий процесс роста или изменение метаболизма, в результате которого эффективность А как клетки, возбуждающей В, увеличивается≫. Проще говоря, если связь между двумя нейронами часто используется, она от этого становится сильнее. Эта простая, но очень мощная идея не только мотивировала дальнейшие исследования, но и сама по себе легла в основу так называемого **обучения по Хеббу (Hebbian learning)**, группы методов обучения без учителя, основанных на этом базовом правиле. Обучение **сети Хопфилда (Hopfield networks)** основано на этом принципе.

В 1951 году **Марвин Минский**, и его аспирант **Дин Эдмунд (Dean Edmund)** построили сеть из сорока синапсов. Синапсы были случайным образом соединены друг с другом и обучались по правилам Хебба на основе вознаграждений, которые им давали исследователи. Эта модель получила название SNARC (Stochastic Neural Analog Reinforcement Calculator), и хотя непосредственного развития она не получила, это была первая реализация обучения с подкреплением.

В 1958 году **Фрэнк Розенблатт** предложил модель первого персептрона, для которого был разработан алгоритм обучения – правило персептрона.

История развития

В 1969 году **Марвин Минский и Сеймур Пейперт** опубликовали книгу с названием ≪Персептроны≫, которая содержала критику конструкции персептронов Розенблатта.

Действительно, один персептрон Розенблатта может обучиться разделять только те множества точек, между которыми можно провести гиперплоскость (такие множества логично называются **линейно разделимыми**), а существует много линейно неразделимых множеств. Например, одинокий линейный персептрон никогда не обучится реализовывать функцию XOR: множество ее нулей и множество ее единиц, линейно неразделимы. Другое минус линейной конструкции состоит в том, что комбинировать линейные персептроны в сеть тоже не имеет никакого смысла: сеть из большого числа линейных персептронов сможет реализовать только те же самые линейные функции.

Негативные утверждения в книге ≪Персептроны≫ касались только некоторых конкретных архитектур: например, Минский и Пейперт показали, что сети с одним скрытым слоем не могут вычислять некоторые функции, если персептроны скрытого слоя не связаны со всеми.

Сегодня это нельзя считать серьезными аргументами против персептронов: линейный классификатор не может реализовать XOR, но сеть из нескольких классификаторов с любой нелинейностью сделает это. Соединять линейные персептроны в сеть бессмысленно, но если в персептрон добавить нелинейную функцию активации, смысл тут же появляется.

История развития

Однако в конце 1960-х годов о нейронных сетях и тем более глубоких сетях еще не было широко известно, и книга Минского и Пейперта, получившая широкую известность, надолго оттолкнула многих исследователей от продолжения изучения персептронов и вообще нейронных сетей. Целых десять лет заниматься нейронными сетями было немодно и неприбыльно.

Тем не менее, в 1970-е годы был достигнут очень серьезный прогресс в разработке и изучении нейронных сетей.

Затем исследователи снова вспомнили о нейронных сетях**: к началу 1980-х годов был разработан алгоритм обратного распространения ошибки,** что открыло дорогу самым разнообразным архитектурам.

Следующим шагом в нейробиологии стали работы **Хьюбела** и **Визеля**, которые смогли достаточно подробно изучить активации нейронов в зрительной коре, что стало мотивацией для появления сверточных нейронных сетей и в некотором смысле глубокого обучения вообще.

К концу восьмидесятых уже появилась большая часть основных архитектур: сверточные сети, автокодировщики, рекуррентные сети и т.д.

В 1990-е годы основной акцент сместился на машинное обучение и поиск закономерностей в данных, нейронных сетей и Интернета, становилось все больше, компьютеры становились все быстрее. В итоге в середине 2000-х годов новая идея машинного обучения получила широкое распространие.

Развитие систем ИИ

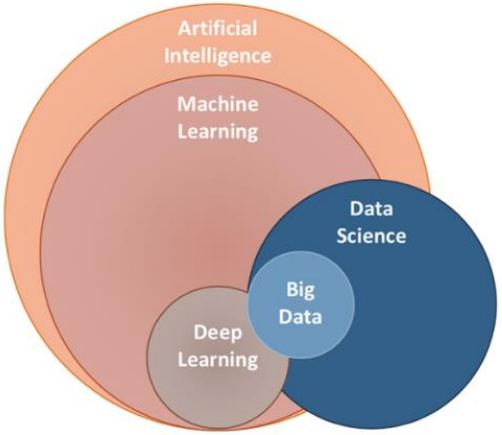
- 1997: IBM Deep Blue обыграл чемпиона мира по шахматам
- 2005: Беспилотный автомобиль: DARPA Grand Challenge
- 2006: Google Translate – статистический машинный перевод
- 2011: 40 лет DARPA CALO привели к созданию Apple Siri
- 2011: IBM Watson победил в ТВ-игре «Jeopardy!»
- 2011–2018: ImageNet: 25% → 2,5% ошибок против 5% у людей
- 2015: Фонд OpenAI в $1 млрд. Илона Маска и Сэма Альтмана
- 2016: DeepMind, OpenAI: динамическое обучение играм Atari
- 2016: Google DeepMind обыграл чемпиона мира по игре ГО
- 2017: OpenAI обыграл чемпиона мира по компьютерной игре Dota 2
- … … …

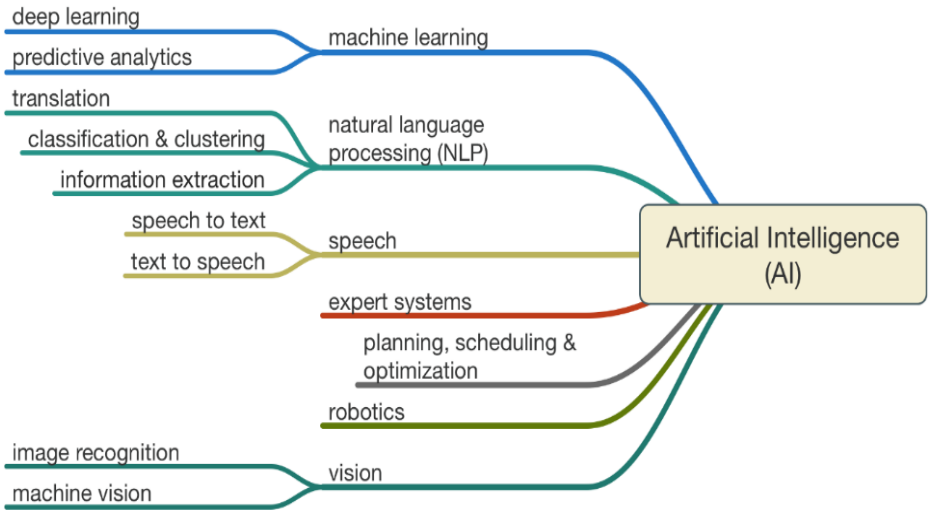
OpenAI Google DeepMind

ImageNet: открытая выборка 15M размеченных изображений

Искусственный интеллект и машинное обучение

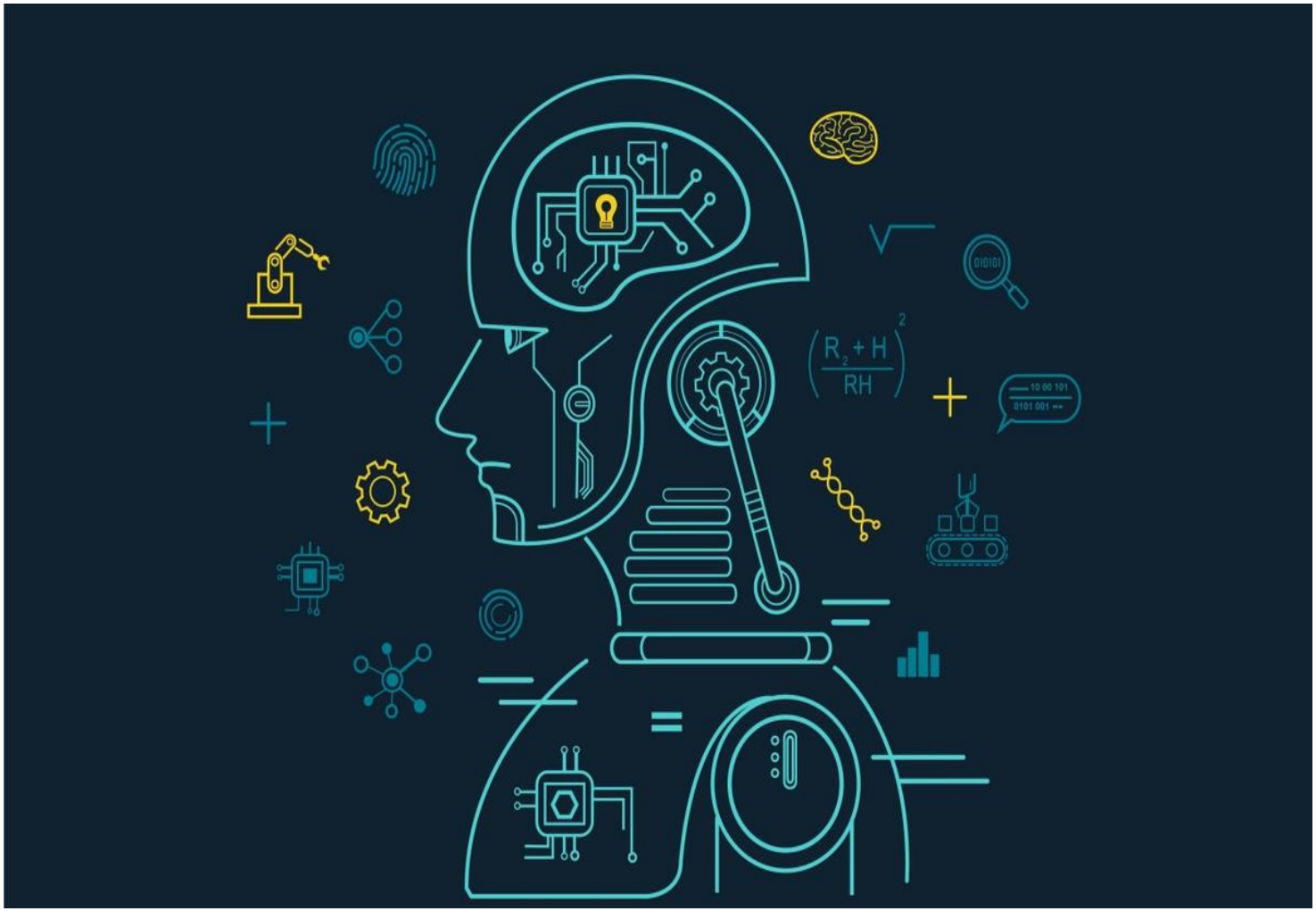
- Статистический анализ данных (Statistical Data Analysis)
- Искусственный интеллект (Artificial Intelligence) — 1955
- Распознавание образов (Pattern Recognition)
- Машинное обучение (Machine Learning) — 1959
- Статистическое обучение (Statistical Learning)
- Интеллектуальный анализ данных (Data Mining) — 1989





- Машинный интеллект (Machine Intelligence) — 2000
- Бизнес-аналитика (Business Intelligence, Business Analytics)
- Предсказательная аналитика (Predictive Analytics) — 2007
- Большие данные (Big Data) — 2008
- Аналитика больших данных (Big Data Analytics)
- Наука о данных (Data Science) — 2011

Машинное обучение



Машинное обучение

Понятие машинного обучения основано на понятии компьютерного моделирования объектов, явлений или процессов окружающего нас мира.

Модель это совокупность законов, правил, описывающих поведение изучаемого процесса, явления, объекта. Модель может быть физической. Но чаще всего модель представляется в виде математических формул, ее так и называют ***математическая модель***. Для разных целей могут потребоваться разные модели одного и того же объекта, они могут быть проще и сложней, менее или более точные, описывать разные стороны поведения объекта. Представляя модели в виде формул, уравнений, можно их решить и тем самым предсказать поведение исследуемого объекта. Точность предсказания зависит, в первую очередь, от точности модели, насколько хорошо та описывает поведение объекта. Точность также зависит и от точности решения уравнений модели. Поведение многих объектов похоже, поэтому для них надо использовать похожие модели. Например, тела массой 1 кг, 2 кг или 3 кг, при падении подчиняются одним и тем же законам, но двигаются по-разному. Это можно учесть если ввести в уравнение, описывающее движение массивных тел, некоторый ***параметр*** – массу этого тела и изменять только параметр, не изменяя форму самого уравнения. Модели с параметрами называются ***параметрическими моделями*** и удобны для описания множества похожих объектов.

Машинное обучение

Если бы было возможно сделать модель с параметрами, такую, что при изменении параметров описываемое ей поведение изменялось бы в широких пределах (в идеале давало бы любое поведение), то можно было бы подбирать параметры модели так, чтобы в конце концов поведение модели совпало с измеренным поведением объекта. Каждый раз, подбирая параметры, надо передавать информацию модели, плохо или хорошо она описывает поведение объекта, такой процесс называют ***обучением***.

Если модель создается в компьютере, такой процесс называют ***машинным обучением***. Модель должна вести себя также, как и объект, обучение это подбор параметров модели для достижения заданной цели – уменьшить ***ошибку*** модели, отклонение ее поведения от заданного поведения объекта. Идеал – нулевая ошибка – редко достижим, поэтому стремятся уменьшить ошибку до какой-то приемлемой величины.

Однако цель обучения не просто запомнить поведение объекта на измеренных данных, но в том, чтобы эта модель могла давать правдоподобные результаты и на других примерах, на которых ее не обучали. Также и модель, должна научиться моделировать объект не только на данных о поведении объекта на которых ее обучали.

Машинное обучение

Чтобы узнать, а как модель себя ведет на других данных (данные, на которых обучали, называют обучающими), такую модель ***тестируют***, а данные на которых ее тестируют, называют тестовыми.

Можно посчитать и ошибку тестирования. Важно, чтобы тестовые данные действительно отличались от обучающих. Способность модели давать приемлемые результаты на тестовых данных называют ***обобщающей способностью***, и именно повышение ее является целью обучения.

Различают ***параметры модели*** (обучаемые параметры) – те числа, которые изменяются в процессе обучения – и другие числа, которые описывают модель, но не меняются при обучении, а задаются до его начала (размер модели, число ее элементов и др.). Последние называют ***гиперпараметрами***. Из-за сложности моделируемых объектов, наличия шумов, модели машинного обучения дают ***приближенные*** результаты. Если ошибки обучения и тестирования маленькие и приемлемые, то, скорей всего (но не гарантировано) модель хорошая и её можно использовать.

Машинное обучение

Модели с обучением очень ***требовательны к данным***. Данные, на которых модель обучается и тестируется:

1) должны быть ***разнообразны***, на одинаковых данных, ничему новому модель не научится.
2) должны быть ***непротиворечивы***, когда данных много некоторые противоречия (ошибки) допустимы, модель с хорошей обобщающей способностью справится с этим, но таких противоречий должно быть немного.
3) должны быть ***релевантны*** задаче, действительно полностью описывать реальные моделируемые объекты. При измерениях неизбежны ошибки, если они маленькие, измеренные данные просто примерно описывают объект, но если большие – то невозможно узнать, что конкретное измеренное данное это ошибка или действительное описание объекта.
4) ***тестовые данные должны отличаться от обучающих***, иначе тест бесполезен.

Это понятные, но трудно выполнимые требования. И только для тех областей, где можно собрать требуемые данные, модели с обучением будут хорошо работать.

Виды машинного обучения

В зависимости от того, какие именно данные доступны и какие цели преследует обучение, можно выделить несколько видов обучения:

1) ***Обучение с учителем.*** Данные представляются в виде пар примеров входов и эталонных выходов («указания учителя») модели. Например, при распознавании изображений примерами входов являются сами изображения, а эталонными выходами – название того, что на изображении изображено. Модель учится на этих парах, пытаясь подобрать свои параметры так, чтобы ее выходы совпали или были близки к эталонным. Это наиболее распространенный вид обучения. «Учитель» — это процесс, который собрал эти пары данных (это может быть и человек, и компьютер).

2) ***Обучение без учителя*** или самообучение. Бывает, что эталонные выходы модели неизвестны, но есть примеры входов. В этом случае модель может попытаться установить взаимосвязи между элементами входов, выделить какие-то общие черты. Для этого должен быть задан критерий такого выделения (явно или неявно). Это тоже популярный вид обучения, но более сложный, часто используется не сам по себе, а в дополнение к обучению с учителем.

Виды машинного обучения

3) ***Обучение с подкреплением***. Бывает, что «учитель» сам не знает, что именно должно быть, но понимает какие действия обучаемого хорошие, а какие плохие. Перебирая возможные варианты действий модель будет получать штрафы или награды за действия и может научиться, какие действия приводят к наибольшей суммарной награде. Такой вид обучения отличается и от обучения с учителем, поскольку здесь нет эталонных данных, и от обучения без учителя, поскольку все-таки есть некоторая информация учителя в виде наград и штрафов от среды. Такой подход используют, например, для создания ботов для компьютерных игр. Можно сочетать разные виды обучения между собой.

В зависимости от того, как часто поступают данные и как часто нужен результат, все перечисленные выше виды обучения могут выполняться несколькими способами:

- ✓ Динамическое или ***онлайн обучение***. Как только данное поступило, необходимо сразу же, выдавать результат для него и доучивать модель (т.е. изменять параметры ее) для этого данного. Для большинства моделей с этим видом обучения важен порядок поступления данных: те же самые данные для той же самой модели, но представленные в другом порядке, могут привести к другому результату. Онлайн обучение хорошо тем, что оно быстрое, но обычно менее точное, и зависит от порядка поступления данных.

Способы машинного обучения

- Отложенное или ***оффлайн обучение***. Здесь параметры модели изменяются только после того, как для всех данных будут рассчитаны выходы. Обычно порядок поступления данных не важен. Оффлайн обучение хорошо тем, что обычно не зависит от порядка поступления данных и более точное, но зато более медленное.
- Компромиссный вариант между онлайн и оффлайн обучением это ***пакетное обучение***. Данные разбиваются на маленькие группы (пакеты, или по-английски batch), и параметры модели изменяются только после расчета пакета целиком. Изменяя размер пакета, сколько в него входит данных, можно балансировать между онлайн (размер пакета = 1) и оффлайн (размер пакета = количеству всех данных) обучением. Сегодня это самый распространенный вариант организации обучения. Так как он похож на онлайн обучение, то результат может зависеть от порядка поступления пакетов, чтобы как-то с этим бороться пакеты часто перемешивают случайно. Вариант, когда модель может сама выбирать порядок поступления данных, называют ***активное обучение***.

Задачи машинного обучения

Множество задач можно решить с помощью машинного обучения, вот только некоторые из них:

1) ***Классификация***. Имея набор примеров входов и список классов (названий, меток, номеров) нужно распределить примеры входов по этим классам. Например, отличить мальчиков от девочек по фотографии. Список классов состоит из двух названий «мальчик» или «девочка», примеры входов – фотографии. Список возможных классов известен заранее и не бесконечен. Модель, которая решает такую задачу называется классификатором. Классификация может быть двухклассовой или бинарной, когда классов всего два, или многоклассовой, когда классов больше (одноклассовая классификация бессмыслена). Часто вводят специальный класс «не класс» (шум), к которому относят все, что не смогли отнести к другим классам. Обычно решается с помощью моделей с обучением с учителем.

2) ***Кластеризация***. Имея только примеры входов, разбить их на группы (кластеры) так, чтобы внутри кластеров примеры были чем-то похожи, а примеры из разных кластеров чем-то отличались между собой. Обычно решается с помощью моделей с обучением без учителя, например, разделить звукозаписи по жанрам, это задача кластеризации. Часто число кластеров неизвестно заранее, и даже их названия неизвестны. Также вводят специальный кластер «не кластер» (шум), в который группируют все, что не попало в другие кластеры.

Задачи машинного обучения

3) ***Регрессия***. Имея набор примеров входов и соответствующих им выходов научиться выдавать выходы для других примеров входов с той же взаимосвязью входов и выходов. Отличается от классификации тем, что в классификации выход это класс (номер, название или другое его представление), т.е. дискретная величина, а в регрессии выход непрерывен (не дискретен). Классификация – частный случай регрессии, т.к. дискретную величину можно представить вариантом непрерывной. Пример: зная температуру на улице, определить температуру в доме.

4) ***Прогнозирование***. Зная, как изменялась какая-то величина до определенного времени, узнать, как она будет изменяться после. Например, узнать температуру на улице на завтра, зная, как она изменялась раньше. Обычно сводится к задаче регрессии.

5) ***Оптимизация***. Имея примеры данных и некоторую функцию, заданную на этих данных, найти такие данные, для которых функция принимает оптимальное (минимальное или максимальное) значение. Например, есть набор шариков разного размера и материала, зададим вес как функцию от размера и плотности материала шарика, нужно найти шарик с максимальным весом. Обучение — это, по сути, оптимизация функции ошибки модели. Поэтому, так или иначе, все задачи сводятся к оптимизации некоторой функции. Задачи мини- и максимизации не следует различать, потому что это одинаковые задачи. Если определяется минимум функции, то при смене знака определяется максимум функции.

Задачи машинного обучения

Другие задачи так или иначе сводятся к перечисленным выше:

а) ***ранжирование***. Отличается от регрессии или классификации тем, что выходы надо получить сразу на множестве примеров, после чего отсортировать их по значениям выходов. Встречается, например, в поисковых системах типа Google: по имеющимся данным (в поисковой системе это будут тексты документов и прошлое поведение пользователей) расставить имеющиеся объекты в порядке убывания целевой функции. Эта задача похожа на задачу регрессии — так или иначе нужно предсказывать непрерывную целевую функцию. Но при этом значение самой функции не важно. Имеют значение только результаты сравнения этой функции на разных объектах (какой документ будет выше другого в поисковой выдаче)

б) ***сокращение размерности***. Входы часто являются объектами большой размерности (вектора, матрицы, массивы), однако не все ли элементы входов одинаково важны. Зачастую можно отбросить или преобразовать входы так, что их размерность будет меньше, но при этом наиболее важные черты данных сохранятся.

Формальное определение модели машинного обучения

Для построения модели требуется построить отображение $a : X \to Y$ такое что:

- Отображение воспроизводит на обучающей выборке требуемые ответы с некоторой вероятностью.
- Отображение способно к обобщению – аппроксимирует целевое отображение на всём множестве (проверяется на тестовой выборке).
- Возможны дополнительные ограничения целевой функции – гладкость, монотонность и т.д. (определяется задачей).

Для задачи классификации: Y – конечное множество.

- Бинарная классификация- $Y = \{0,1\}$.
- Классификация по нескольким классам - $Y = \{1,\ldots,M\}$.

Для задачи регрессии: $Y = \Re$, действительные числа.

Формальное определение признака объекта

Признак – это результат измерения некоторой величины, отличительная характеристика объекта, качественная или количественная.

Формально признаком называется отображение $f : X \to D_f$, где D_f – множество допустимых значений признака.

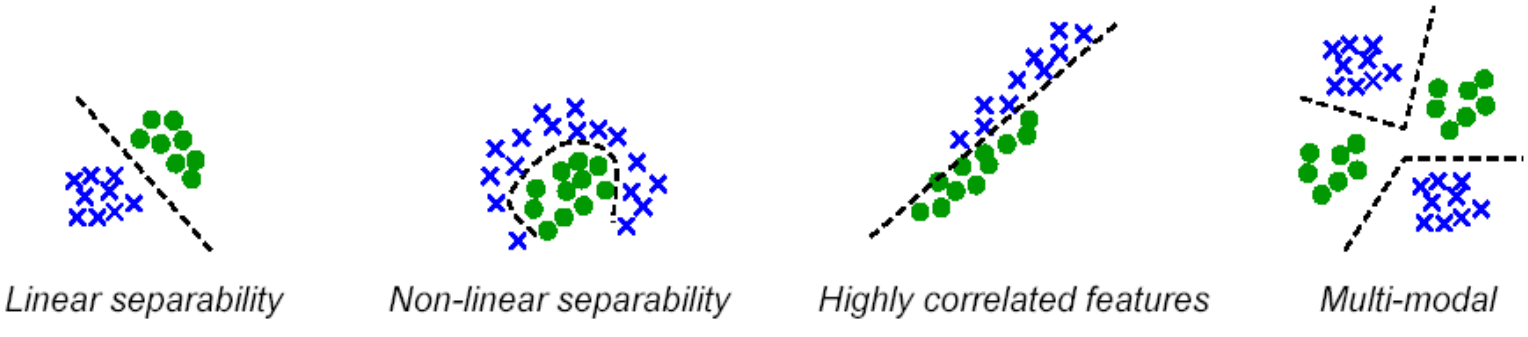
- Комбинация *N* признаков, представленная в виде вектора называется **вектором признаков.**
- N-мерное пространство, определяемое векторами признаков, называется **пространством признаков.**
- Объект — представляется N-мерными точками в пространстве признаков.

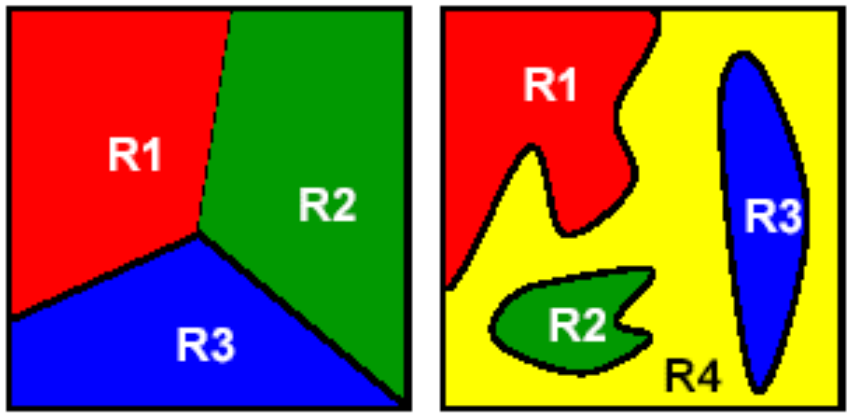
Что такое «хороший» вектор признаков? Качество вектора признаков связано с его разделяющей способностью для объектов из разных классов

- Образцы из одного ы иметь подобные вектора признаков.
- Образцы из разных ы иметь различные вектора признаков.

Формальное определение признака объекта

Свойства, зависящие от векторов признаков:

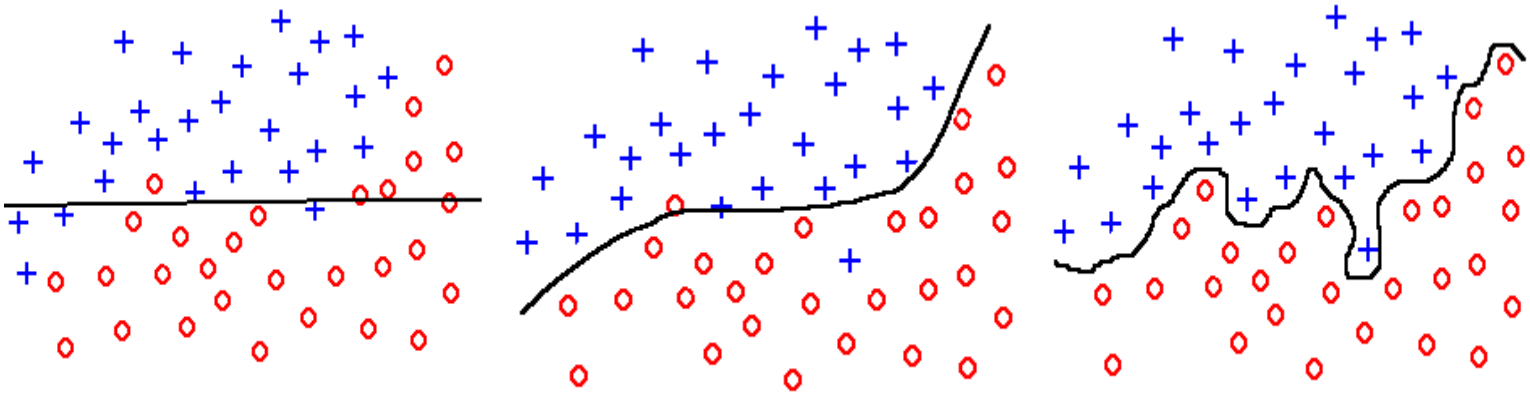




Проблемы недообучения и переобучения

Переобучение (overtraining, overfitting) — вероятность ошибки обученной модели на объектах тестовой выборки оказывается существенно выше, чем средняя ошибка на обучающей выборке.

Недообучение (underfitting) — алгоритм обучения не обеспечивает достаточно малой величины средней ошибки на обучающей выборке. Недообучение возникает при использовании недостаточно сложных моделей.



излишнее обобщение, недообучение, underfitting

хорошая модель good fit

излишняя избирательность, переобучение, overfitting

Формальное определение модели машинного обучения с учителем

- Пусть существуют два множества:
  - Множество *объектов – образов X*.
  - Множество *ответов Y*.
- Существует целевая функция $y^* : X \to Y$, значения которой известны только на конечном подмножестве объектов:

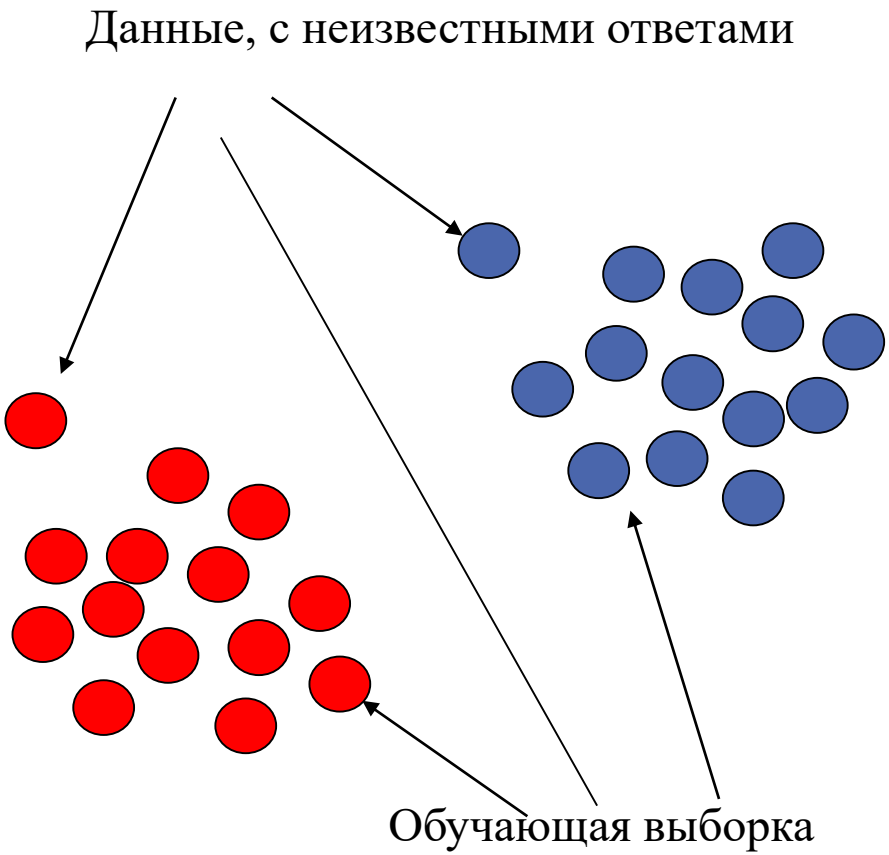
$$\{x_1, \ldots, x_n\} \subset X \quad .$$

- Существует совокупность пар «объект-ответ»

$X^N = (x_i, y_i)_{i=1}^N$, где X^N - обучающая выборка.

Задача обучения заключается в том, чтобы по X^N выборке построить решающую функцию (отображение) $a : X \to Y$, которая бы приближала целевую функцию не только на объектах обучающей выборки, но и на всем множестве

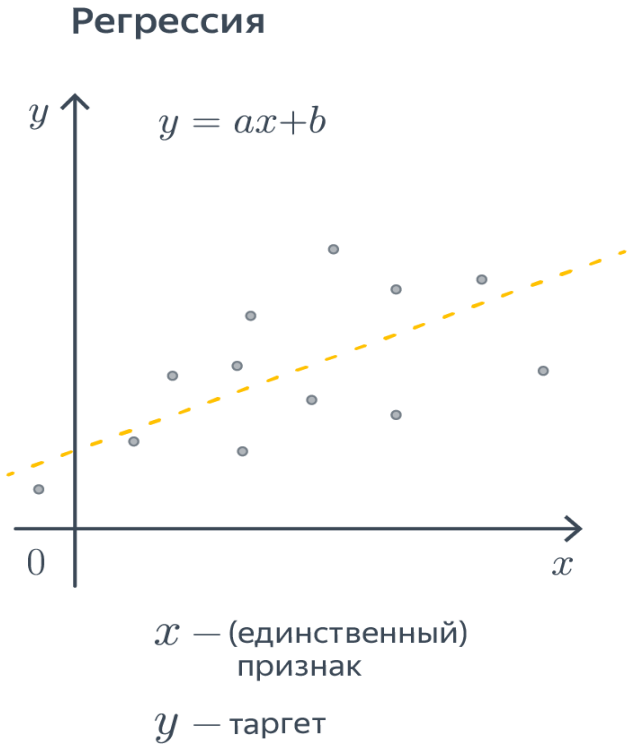
Иллюстрация задачи машинного обучения с учителем

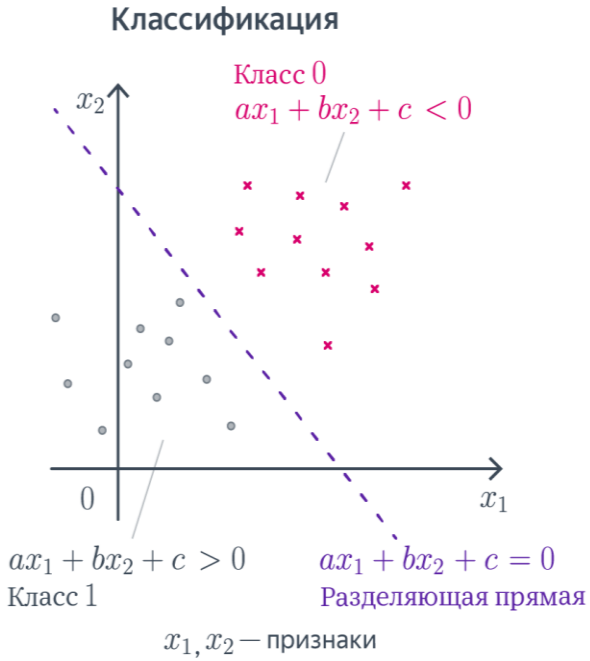


Формальное определение модели алгоритма

- Требуется построить отображение (гипотезу)
$$a : X \to Y .$$
- Пусть ***A*** – параметрическое семейство отображений
$$A = \{a(x, \gamma) \mid a : X \times \Gamma \to Y\} ,$$ где Γ – пространство допустимых значений параметра γ (*пространство поиска*).
- Будем выбирать отображение для решение задачи из ***A:***
  - Процесс выбора – ***обучение.***
  - Построение отображения $\mu : X^l \to a$ по обучающей выборке X^l – ***метод обучения .***
  - Обучение сводится к поиску точки в пространстве поиска Γ.

Модели алгоритмов для решения задач регрессии и классификации





Функционалы ошибки и метрики оценки качества

Обучение – это подбор параметров модели с целью уменьшения функции ошибки (по-английски loss). Сами функции ошибки разнообразны, зависят от задачи, которую требуется решить, используемой модели, данных, на которых проходит обучение.

Самой простой и распространенной функцией ошибки ***L***, является функция квадрата расстояния между вектором ***Y*** выхода модели и указаний учителя ***D*** (эталонного выхода) на обучающих данных ***X: L2=(Y(X)-T(X))^2.***

Если примеров данных несколько (для пакетного или оффлайн обучения), то берут среднее значение L между примерами. Любые другие расстояния, или, как их называют математики, ***меры***, тоже могут использоваться в функциях ошибки (например модуль разности, а не квадрат, это так называемая ***L1*** мера).

Функция ошибки может быть и очень сложной, например, в обучении с подкреплением, она определяется симуляцией внешней среды. Одну и ту же модель можно использовать с разными функциями ошибки, получая разные результаты. Часто в функцию ошибки добавляют ограничения для модели, например, чтобы ее параметры были не очень велики.

Функционалы ошибки и метрики оценки качества

Вообще, сложилась такая ситуация, что для подавляющего большинства моделей оптимизируется вовсе не то, что надо. Для задач классификации, функция ошибки обычно это средний квадрат расстояния между векторами выхода и указаний учителя. Но цель решения задачи классификации - уменьшить число неправильно классифицированных примеров. В идеале, когда ошибка ноль, то и число неправильно классифицированных примеров тоже ноль, но в остальном эти функции – разные.

К функции ошибки, которая будет использоваться для оптимизации параметров, предъявляют жесткие требования, например, дифференцируемости. Число ошибок — это не дифференцируемая функция, а квадрат расстояния – дифференцируемая. Тогда оценкой качества решения задачи будет являться **метрика (по-английски metrics**). Для одной задачи можно посчитать множество разных метрик. Именно метрики описывают качество решения задачи, а функция ошибки — это косвенное средство достижения заданных метрик.

Определение эмпирического риска

- Пусть $X^l = \{x_1, ..., x_l\}$ - обучающая выборка.
- Эмпирический риск (ошибка обучения) определяется как:

$$R_{emp}(a, X^l) = \frac{1}{k} \sum_{i=1}^{l} L(a(x_i), y_i)$$

- Задача обучения - минимизация ошибки обучения, то есть минимизации эмпирического риска:

$$\mu(X^l) = \arg\min_{a \in A} R_{emp}(a, X^l)$$

Таким образом задача машинного обучения сводится к задаче оптимизации!

$L(a, x)$ – функция ошибки - величина ошибки алгоритма a на объекте x .

Эмпирический риск

Функция ошибки $L = Y \times Y \to R$ характеризует отличие правильного ответа от ответа данного построенным алгоритмом:

$L(y, y') = [y \neq y']$ - индикатор ошибки,

$L(y, y') = |y - y'|$ - отклонение,

$L(y, y') = (y - y')^2$ - квадратичное отклонение,

$L(y, y') = [|y - y'| > \delta]$ - индикатор существенного отклонения,

$y' = y^*(x)$ - целевая функция,

$y = a(x)$ - алгоритм.

Алгоритм обучения обладает *способностью к обобщению*, если вероятность ошибки на тестовой выборке достаточно мала или хотя бы предсказуема, то есть не сильно отличается от ошибки на обучающей выборке.

Проблема обобщения: малый **эмпирический риск R_{emp}** не означает, что **истинный ожидаемый риск R** будет мал

Пример расчёта вероятности

- Пусть некий тест на какую-нибудь болезнь имеет вероятность успеха 95%.
  - 5% — вероятность как позитивной, так и негативной ошибки.
- Всего болезнь имеется у 1% респондентов.
- Пусть некий человек получил позитивный результат теста.
  - тест говорит, что он болен.
- С какой вероятностью он действительно болен?

$$P(y/x)$$

Пусть: *t*-результат теста; *d*-наличие болезни.

По теореме Байеса

$$P(d=1|t=1)=\frac{P(t=1|d=1)P(t=1)}{P(d=1|t=1)P(t=1)+P(d=1|t=0)P(t=0)}$$

$$=\frac{0{,}95\times 0{,}01}{0{,}95\times 0{,}01+0{,}05\times 0{,}99}$$

Ответ: ***16%***

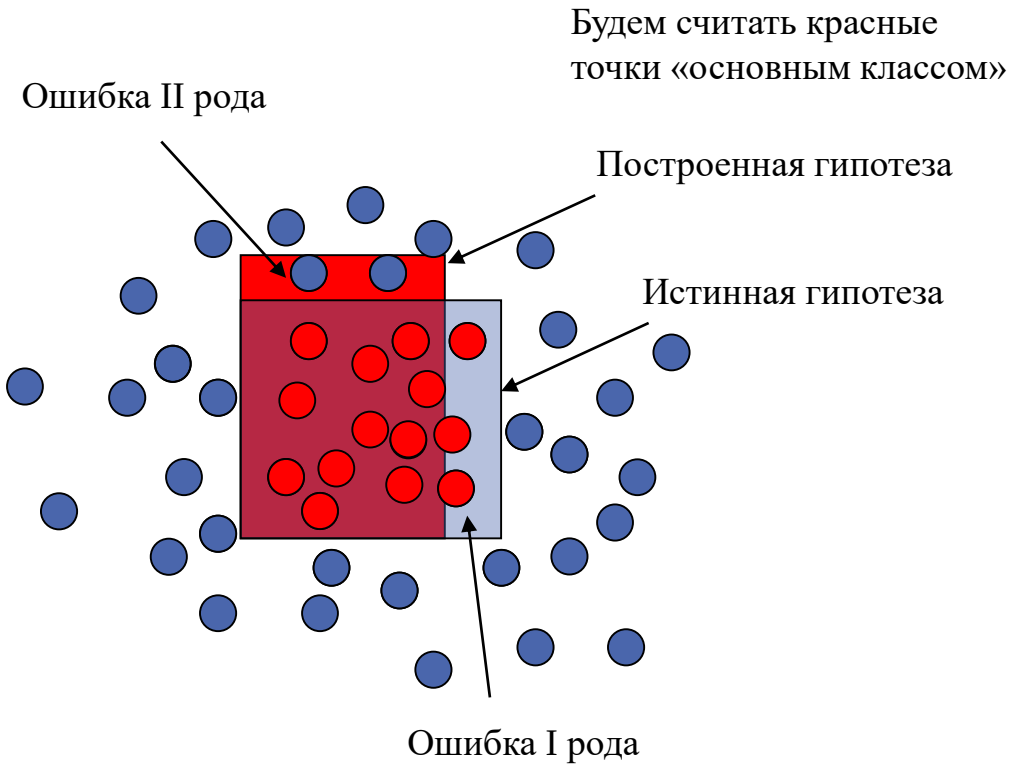
Виды ошибок

- Допустим, существует «основной класс».
  - Обычно, это класс, при обнаружении которого, предпринимается какое-либо действие;
  - Например, при постановке диагноза основным классом будет «болен», а вторичным классом «здоров».
- Измерения ошибки, как оценки «вероятности выдать неверный ответ», может быть не всегда достаточно:
  - 16% ошибки при постановке диагноза может означать как и то что, 16% больных будут признаны здоровыми (и возможно умрут от отсутствия лечения), так и то, что 16% здоровых больными (и деньги на лечение будут потрачены зря).
- При неравнозначности ошибок для разных классов вводят понятие **ошибки первого и второго рода** и замеряют их по отдельности.

Ошибки I и II рода

- **Ошибка первого рода** равна вероятности принять основной класс за вторичный.
  - Вероятность «промаха», когда искомый объект будет пропущен.
- **Ошибка второго рода** равна вероятности принять вторичный класс за основной.
  - Вероятность «ложной тревоги», когда за искомый объект будет принят «фон».
- Особенно важно оценивать ошибки I и II рода раздельно при несбалансированности классов:
  - Пусть $P(y=+1)=0{,}01;\ P(y=-1)=0{,}99$.
  - Тогда при нулевой ошибке II рода и ошибке I рода $P(a(x)=-1 \mid y=+1)=0{,}5$
  - Общая ошибка всего лишь $P(a(x)\neq y)=0{,}005$.

Ошибки I и II рода



Некоторые метрики оценки качества решения задачи классификации

Рассмотрим бинарную классификацию, когда есть всего два класса. Основной метрикой для такой задачи является ***матрица неточностей*** (confusion matrix), у которой одно измерение отвечает за классы, посчитанные моделью, второе за истинные классы из данных, а в клетках указывается количество соответствующих (или доля) примеров. Возможно всего четыре ситуации:

- был объект основного класса и модель приняла его за объект основного класса, это верное распознавание, обозначается как TP.
- был объект вторичного класса и модель так и ответила, это тоже верное распознавание, обозначается как TN.
- был объект основного класса, но модель сказала, что это объект вторичного класса, это ошибка, обозначается как FP.
- был объект вторичного класса, но модель сказала, что был объект основного класса, это тоже ошибка, обозначается как FN.

	$y = 1$	$y = 0$
$\hat{y} = 1$	True Positive (TP)	False Positive (FP)
$\hat{y} = 0$	False Negative (FN)	True Negative (TN)

Некоторые метрики оценки качества решения задачи классификации

Матрица несет в себе всю информацию о качестве модели, в идеале элементы вне главной диагонали (ошибки) равны нулю. Но матрица сложна для понимания человеком, поэтому на основе этой матрицы считают множество других метрик.

В матрице $\hat{y}$ представляет собой ответ алгоритма на объекте, а y является истинным классом объекта.

Доля правильных ответов алгоритма среди общего количества ответов часто задается в виде меры **аккуратности**:

$$\boldsymbol{accuracy} = \frac{\mathbf{TP} + \mathbf{TN}}{\mathbf{TP} + \mathbf{TN} + \mathbf{FP} + \mathbf{FN}}.$$

Особенность данной метрики заключается в том, что она не учитывает случай с несбалансированными данными в выборке. В таком случае следует либо использовать сбалансированную выборку данных, либо изменить метрику оценки качества классификации.

Некоторые метрики оценки качества решения задачи классификации

Для оценки в пределах класса используются такие метрики, как **точность и полнота**. Точность показывает долю объектов, верно рассчитанную алгоритмом классификации среди всех, рассчитанных им как положительные:

$$\boldsymbol{precision} = \mathbf{P} = \frac{\mathbf{TP}}{\mathbf{TP} + \mathbf{FP}}$$

Полнота показывает способность алгоритма находить данный класс вообще, и описывается формулой:

$$\boldsymbol{recall} = \frac{\mathbf{TP}}{\mathbf{TP} + \mathbf{FN}}$$

Точность и полнота применимы даже в условиях несбалансированных выборок. На основании этих оценок можно рассчитать их взвешенное среднее.

Recall демонстрирует способность алгоритма обнаруживать данный класс вообще, а ***precision*** — способность отличать этот класс от других классов.

Некоторые метрики оценки качества решения задачи классификации

FPR – показывает уровень неверных позитивных ответов:

$$FPR = \frac{\mathbf{FP}}{\mathbf{FP} + \mathbf{TN}}$$

TPR – показывает уровень верных позитивных ответов:

$$TPR = \frac{\mathbf{TP}}{\mathbf{TP} + \mathbf{FN}}$$

F-мера — среднее гармоническое precision и recall.

$$F_\beta = \left(\mathbf{1} + \boldsymbol{\beta}^2\right) \frac{\mathbf{precision} * \mathbf{recall}}{\boldsymbol{\beta}^2 \mathbf{precision} + \mathbf{recall}}$$

$\boldsymbol{\beta}$ в данном случае определяет вес точности в метрике.

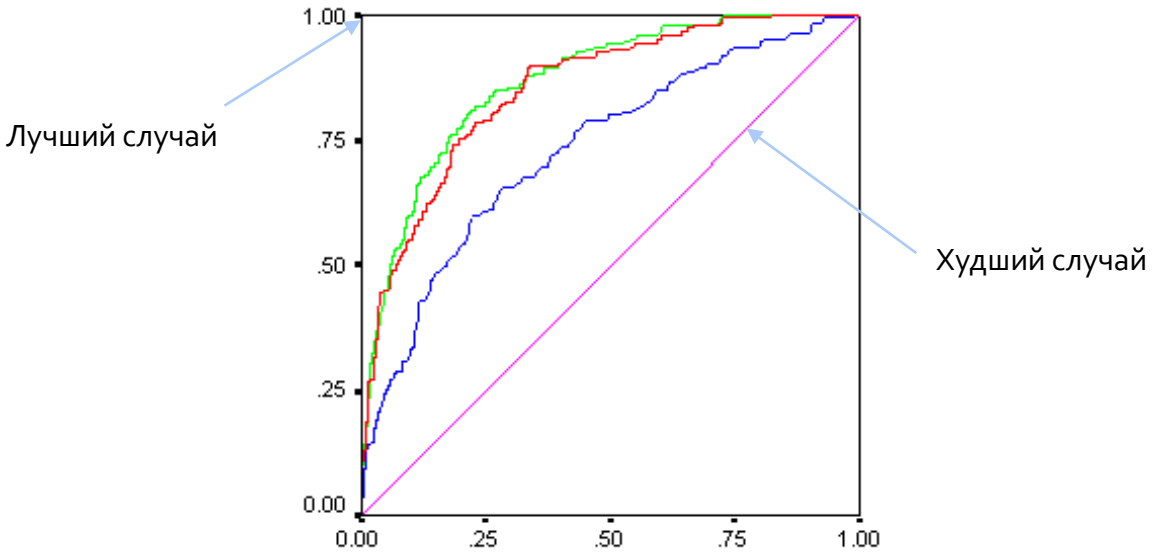
F-мера достигает максимума при полноте и точности, равными единице, и близка к нулю, если один из аргументов близок к нулю.

В случае задач с несбалансированными классами, которые превалируют в реальной практике, часто приходится прибегать к техникам искусственной модификации выборок для выравнивания соотношения классов.

ROC кривая

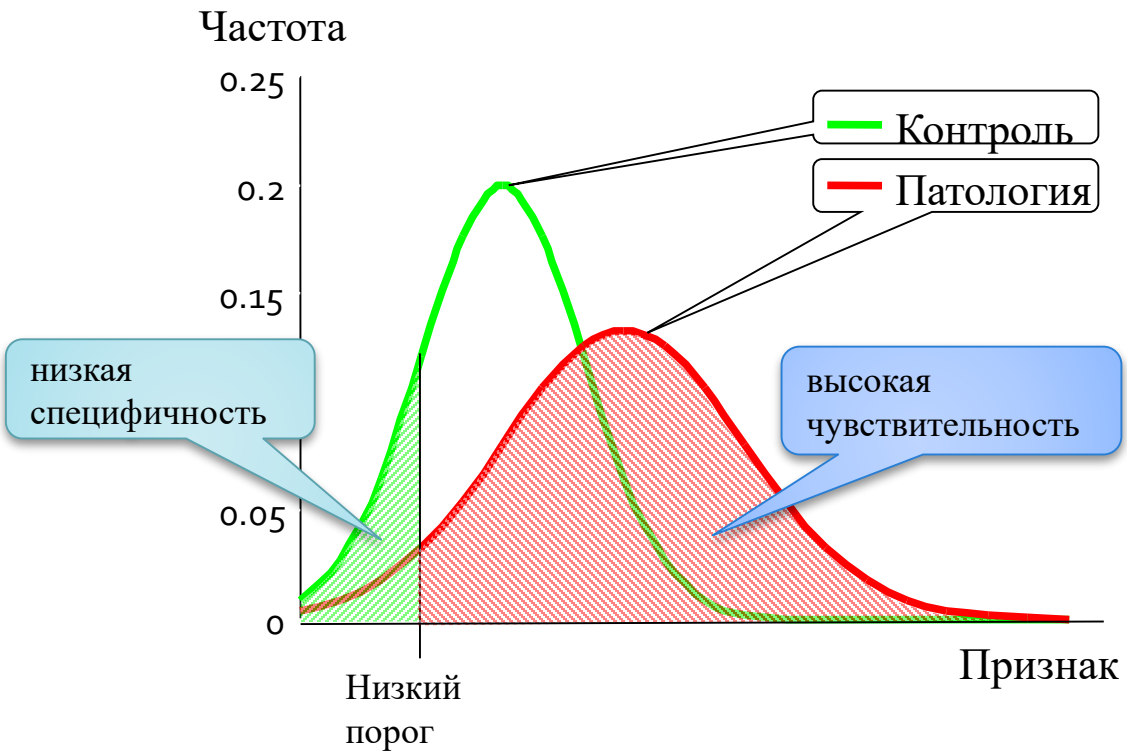
– **Receiver Operating Characteristic curve**

- Кривая, отражающая зависимость зависимость ***TPR*** от ***FPR*** при плавном изменении порога

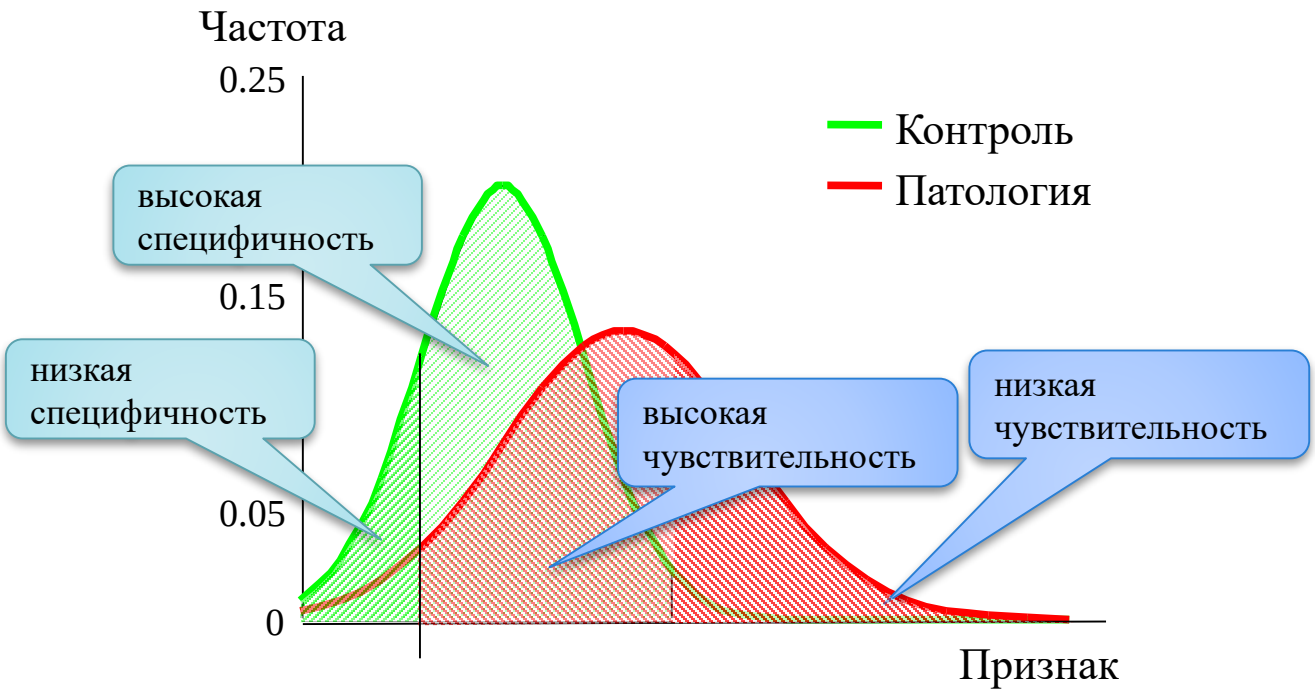


- С помощью ROC можно анализировать возможность текущего решения соответствовать требованиям

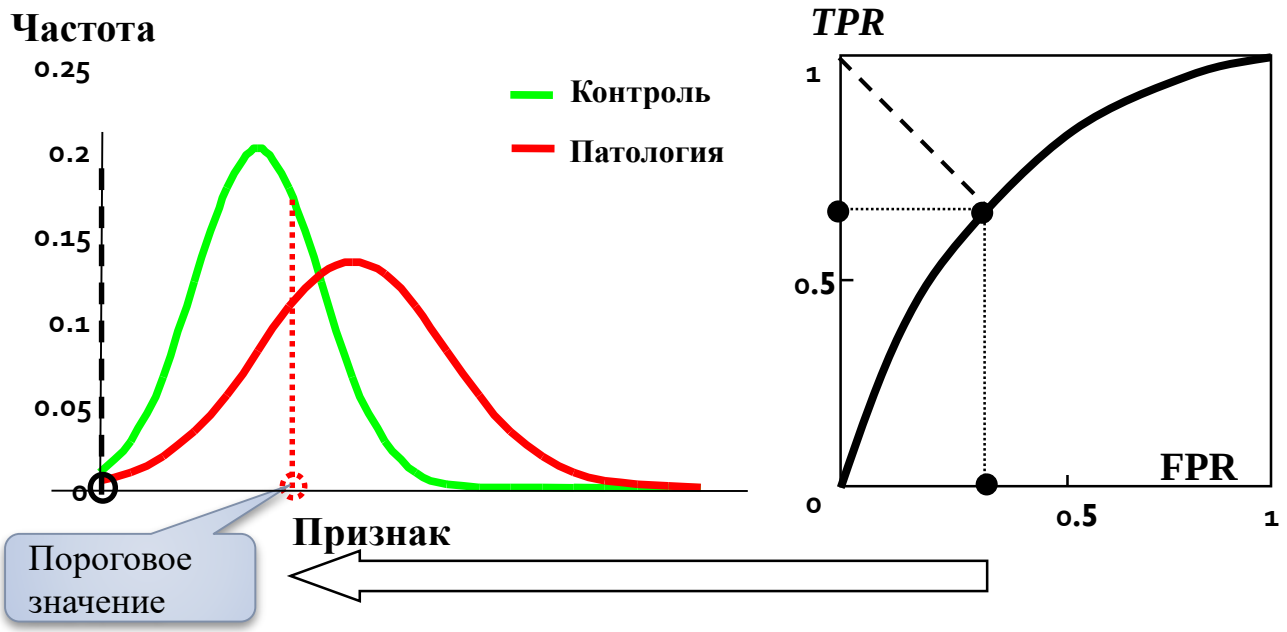
Количественный тест: выбор порога признака



Количественный тест: выбор порога признака



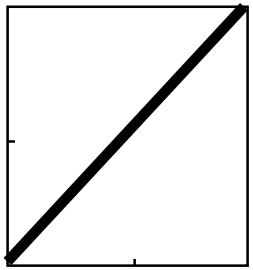
Построение ROC кривой



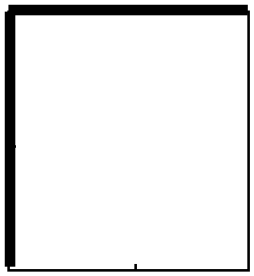
Оптимальный выбор порога:
выбирается точка на ROC-кривой, которая ближе всех к левому верхнему углу (0,1)

Форма ROC кривых

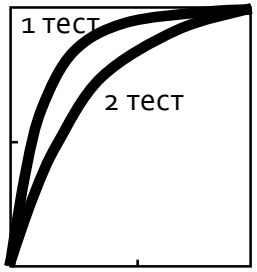
- Признак не работает



- Идеальный признак



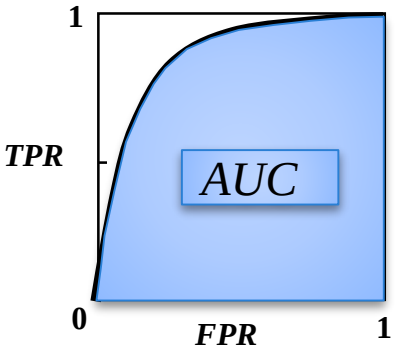
- 1-ый тест лучше, чем 2-ой



Интегральный показатель прогностической эффективности признака

AUC - это площадь под ROC-кривой (**A**rea **U**nder **C**urve)

Вероятность того, что значение признака-признака у случайно выбранного объекта относящегося к классу больше, чем у случайно выбранного не относящегося



AUC	
0,5	Случайный классификатор
0,5-0,6	Плохой классификатор
0,6-0,7	Средний классификатор
0,7-0,8	Хороший классификатор
>0.8	Отличный классификатор

Некоторые метрики оценки качества решения задачи регрессии

Средняя квадратичная ошибка ***MSE,*** один из наиболее распространённых показателей для оценки регрессионных моделей, в том числе и в задачах прогнозирования. ***MSE*** измеряет среднеквадратичную ошибку прогнозов. Для каждой точки вычисляется квадратная разница между прогнозами и целью, а затем усредняются эти значения:

$$\boldsymbol{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \bar{y}_i)^2.$$

Чем выше это значение, тем хуже модель. Ошибка никогда не бывает отрицательной, для идеальной модели это будет ноль.

Среднеквадратическая ошибка ***RMSE***, это квадратный корень из ***MSE***.

$$\boldsymbol{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \bar{y}_i)^2}$$

Некоторые метрики оценки качества решения задачи регрессии

RMSE служит для объединения величин ошибок в прогнозах для различных точек данных в единую меру прогноза. ***RMSE*** — это мера точности, позволяющая сравнивать ошибки прогнозирования различных моделей для конкретного набора данных, а не между наборами данных, поскольку она зависит от масштаба. RMSE всегда неотрицательна, и значение 0 указывает на идеальное соответствие данным.

Средняя абсолютная ошибка ***MAE*** это линейная оценка, показывающая что все индивидуальные различия взвешены одинаково в среднем

$$\boldsymbol{MAE} = \frac{\mathbf{1}}{\boldsymbol{N}} \sum_{\mathbf{i=1}}^{\boldsymbol{N}} |\boldsymbol{y_i} - \boldsymbol{\bar{y}_i}|.$$

Данная оценка является более робастной чем ***MSE***.

Некоторые метрики оценки качества решения задачи регрессии

Средняя абсолютная ошибка ***MAPE*** – это та же ошибка ***MAE*** выраженная в процентах. Данная оценка применяется для временных рядов, фактические значения которых значительно больше 1. Для каждого объекта абсолютная ошибка делится на целевое значение, что дает относительную ошибку.

$$\boldsymbol{MAPE} = \frac{1}{N}\sum_{i=1}^{N}\frac{|y_i - \bar{y}_i|}{y_i} \cdot 100\%$$

Считается, что точность модели хорошая, если среднее значение относительной погрешности не превышает 5%, удовлетворительная, если среднее значение относительной погрешности не превышает 15%, и неудовлетворительная, если среднее значение относительной погрешности больше 15%.