

Лекция 1

Введение в машинное обучение

Е. А. Соколов
ФКН ВШЭ

5 сентября 2021 г.

В последнее время стало появляться всё больше данных. Университеты знают всё про своих студентов — оценки и достижения за школьное время, выбранные курсы, результаты всех сессий, стажировки и практики, темы курсовых, карьерный путь после выпуска. Многие знают про своих клиентов банки, мобильные операторы, страховые компании, авиаперевозчики. Автомобили и самолёты порождают огромное количество данных за каждую минуту работы. Что с этим всем делать? Можно строить аналитику, следить за какими-нибудь падениями, пытаться найти закономерности в поведении пользователей — например, искать неэффективные тарифы или неприбыльные направления. Но ещё нередко хочется использовать данные, чтобы принимать решения — кому какую страховку предлагать, когда и что менять в самолёте, какие олимпиады учитывать при наборе студентов, какие страницы показывать конкретному пользователю в поисковой выдаче. Для этих задач не существует точных решений, поскольку мы слишком мало понимаем в процессах, лежащих в их основе, и не можем построить точную модель, как это делается, скажем, в физике. Но зато у нас есть много данных с примерами правильных решений! Мы уже знаем, какая страховка подошла в прошлом каждому из клиентов, и кто из студентов смог успешно завершить обучение. Так возникает машинное обучение, методы которого пытаются строить модели на основе данных, а не исходя из понимания природы. О нём и пойдёт речь в нашем курсе.

1 Основные определения и постановки задач

Машинное обучение — это наука, изучающая способы извлечения закономерностей из ограниченного количества примеров. Есть ещё много близких направлений. Например, к *анализу данных* можно отнести любую работу, связанную с извлечением полезной информации из данных, даже если это извлечение делается методом пристального взгляда. Последнее время нередко говорят про *искусственный интеллект*, но зачастую это оказывается лишь более красивым названием для результатов, полученных методами машинного обучения. Впрочем, обсуждение терминологии вряд ли нам сильно поможет, поэтому лучше перейдём к конкретному примеру.

Представим, что нам принадлежит большая сеть ресторанов, и мы хотим открыть в некоем городе новое заведение¹. Мы нашли несколько точек в городе, где есть возможность приобрести помещение и организовать там ресторан. Нам важно, чтобы через определенное время он стал приносить прибыль — точнее, хочется открыть его в той точке, в которой прибыль окажется максимальной. Поставим задачу: для каждого возможного размещения ресторана предсказать прибыль, которую он принесет в течение первого года.

Объектом мы будем называть то, для чего хотим сделать предсказание. В нашем случае это конкретная точка размещения ресторана. Обозначать объект мы будем маленькой буквой x , а если их несколько, то будем добавлять нижние индексы. Множество всех возможных точек размещения называется *пространством объектов* и обозначается через \mathbb{X} . Величина, которую мы хотим определять (т.е. прибыль ресторана), называется *ответом* или *целевой переменной*, а множество ее значений — *пространством ответов* \mathbb{Y} . В нашем случае пространство ответов является множеством вещественных чисел: $\mathbb{Y} = \mathbb{R}$. Отдельные ответы будем обозначать маленькой буквой y .

Мы не являемся специалистами в экономике, поэтому не можем сделать такие прогнозы на основе своих экспертных знаний. У нас есть лишь примеры — поскольку мы владеем целой сетью ресторанов, то имеем данные по достаточно большому числу ранее открытых ресторанов и по их прибыли в течение первого года. Каждый такой пример называется *обучающим*, а вся их совокупность — *обучающей выборкой*, которая обозначается как $X = \{(x_1, y_1), \dots, (x_\ell, y_\ell)\}$, где x_1, \dots, x_ℓ — обучающие объекты, а ℓ — их количество. Особенность обучающих объектов состоит в том, что для них известны ответы y_1, \dots, y_ℓ .

Отметим, что объекты — это некие абстрактные сущности (точки размещения ресторанов), которыми компьютеры не умеют оперировать напрямую. Для дальнейшего анализа нам понадобится описать объекты с помощью некоторого набора характеристик, которые называются *признаками* (или факторами). Вектор всех признаков объекта x называется *признаковым описанием* этого объекта. Далее мы будем отождествлять объект и его признаковое описание. Признаки могут быть очень разными: бинарными, вещественными, категориальными (принимают значения из неупорядоченного множества), ординальными (принимают значения из упорядоченного множества), множественными (set-valued, значения являются подмножествами некоторого универсального множества). Признаки могут иметь сложную внутреннюю структуру: так, в качестве признака для конкретного человека в задаче предсказания его годового дохода может служить фотография. Разумеется, фотографию можно представить и как некоторое количество бинарных или вещественных признаков, каждый из которых кодирует соответствующий пиксель изображения. Однако, работа с изображением как с одной сложной структурой позволяет вычислять по нему различные фильтры, накладывать требование инвариантности ответа к сдвигам и т.д. На работе со сложными данными специализируется активно развивающееся сейчас *глубинное обучение* (deep learning).

В нашей задаче полезными могут оказаться признаки, связанные с демографией (средний возраст жителей ближайших кварталов, динамика изменения количества жителей) или недвижимостью (например, средняя стоимость квадратного метра в

¹Задача по мотивам конкурса Restaurant Revenue Prediction: <https://www.kaggle.com/c/restaurant-revenue-prediction>

окрестности, количество школ, магазинов, заправок, торговых центров, банков поблизости). Разработка признаков (feature engineering) для любой задачи является одним из самых сложных и самых важных этапов анализа данных.

Описанная задача является примером задачи *обучения с учителем* (*supervised learning*), а более конкретно задачей *регрессии* — именно так называются задачи с вещественной целевой переменной. Перечислим несколько других видов задач обучения с учителем:

1. $\mathbb{Y} = \{0, 1\}$ — бинарная классификация. Например, мы можем предсказывать, кликнет ли пользователь по рекламному объявлению, вернет ли клиент кредит в установленный срок, сдаст ли студент сессию, случится ли определенное заболевание с пациентом (на основе, скажем, его генома).
2. $\mathbb{Y} = \{1, \dots, K\}$ — многоклассовая (multi-class) классификация. Примером может служить определение предметной области для научной статьи (математика, биология, психология и т.д.).
3. $\mathbb{Y} = \{0, 1\}^K$ — многоклассовая классификация с пересекающимися классами (multi-label classification). Примером может служить задача автоматического проставления тегов для ресторанов (логично, что ресторан может одновременно иметь несколько тегов).
4. Частичное обучение (semi-supervised learning) — задача, в которой для одной части объектов обучающей выборки известны и признаки, и ответы, а для другой только признаки. Такие ситуации возникают, например, в медицинских задачах, где получение ответа является крайне сложным (например, требует проведения дорогостоящего анализа).

Существует также обучение без учителя — класс задач, где ответы неизвестны или вообще не существуют, и требуется найти некоторые закономерности в данных лишь на основе признаковых описаний:

1. Кластеризация — задача разделения объектов на группы, обладающие некоторыми свойствами. Примером может служить кластеризация документов из электронной библиотеки или кластеризация абонентов мобильного оператора.
2. Оценивание плотности — задача приближения распределения объектов. Примером может служить задача обнаружения аномалий, в которой на этапе обучения известны лишь примеры «правильного» поведения оборудования (или, скажем, игроков на бирже), а в дальнейшем требуется обнаруживать случаи некорректной работы (соответственно, незаконного поведения игроков). В таких задачах сначала оценивается распределение «правильных» объектов, а затем аномальными объявляются все объекты, которых в рамках этого распределения получают слишком низкую вероятность.
3. Визуализация — задача изображения многомерных объектов в двумерном или трехмерном пространстве таким образом, чтобы сохранялось как можно больше зависимостей и отношений между ними.

4. Понижение размерности — задача генерации таких новых признаков, что их меньше, чем исходных, но при этом с их помощью задача решается не хуже (или с небольшими потерями качества, или лучше — зависит от постановки). К этой же категории относится задача построения латентных моделей, где требуется описать процесс генерации данных с помощью некоторого (как правило, небольшого) набора скрытых переменных. Примерами являются задачи тематического моделирования и построения рекомендаций, которым будет посвящена часть курса.

Бывают и более сложные постановки — например, обучение с подкреплением (reinforcement learning), где алгоритм на каждом шаге наблюдает какую-то ситуацию, выбирает одно из доступных ему действий, получает некоторую награду и корректирует свою стратегию. Задачей алгоритма при этом является максимизация награды в некотором смысле. В такую постановку хорошо вписывается, например, задача создания беспилотного автомобиля: машина «видит» текущее окружение за счёт сенсоров и должна решить, как сейчас повернуть руль, как сильно ускориться или затормозить. При этом она должна приехать в конкретную точку, не нарушая правила, и согласно этим требованиям в каждый момент она получает некоторую награду — скажем, если она превысила допустимую скорость и отделилась от точки назначения, то награда будет очень низкой. Впрочем, задача обучения с подкреплением очень непростая, и пока что построение беспилотных автомобилей исключительно на таком подходе — скорее исследовательский вопрос.

Вернемся к нашей задаче по предсказанию прибыли ресторана. Предположим, что мы собрали обучающую выборку и изобрели некоторое количество признаков. Результатом будет матрица «объекты-признаки» $X \in \mathbb{R}^{\ell \times d}$ (ℓ — число объектов, d — число признаков), в которой каждая строка содержит признаковое описание одного из обучающих объектов. Таким образом, строки в этой матрице соответствуют объектам, а столбцы — признакам. Отметим, что здесь имеет место небольшая терминологическая путаница: буквой X мы обозначаем и обучающую выборку (которая содержит в себе объекты и ответы), и матрицу «объекты-признаки» (которая содержит в себе только объекты). Тем не менее, из контекста всегда будет ясно, о чём именно идёт речь.

Нашей задачей является построение функции $a : \mathbb{X} \rightarrow \mathbb{Y}$, которая для любого объекта будет предсказывать ответ. Такая функция называется *алгоритмом* или *моделью*. Понятно, что нам подойдет далеко не каждый алгоритм — например, вряд ли мы извлечем какую-то выгоду из алгоритма $a(x) = 0$, который предсказывает нулевую прибыль для любого ресторана независимо от его признаков. Чтобы формализовать соответствие алгоритма нашим ожиданиям, нужно ввести *функционал качества*², измеряющий качество работы алгоритма. Если функционал устроен так, что его следует минимизировать, то логичнее называть его *функционалом ошибки*. Крайне популярным функционалом в задаче регрессии является среднеквадратичная ошибка (mean squared error, MSE):

$$Q(a, X) = \frac{1}{\ell} \sum_{i=1}^{\ell} (a(x_i) - y_i)^2.$$

²Достаточно часто употребляется термин «метрика качества», но он является не очень удачным, поскольку вызывает лишние ассоциации с метриками, обсуждаемыми в алгебре. Тем не менее, мы иногда будем использовать этот термин для разнообразия.

Чем более маленькое значение этого функционала дает алгоритм, тем он лучше. Данный функционал является очень удобным благодаря дифференцируемости и простоте, но, как мы увидим дальше в курсе, обладает и большим количеством недостатков.

В большинстве случаев функционалы представляют собой сумму или среднее значение ошибок на отдельных объектах обучающей выборки. Функция, измеряющая ошибку одного предсказания, называется *функцией потерь* $L : \mathbb{Y} \times \mathbb{Y} \rightarrow \mathbb{R}_+$ (в нашем случае $L(y, z) = (y - z)^2$). Аргументами функции потерь являются правильный ответ на данном объекте и прогноз модели на нём же.

Заметим, что именно функционал ошибки будет определять во всех дальнейших рассуждениях, какой алгоритм является лучшим. Если функционал выбран неудачно и не соответствует бизнес-требованиям или особенностям данных, то все дальнейшие действия обречены на провал. Именно поэтому выбор функционала является крайне важным этапом в решении любой задачи машинного обучения. Он не обязательно должен обладать хорошими математическими свойствами (непрерывность, выпуклость, дифференцируемость и т.д.), но обязан отражать все важные требования к решению задачи.

Как только функционал ошибки зафиксирован, можно приступать к построению алгоритма $a(x)$. Как правило, для этого фиксируют некоторое *семейство алгоритмов* \mathcal{A} , и пытаются выбрать из него алгоритм, наилучший с точки зрения функционала³. В машинном обучении было изобретено большое количество семейств алгоритмов, и, наверное, самым простым и самым тщательно изученным среди них является семейство *линейных моделей*, которые дают предсказание, равное линейной комбинации признаков (с добавлением свободного коэффициента w_0):

$$\mathcal{A} = \{a(x) = w_0 + w_1x_1 + \dots + w_dx_d \mid w_0, w_1, \dots, w_d \in \mathbb{R}\},$$

где через x_i обозначается значение i -го признака у объекта x . Лучшая из таких моделей может быть выбрана, например, путем минимизации MSE:

$$\frac{1}{\ell} \sum_{i=1}^{\ell} \left(w_0 + \sum_{j=1}^d w_j x_{ij} - y_i \right)^2 \rightarrow \min_{w_0, w_1, \dots, w_d}.$$

Через x_{ij} здесь обозначается значение j -го признака на i -м объекте. Процесс поиска оптимального алгоритма называется *обучением*. Если модель $a(x)$ дифференцируема по параметрам w , то можно искать лучший набор параметров с помощью градиентных методов — стартовать из случайной точки и двигать параметры в сторону наискорейшего убывания функционала ошибки (то есть в сторону антиградиента). Для выпуклых функционалов такой метод найдёт глобальный минимум; для невыпуклых функционалов есть лишь гарантии сходимости к локальному минимуму. При этом может возникнуть вопрос о том, как добиться сходимости к лучшему из локальных минимумов, и подходить к нему можно по-разному — например, с помощью выбора грамотного начального приближения или через выбор более сложного метода оптимизации. В то же время дифференцируемые модели, поддающиеся оптимизации, по сути, представляют собой последовательность несложных преобразований

³ «Пытаются выбрать» — потому что функционал может оказаться слишком сложным, не позволяющим сделать точный поиск глобального минимума. В этом случае часто ограничиваются поиском локального экстремума.

данных, и не факт, что такие модели смогут заменить сложные структуры данных или программы с ветвлениями и циклами. На сегодняшний день существует не так много видов недифференцируемых моделей, поддающихся эффективному обучению, но среди них есть очень успешные примеры — например, градиентный бустинг над решающими деревьями.

Зачастую возникает потребность в *предобработке данных* до начала построения модели. Здесь может идти речь о некотором ряде манипуляций:

- Некоторые модели хорошо работают только при выполнении определенных требований. Так, при обучении линейных моделей важно, чтобы признаки были *масштабированными*, то есть измерялись в одной шкале. Примером способа нормировки данных является вычитание среднего и деление на дисперсию каждого столбца в матрице «объекты-признаки».
- Бывает, что в выборку попадают *выбросы* — объекты, которые не являются корректными примерами из-за неправильно посчитанных признаков, ошибки сбора данных или чего-то еще. Их наличие может сильно испортить модель.
- Некоторые признаки могут оказаться *шумовыми*, то есть не имеющими никакого отношения к целевой переменной и к решаемой задаче. Примером, скорее всего, может служить признак «фаза луны в день первого экзамена» в задаче предсказания успешности прохождения сессии студентом.

Как показывает практика, простейшая предобработка данных может радикально улучшить качество итоговой модели.

Во время обучения модели очень важно следить за тем, чтобы не произошло *переобучения* (overfitting). Разберем это явление на примере. Допустим, что мы выбрали очень богатое семейство алгоритмов, состоящее из всех возможных функций: $\mathcal{A} = \{a : \mathcal{X} \rightarrow \mathcal{Y}\}$. В этом семействе всегда будет алгоритм, не допускающий ни одной ошибки на обучающей выборке, который просто запоминает ее:

$$a(x) = \begin{cases} y_i, & \text{если } x = x_i, \\ 0, & \text{если } x \notin X. \end{cases}$$

Очевидно, что такой алгоритм нас не устраивает, поскольку для любого нового ресторана предскажет нулевую прибыль. Алгоритм оказался *переобученным* — он слишком сильно подогнался под обучающую выборку, не выявив никаких закономерностей в ней. Существует ряд методов, направленных на борьбу с переобучением, которые мы будем обсуждать на следующих занятиях. Впрочем, одну из идей мы можем обсудить уже сейчас. В нашем примере переобучение возникло из-за большой сложности семейства — алгоритмом могла оказаться любая функция. Очевидно, что если бы мы ограничили себя только линейными моделями, то итоговый алгоритм уже не смог бы запомнить всю выборку. Таким образом, можно бороться с переобучением путем *контроля сложности семейства алгоритмов* — чем меньше у нас данных для обучения, тем более простые семейства следует выбирать.

После того, как модель построена, нам нужно оценить, насколько хорошо она будет работать на новых данных. Для этого, например, можно в самом начале отложить часть обучающих объектов и не использовать их при построении модели. Тогда можно будет измерить качество готовой модели на этой *отложенной выборке*,

получив тем самым оценку того, насколько она готова к работе на новых данных. Существуют и более сложный класс методов, называемый кросс-валидацией, о котором речь пойдет позже.

Итак, перечислим основные этапы решения задачи машинного обучения:

1. Постановка задачи;
2. Выделение признаков;
3. Формирование выборки;
4. Выбор функционала ошибки;
5. Предобработка данных;
6. Построение модели;
7. Оценивание качества модели.

На самом деле, этим всё не заканчивается, и дальше построенную модель надо *внедрить*. Например, мы делали модель, предсказывающую для каждого клиента мобильного оператора, уйдёт ли он к конкурентам в ближайший месяц. Когда модель готова, надо встроить её в инфраструктуру, чтобы она автоматически применялась для каждого клиента, информация по потенциально уходящим клиентам сама приходила куда надо и сразу же использовалась для удержания. А ещё наверняка будут определённые требования к быстродействию этой модели — и этого тоже надо будет добиться. Это большое направление, но оно выходит за рамки нашего курса.

2 Немного примеров

Очевидно, что машинное обучение не следует применять везде, где требуется построить зависимость одной переменной от набора других. Например, нет смысла восстанавливать зависимость силы от массы и ускорения — на эту задачу точный ответ даёт второй закон Ньютона. Аналогично, нет смысла строить модель, которая выбирает сортирующую перестановку для массива чисел — это можно быстро и точно сделать с помощью, например, сортировки слиянием.

Задача предсказания прибыли ресторана, которую мы разобрали, хорошо подходит для решения методами машинного обучения по ряду причин:

- Нельзя вывести корректную формулу будущей прибыли из каких-либо знаний об устройстве мира.
- Предсказываемая прибыль действительно зависит от признаков, но при этом вид зависимости достаточно сложный, и подобрать его вручную может быть слишком трудно.
- Можно набрать достаточное количество примеров (т.е. объектов с известными ответами), по которым можно оценить зависимость целевой переменной от признаков.

Задачи такого типа встречаются часто, и машинное обучение используется в них достаточно широко — собираются большие объёмы данных, формируются признаки, которые потенциально могут влиять на целевую переменную, и затем автоматически строится предсказывающая модель. К этому классу задач относятся предсказание дефолта (вернёт ли клиент кредит), предсказание оттока (уйдёт ли клиент к конкуренту), предсказание спроса на товары, построение рекомендаций (какой товар может заинтересовать пользователя).

Машинное обучение может использоваться и по другим причинам. Рассмотрим ещё несколько типов задач:

1. Бывает, что зависимость ответа от признаков вполне понятна, но при этом само построение ответа достаточно трудоёмко. Например, фермерам приходится классифицировать огурцы по качеству⁴. Критерии попадания в каждый класс можно в той или иной степени формализовать, но при этом проверять их для каждого огурца получается не очень быстро. Методами машинного обучения можно построить модель, которая будет автоматически извлекать информативные факторы из фотографии и определять класс. Если качество классификации будет высоким, то такая модель позволит существенно повысить эффективность работы.
2. Алгоритм решения некоторых задач невозможно записать, но при этом люди успешно справляются с этими задачами при наличии опыта. Примером может служить управление процессом выплавки стали (опытные инженеры делают это практически без ошибок, но при этом не существует формального алгоритма) или игра в го. Определять наиболее подходящий ход в го с помощью перебора невозможно, но лучшие игроки могут очень неплохо оценить выгоду того или иного хода. С помощью машинного обучения можно по примерам матчей сформировать модель, оценивающую ходы — как показал пример алгоритма AlphaGo, такой подход может давать достаточно неплохие результаты.

⁴<https://cloud.google.com/blog/big-data/2016/08/how-a-japanese-cucumber-farmer-is-using-deep-learning-and-tensorflow>