Глава 1.Общие сведения и понятия машинного обучения

В этой главе мы обсудим с вами основные понятия, связанные с машинным обучением, чтобы в следующих главах общаться с вами на одном языке. Мы обсудим с вами типы данных, которые обычно обрабатываются, типовые задачи машинного обучения, а также минимальные знания линейной алгебры и математического анализа, которые необходимы для освоения данного пособия.

Ключевые понятия стоит обсудить с конкретного примера данных. На Рис. 1-1 представлены данные об успеваемости некоторой совокупности студентов.

id Студента	Пол	Возраст	Институт	Общежитие	Работа	Оценка Python	ЕГЭ Инф.	Быллы по МО	Экзамен по МО
0	Ж	24	ФТИ	нет	нет	75	83	54	Отл.
1	М	23	Другой	нет	нет	79	40	98	Уд.
2	М	24	Другой	нет	да	43	59	43	Уд.
3	Ж	24	ИРИТ-РТФ	нет	да	98	83	46	Отл.
4	М	24	ИРИТ-РТФ	да	да	50	65	49	Неуд.
5	М	24	ФТИ	да	да	45	96	90	Неуд.
6	Ж	23	ИРИТ-РТФ	нет	нет	71	98	50	Хор.
7	Ж	23	ИРИТ-РТФ	нет	нет	98	43	55	Уд.
8	Ж	24	ИРИТ-РТФ	да	да	49	61	83	Неуд.
9	Ж	24	ИЕНиМ	да	да	63	46	71	Хор.

Рис. 1-1 Пример данных успеваемости студентов

Первые ключевые понятия и обозначения, которые стоит ввести:

- x объект, требующий некоторого предсказания (на Рис. 1-1 это строка для отдельного студента, характеризуемого своим id);
- $\mathbb{X}$ полный набор объектов (на Рис. 1-1 это совокупность студентов, все строки);
- y цель (target), которая является ожидаемым предсказанием (на Рис. 1-1 две целевые переменные для каждого студента, баллы по курсу Машинное обучение и оценка за Экзамен по машинному обучению);
- $\mathbb{Y}$ полный набор целей (на Рис. 1-1 целевые переменные для совокупности студентов, оба столбца);

- *Признаки* (англ. *features*) что-то, что описывает объекты (на Рис. 1-1 это пол, возраст, институт, общежитие, работа, Баллы ЕГЭ по информатике и Оценка за курс по Python).

В связи с этим, на высоком уровне типичная задача машинного обучения формулируется следующим образом: искать возможности получать целевые переменные при использовании некого признакового пространства данных.

Типы данных

К слову, о признаках и данных. Существуют различные подходы к классификации признаков, назовем их **микроуровень** и **макроуровень**.

На **микроуровне** признаки можно разделить на **Числовые** и **Категориальные**.

Числовые признаки, это некоторые количественные оценки объектов. Числовые признаки делят на **дискретные**, которые «невозможно измерить, но можно посчитать». Например, ученики в классе, пальцы, результат в футболе. Также выделяют **непрерывные** данные, которые «не могут быть подсчитаны, но их можно измерить». Например, температура, напряжение, высота. Можете отложить на время учебное пособие и придумать другие примеры дискретных и непрерывных числовых признаков.

Для числовых данных используются следующие обозначения:

- $\mathbb{N}$ натуральные числа;
- $\mathbb{Z}$ целые числа;
- $\mathbb{Q}$ рациональные числа;
- $\mathbb{R}$ действительные числа;
- $\mathbb{C}$ комплексные числа.

Категориальные признаки, это характеристики объектов. Обычно категориальные признаки делят на **Номинальные (nominal)** признаки, которые отвечают на вопрос о том какое значение принимает данная характеристика, например, цвет, пол, язык, институт и т. д.; и **Порядковые данные (ordinal)** признаки, дискретные и упорядоченные величины, например, уровень английского, итоговая оценка за курс машинного обучения. **Номинальные** признаки, в которых всего два возможных значения называют **бинарными** (это ответы на такие вопросы, когда ожидается ответ «да» или «нет»).

На практике **категориальные** данные могут быть представлены в виде числовых значений, как это представлено на Рис. 1-2.

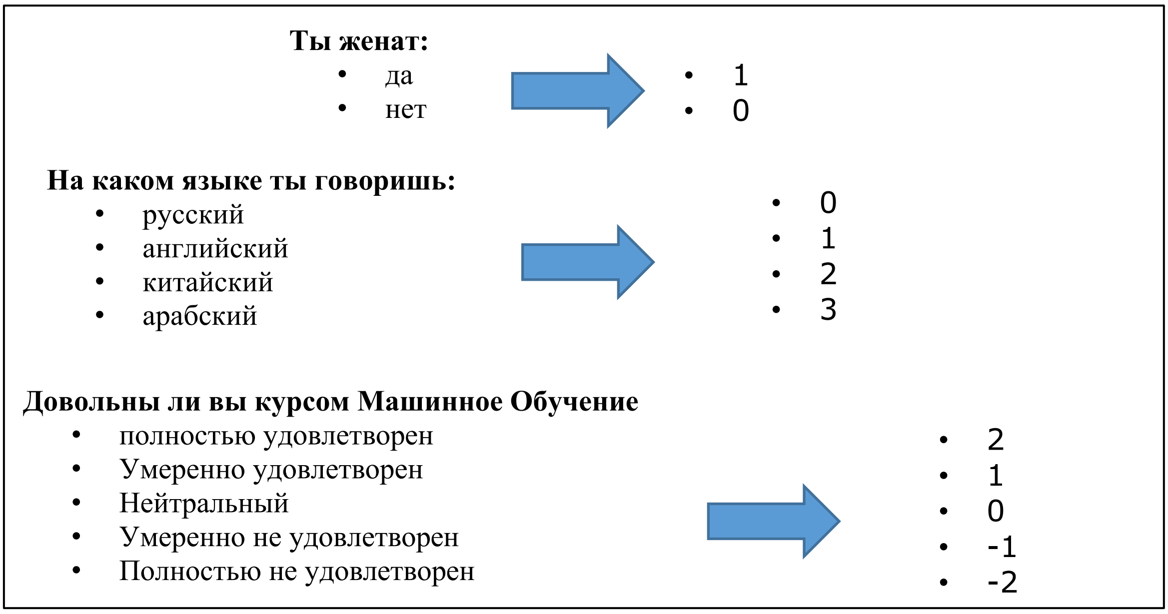

Рис. 1-2 Пример представления категориальных данных в виде числовых данных

Но нужно помнить, что подобные числа не имеют математического значения, как случае числовых признаков. Т. е. их нельзя складывать (например, нельзя сложить английский и китайский и получить арабский), или сравнивать между собой (английский не в три раза «меньше» чем арабский). Это накладывает определенные ограничения на использование категориальных признаков в моделях машинного обучения.

На **макроуровне** признаки можно разделить на **Табличные** и **другие**. **Табличные** – это данные как на Рис. 1-1, представленные в виде таблице представляющих совокупность различных числовых и категориальных признаков. По строкам представлены различные объекты, по столбцам –

различные признаки. Именно этот тип данных в основном используется, когда речь идет о классических алгоритмах машинного обучения.

Помимо этого, существуют изображения, временные ряды и естественный язык. Каждый из этих типов данных имеет свои особенности, которые требуют специального подхода. При этом для получения каких-то простых моделей эти данные можно свести к табличным.

Например, можно рассматривать изображение как совокупность интенсивности отдельных пикселей (Рис. 1-3). Принято, что большие значения интенсивности соответствуют белому цвету, а небольшие – черному. При этом в зависимости от разрядности изображения можно иметь разные значения «оттенков серого» между белым и черным. Так на Рис. 1-3 представлено 8-битное изображение (где интенсивность цвета кодируется значением от 0 до 255). Любое изображение можно «спрямить» (flatten), т. е. перейти от двумерного представления к одномерному. Таким образом каждое изображение можно представить в виде большой строки признаков. Например, изображение 10 на 10 пикселов представляется в виде строки из 100 признаков.

241	251	248	244	252	247	254	254	241	240
244	249	252	246	9	2	240	249	251	252
241	244	254	251	250	8	244	250	254	241
246	251	250	243	249	14	245	246	254	252
244	251	246	244	240	1	244	242	241	249
247	241	244	253	244	15	250	248	249	245
245	252	250	245	248	8	249	250	243	247
254	241	240	249	254	13	247	240	248	242
245	247	241	248	248	1	247	249	252	245
243	244	251	242	248	247	248	251	249	246

241	251	248	244	252	247	254	254	241	240	244	249	252	246	9	2	240	249	251	252

Рис. 1-3 Представление одноканального изображения в виде строки признаков

Однако нужно помнить, что реальные изображения состоят из нескольких цветовых каналов, а не одноканальные как на Рис. 1-3. Наиболее распространена трехканальная цветовая модель RGB – Red, Красный; Green, Зеленый; Blue, Синий. На Рис. 1-4 представлен пример разложения, пожалуй, самого известного изображения в Компьютерном Зрении – Лена – на три канала. Таким образом, изображение Лены можно представить в виде строки из $512 \times 512 \times 3 = 786\,432$ признаков.

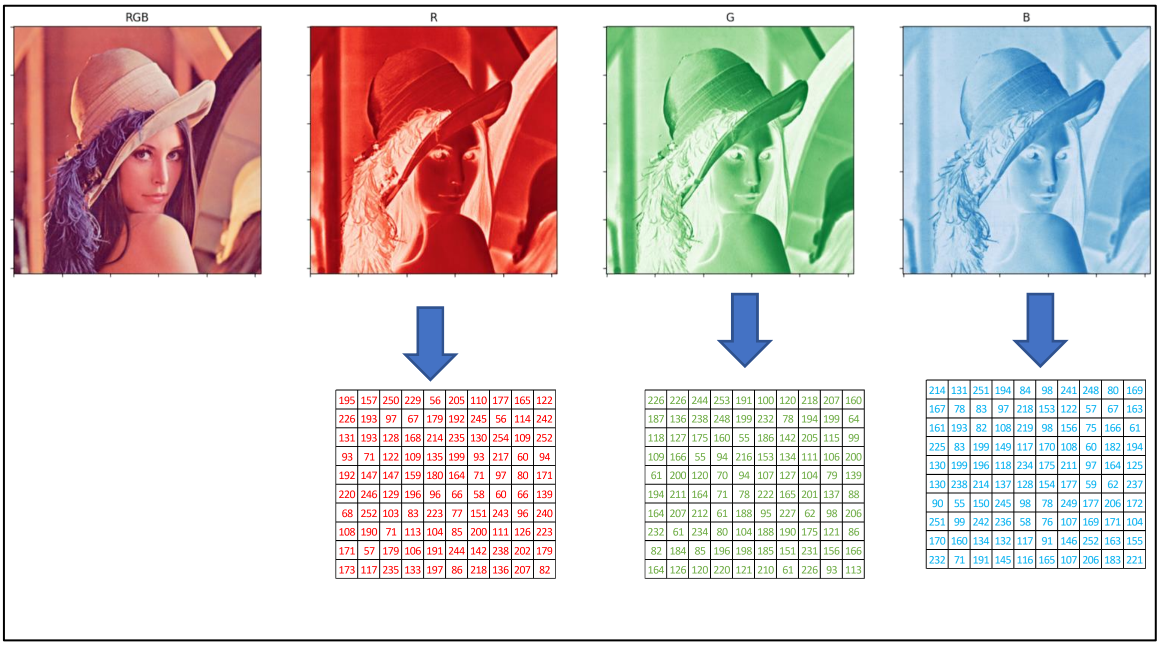

Рис. 1-4 Разложение цветного изображения Лена на каналы [1]

Понятно, что такое представление изображение в виде табличных данных не кажется чем-то эффективным. При этом мы должны быть уверены, что в каждом пикселе находится «однотипная» часть изображения: например, если это изображения котов и собак, то для такого подхода они должны быть сняты под одним углом. Поэтому представление изображений в виде таблиц в основном используется только для однотипных данных (например, изображения цифр). А для обработки более сложных изображений в настоящее время самым распространенным подходом являются модели, использующие Свёрточные Нейронные Сети (Convolutional Neural Networks) [2–4].

В какой-то степени аналогичным способом можно поступать с временными рядами. Временной ряд представляет собой совокупность значений какого-либо измерения в отдельные моменты времени. С одной стороны, можно свести временной ряд к табличным данным. Однако такой подход наивен, если планируется анализировать сигналы разных объектов – мы должны быть уверены в том, что все сигналы «согласованы» по оси времени. Однако такое редко встречается для реальных сигналов. Например, сигнал электрокардиографии (Рис. 1-5) имеет достаточно сложную структуру. При этом информативным является расстояние между последовательными R-зубцами.

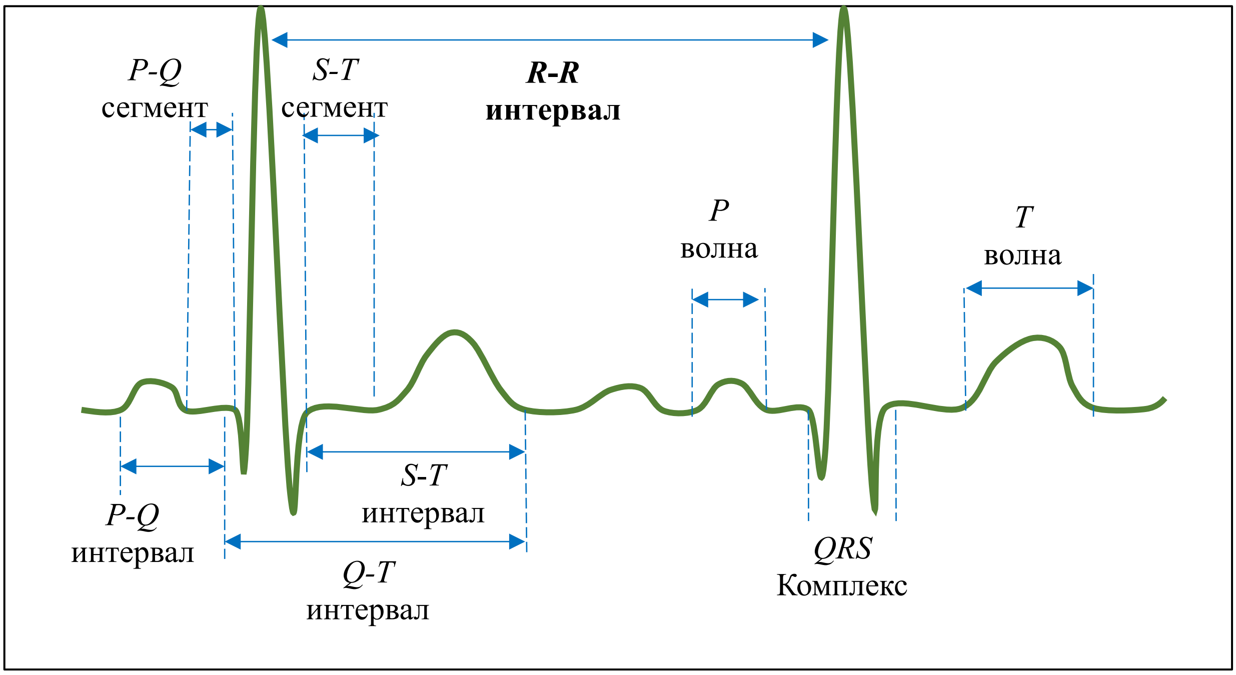

Рис. 1-5 Визуализация сигнала Электрокардиограммы

В этом ключе продуктивным подходом к обработке временных сигналов является расчет признакового пространства. Так подробно обработка биомедицинских сигналов и извлечение признакового пространства освещена в следующих учебных пособиях [5; 6].

С другой стороны, возможно использовать модели, которые учитывают временную составляющую временных рядов. К таким подходам относятся модели, использующие Рекуррентные Нейронные Сети (Recurrent Neural Network) [7].

В упрощенном виде данные естественного языка можно рассматривать как совокупность отдельных слов – категориальных признаков. Такие подходы вполне применимы в обобщенных задачах, связанных с оценкой тональности текстов. Другим подходом к решению задач обработки естественного языка является получение векторных представлений слов (Word Embedding) [8; 9]. Однако для более сложных задач, таких как перевод с одного языка на другой, генерация текста используют т. н. модели Трансформеры (Transformers) [9], которые используют механизм Внимание (Attention) [10].

Обучение модели

В предыдущем разделе мы часто употребляли понятие модель. В общем случае **модель** использует некие операции над признаками для предсказания целевой переменной, т. е. происходит отображение из пространства признаков в пространство целевых предсказаний:

$$a : \mathbb{X} \to \mathbb{Y},$$

где $a \in \mathbb{A}$ семейство Моделей.

Обычно предсказания модели обозначают символом $\hat{y}$ (игрек с «крышечкой»). В общем случае справедливо следующее выражение

$$y_i = a(\boldsymbol{x_i}, w, h),$$

где $\boldsymbol{x}_i$ – вектор признаков для *i*-ого объекта,

w параметры Модели (оптимизируются алгоритмом модели),

h гиперпараметры модели (оптимизируются нами теми кто запускает алгоритмы машинного обучения).

Более подробно о параметрах и гиперпараметрах моделей мы обсудим в последующих главах.

После того, как мы выбрали модель ее необходимо обучить. Следующая концепция, которую стоит обсудить это **Тренировочная** и **Тестовая**

выборка. **Тренировочная выборка** – это такой набор данных, для которого нам известны пары «признаки» – «целевая переменная» для каждого субъекта из выборки. **Тестовая выборка** – это такой набор данных, для которого известны только признаки.

Например, (Рис. 1-6), у нас есть данные по студентам набора 2021 года, по которым мы знаем все собираемые параметры и целевые переменные. И к нам в 2022 году поступили новые студенты. И мы хотим, используя опыт предыдущего года попытаться предсказать, студентов у которых могут быть проблемы с курсом машинное обучение, и наоборот, студенты, которые успешно справятся и им лучше выдавать задачки посложнее.

Тренировочная выборка

id Студента	Пол	Возраст	Институт	Общежитие	Работа	Оценка Python	ЕГЭ Инф.	Быллы по МО	Экзамен по МО
0	Ж	24	ФТИ	нет	нет	75	83	54	Отл.
1	М	23	Другой	нет	нет	79	40	98	Уд.
2	М	24	Другой	нет	да	43	59	43	Уд.
3	Ж	24	ИРИТ-РТФ	нет	да	98	83	46	Отл.
4	М	24	ИРИТ-РТФ	да	да	50	65	49	Неуд.
5	М	24	ФТИ	да	да	45	96	90	Неуд.
6	Ж	23	ИРИТ-РТФ	нет	нет	71	98	50	Хор.
7	Ж	23	ИРИТ-РТФ	нет	нет	98	43	55	Уд.
8	Ж	24	ИРИТ-РТФ	да	да	49	61	83	Неуд.
9	Ж	24	ИЕНиМ	да	да	63	46	71	Хор.

Тестовая выборка

id Студента	Пол	Возраст	Институт	Общежитие	Работа	Оценка Python	ЕГЭ Инф.	Быллы по МО	Экзамен по МО
10	М	23	ИЕНиМ	да	да	51	84	?	?
11	Ж	24	ИЕНиМ	да	нет	90	44	?	?
12	Ж	24	ФТИ	да	нет	60	72	?	?
13	Ж	24	ИЕНиМ	да	нет	55	76	?	?
14	Ж	23	ФТИ	нет	да	40	54	?	?
15	М	24	Другой	нет	да	94	64	?	?
16	М	23	Другой	да	да	65	46	?	?
17	М	24	Другой	нет	нет	50	49	?	?
18	Ж	23	ИРИТ-РТФ	да	да	62	93	?	?
19	М	23	ИЕНиМ	да	да	56	40	?	?

Рис. 1-6 Тренировочная и тестовая выборка

Для того чтобы оценить, насколько плоха или хороша конкретная модель нам необходимо воспользоваться **функцией потерь** $L\left(y_i, y_i\right)$, для оценки качества модели. В зависимости от типа целевой переменной (числовая или категориальная) функции потери могут отличаться, но в них есть общая идеология. Если для конкретного объекта $y_i \sim y_i$, т. е. предсказание модели и реальное значения целевой переменной совпадают, тогда функция потерь $L(y_i, \hat{y_i})$ принимает небольшие значения, иначе, если предсказание модели разнятся, то функция потерь принимает большие значения.

Используя данные Тренировочной выборки, мы можем оценить **функционал потерь**, среднее значения функции потерь для всех объектов из тренировочной выборки $Q(a,\mathbb{X}) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, a(x_i, w, h))$.

Таким образом можно сформулировать цель обучения следующим образом: $Q(a,\mathbb{X}) \to \min\limits_{a\in\mathbb{A}}$. Другими словами, в ходе обучения мы должны подобрать такие параметры и гиперпараметры модели, которые наилучшим образом предсказывают целевые значения в тренировочной выборке.

В общем случае наблюдается закономерность: простые модели (которые содержат ограниченное число признаков, простые зависимости между переменными) обладают большими значениями функционала потерь, в то же время сложные модели (в которых много признаков и существуют сложные зависимости между переменными) могут иметь сколь угодно низкие значения функционала потерь. В общем случае зависимость функционала потерь от сложности модели представлена на Рис. 1-7.

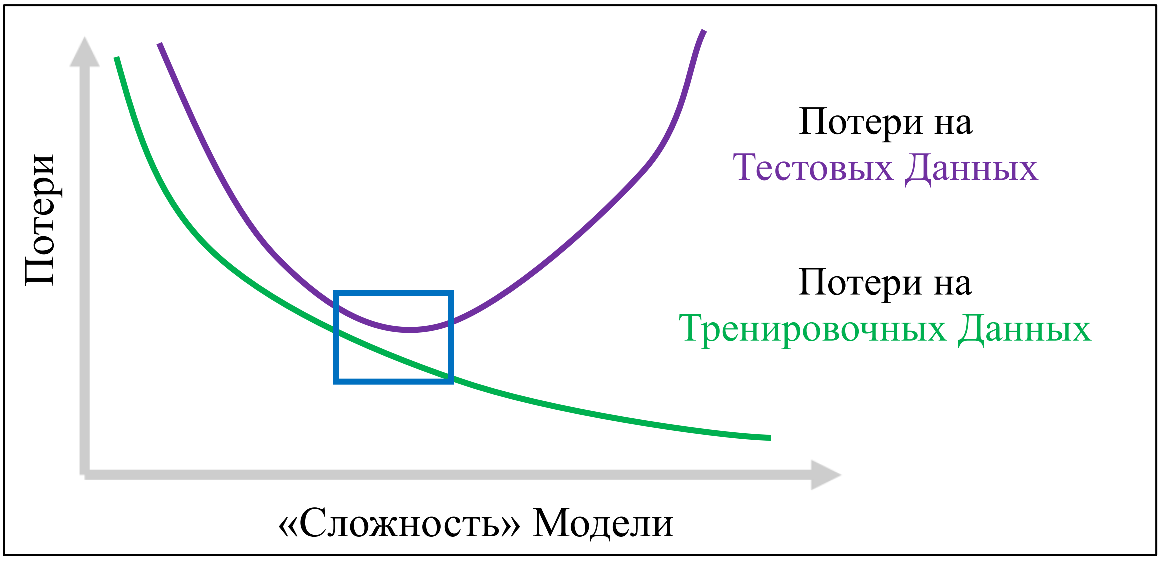

Рис. 1-7 Зависимость потерь от сложности моделей

При усложнении модели потери на тестовых данных как правило тоже в начале убывают. Но в определенный момент может возникнуть ситуация, когда потери на тестовой выборке начинают снова расти, а потери на тренировочной выборке продолжают падать. Это явление называется **переобучением**. Мы не находим общие закономерности, а запоминаем тренировочную выборку. Здесь стоит отметить, что практической пользы, от модели, которая просто «хорошо запоминает тренировочную выборку» немного. Поэтому обычно потери на тренировочных данных рассматривают совместно с потерями на тестовых данных.

Чтобы избежать переобучения мы должны использовать тестовую выборку для своевременной остановки алгоритма обучения и выбора оптимальных параметров и гиперпараметров модели. Но есть проблема: по умолчанию у нас нет значений целевых переменных для тестовой выборки.

Однако мы можем смоделировать тестовую выборку, используя подход, который называется **валидация**. Мы можем взять «кусочек» тренировочной выборки, отложить его, обучить модель на остальной тренировочной выборке и проверить на отложенном «кусочке». Такой подход называется использование **отложенной выборки** (Hold-Out split) Рис. 1-8.

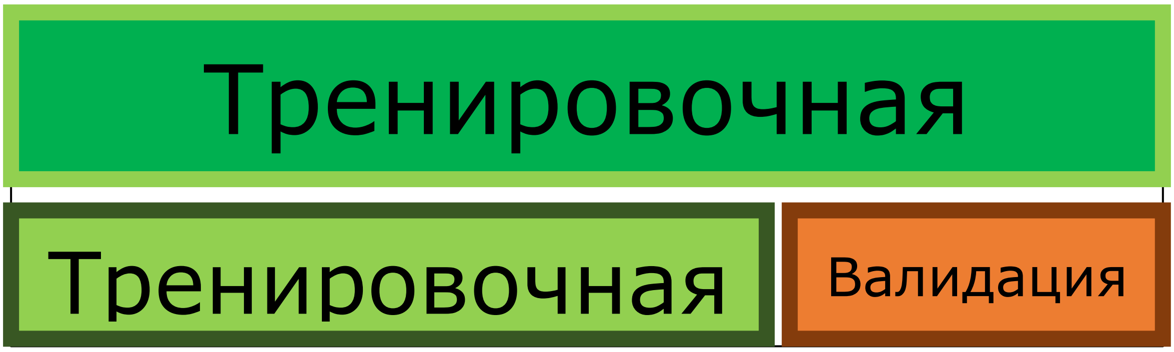

Рис. 1-8 Схема отложенной выборки

Для повышения уверенности в модели можно повторить процесс разбиения на тренировочную и валидационную несколько раз, каждый раз разбивая данные несколько различным образом.

Или же можно воспользоваться подходом n-Fold Кросс-валидация (n-Fold Cross-Validation split) (пример с n = 4 показан Рис. 1-9). Тренировочная выборка разбивается на n одинаковых по объему частей, которые содержат разные объекты. Производится n итераций. На каждой итерации происходит следующее:

- модель обучается на n - 1 частях обучающей выборки;
- модель тестируется на части обучающей выборки, которая не участвовала в обучении.

Итоговая оценка функционала потерь усредняется по всем n итерациям. Как правило, n=10 (5 в случае малого размера выборки).

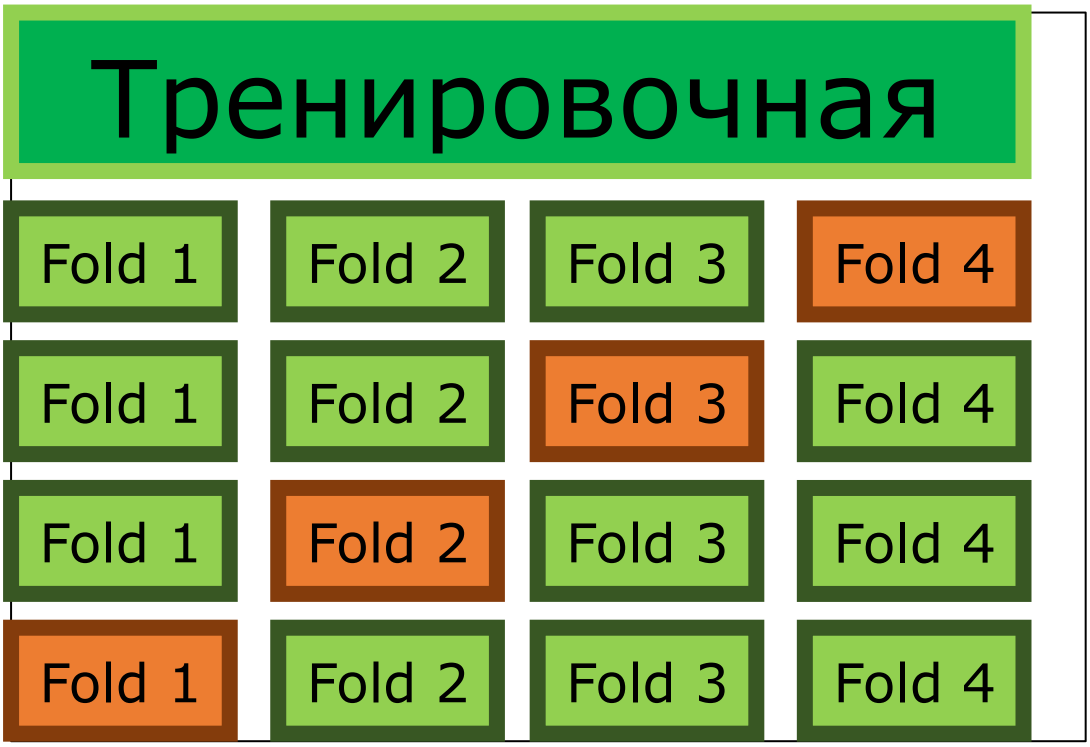

Рис. 1-9 Схема n-fold кросс-валидации

Так или иначе это позволяет *смоделировать* ситуацию, когда модель тестируется на новых, не виденных ранее данных. Поэтому потери на валидационных данных можно использовать для определения оптимальных параметров и гиперпараметров модели (Рис. 1-10).

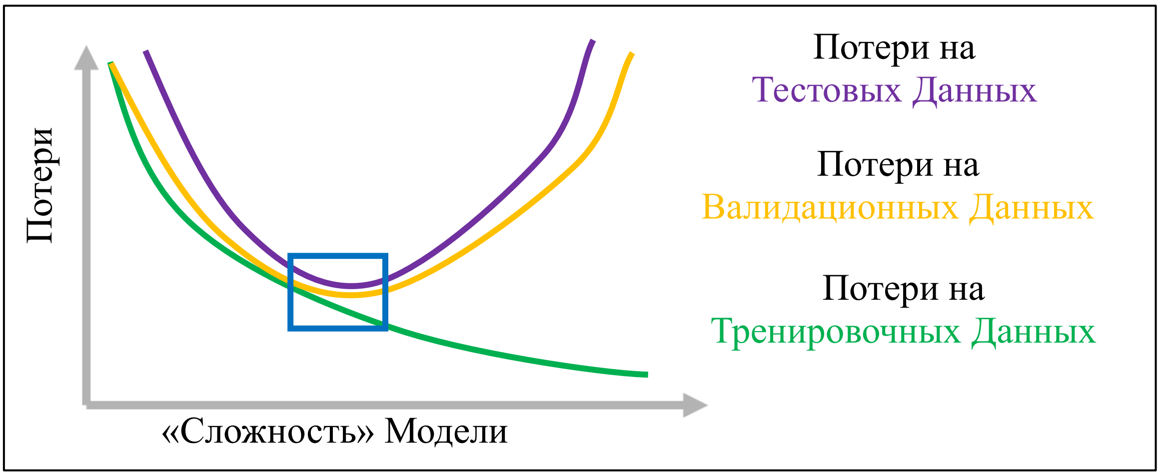

Рис. 1-10 Зависимость потерь от сложности моделей с учетом валидационных данных

Разложение ошибки на смещение и дисперсию

В общем случае функционал потерь можно разложить на следующие составляющие:

$$Loss(a, \mathbb{X}) \sim Bias(a(\mathbb{X})) + Variance(a(\mathbb{X})) + \sigma^2,$$

где σ^2 – случайный шум, с которым мы, к сожалению, ничего не сможем сделать.

В связи с этим мы должны быть готовы, что модели не могут быть совершенными. Однако мы можем уменьшать другие слагаемые в данном разложении.

Ошибка **смещения** (**Bias** Error) – это ошибка из-за ошибочных предположений в алгоритме обучения. Высокая ошибка смещения - модель слишком проста и не способна отразить закономерности в данных. Ошибку смещения мы можем оценить, используя тренировочные данные.

Дисперсия (Variance) – это ошибка из-за чувствительности к небольшим колебаниям обучающей выборки. Высокая дисперсия - модель плохо работает на новых данных. Дисперсию модели мы можем оценить с использованием валидационных и тестовых данных.

В зависимости от величины смещения и дисперсии существуют различные ситуации, которые можно описать с помощью мишеней, как указано на Рис. 1-11. Здесь близость к «яблочку» показывает наименьшее значения функции потерь для конкретного объекта.

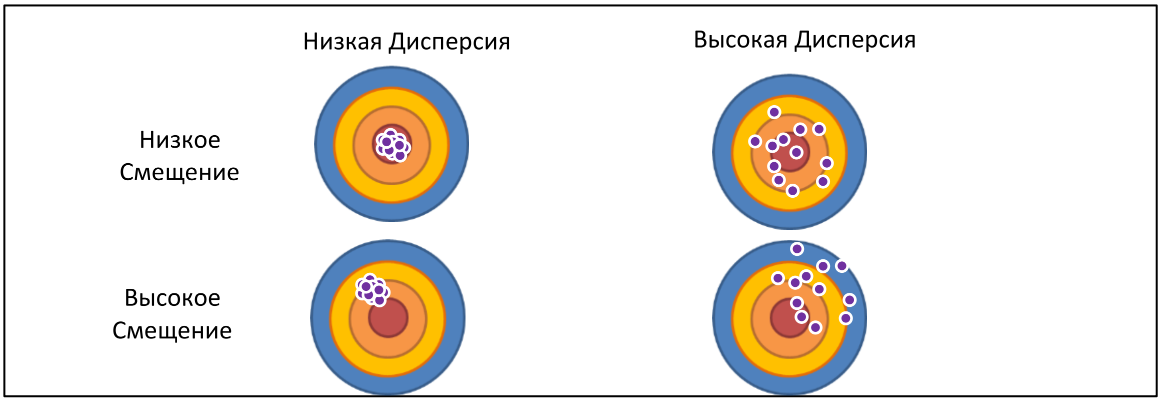

Рис. 1-11 Схематичное представление разложение ошибок модели

Идеальная ситуация – низкое смещение и низкая дисперсия. Модель хорошо работает в среднем и при этом для разных данных эта тенденция сохраняется. В случае высокого смещения и низкой дисперсии модель выдает стабильные предсказания как на тренировочной, так и на валидационной выборке, но, к сожалению, эти предсказания далеки от реальных значений. С другой стороны, ситуация низкого смещения и высокой дисперсии показывает то, что модель в среднем работает неплохо, однако существует достаточно большое количество отдельных объектов, на которых предсказания просто ужасны. Наконец худшая из возможных ситуаций – высокое смещение и высокая дисперсия, говорит о том, что модель совсем не подходит для имеющихся у нас данных.

Идеальные ситуации встречаются редко и поэтому необходимо находить компромисс между высоким смещением и высокой дисперсией. А это уже зависит от конкретной постановки задачи.

Задачи машинного обучения

Настало время поговорить о типовых задачах Машинного обучения. Стандартно задачи машинного обучения разделяют на следующие:

- Обучение с учителем (Supervised learning)
- Обучение без учителя (Unsupervised learning)

- Обучение с подкреплением (Reinforcement learning)

Обучение с учителем: есть **обучающий** набор данных (тренировочная выборка). Для каждого **экземпляра** из набора данных есть пары входные данные/признаки и ожидаемый ответ. В этом случае задачей является поиск модели или "алгоритма", который предсказывает ожидаемые целевые ответы.

Далее возникают различные ситуации в зависимости от того, что мы ожидаем в качестве ожидаемых ответов. Если множество возможных ответов конечно, то речь идет о задаче **Классификации**:

- $\mathbb{Y} \in \{1, \dots, K\}$, $K \in \mathbb{Z}$ в случае многих классов;
- $\mathbb{Y} \in \{-1, +1\}$ или $\mathbb{Y} \in \{0, 1\}$ в случае двух классов.

Схематично это представляется следующим образом: у нас есть некоторые пространство признаков, при этом существует заранее известно разметка (раскраска) отдельных объектов. И задача классификации сводится к построению такой разделяющей кривой, которая способна предсказывать метку или класс объекта (Рис. 1-12).

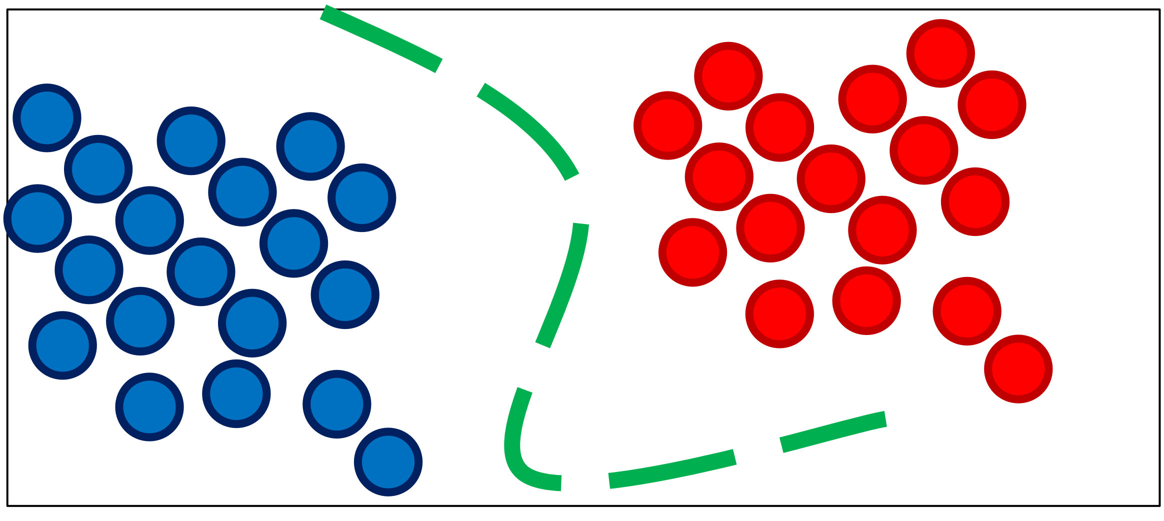

Рис. 1-12 Схематичное представление задачи классификации

В данных из Рис. 1-1 Оценка за экзамен по Машинному обучению является целевой переменной для задачи классификации, т. к. количество возможных ответов конечно.

С другой стороны, множество возможных ответов может быть почти бесконечным, т. е. $\mathbb{Y} \in \mathbb{R}$. В таком случае решается задача **Регрессии** (Рис. 1-13). Простой пример – у нас есть статистика по стоимости квартиры и ее площади. Но не для всех значений площадей. Используя имеющиеся данные, мы хотим предсказать стоимость квартиры для тех значений площадей, которых нет в нашей тренировочной выборке.

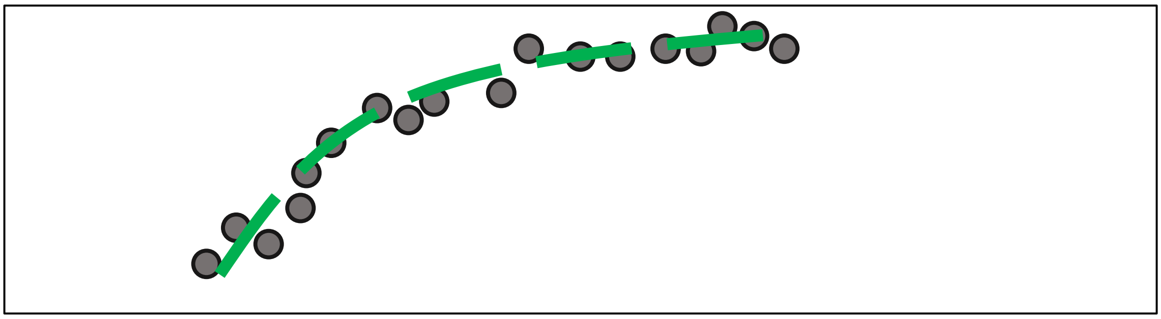

Рис. 1-13 Схематично представление задачи регресии

В данных из Рис. 1-1 Балл за текущую успеваемость по Машинному обучению может быть целевой переменной для задачи регрессии, т. к. количество возможных ответов конечно.

Обучение без учителя: постановка задачи может смутить, так как изначально у нас есть только **данные.** При этом мы хотим что-то понять и найти закономерности в этих данных.

Например, задача **Кластеризации**. Мы хотим разметить принадлежность отдельных объектов к различным кластерам, используя, например, близость или похожесть отдельных точек (Рис. 1-14).

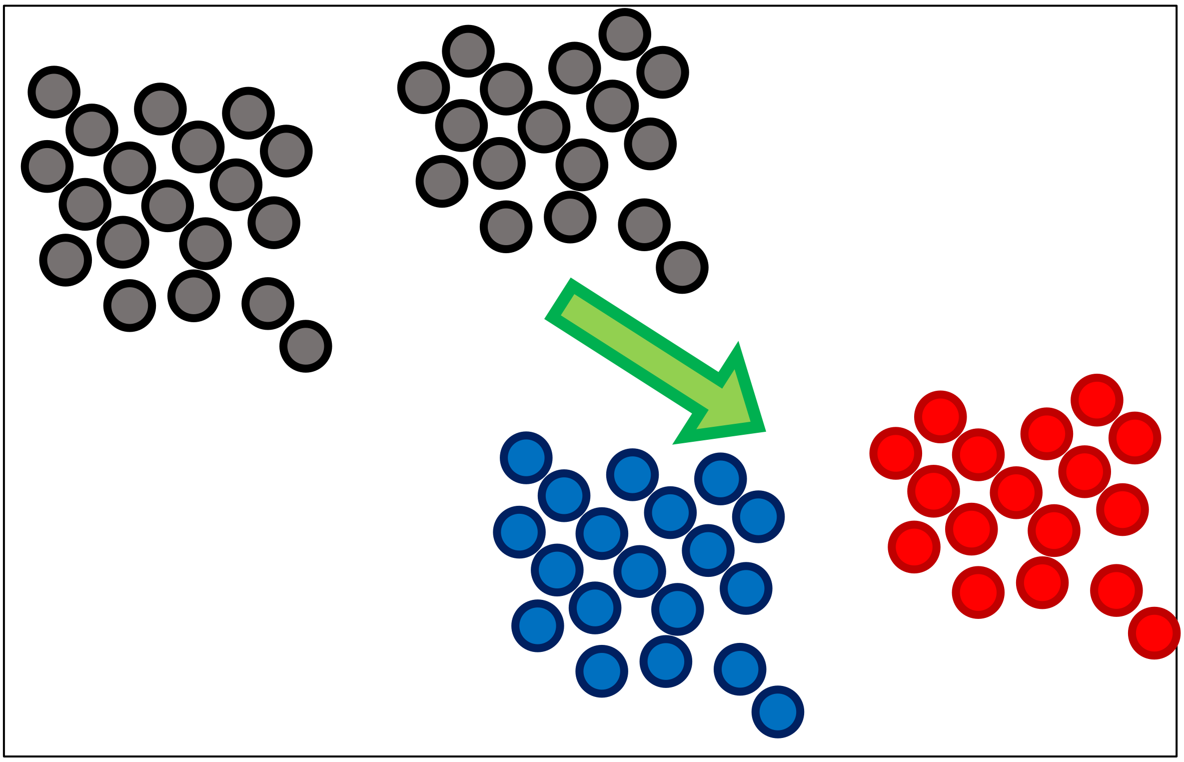

Рис. 1-14 Схематично представление задачи кластеризации

Стоит отметить, что в отличии от задачи классификации нам изначально не известны реальные классы объектов, и алгоритм должен принимать решение исключительно исходя из внутренних закономерностей в данных.

Другой типовой задачей обучения без учителя является задача **Снижения размерности**. Имеются табличные данные, большой размерности. И мы хотим построить такую новую таблицу данных, используя исходные данные, чтобы было проще с этим работать. Например, мы хотим сократить размерность табличных данных до 2 или 3, чтобы мы могли визуализировать имеющиеся у нас данные.

Обучение с подкреплением: есть среда и есть некоторая система, которая взаимодействует со средой. Задача состоит в эффективном взаимодействии со средой.

Линейная алгебра

Перед тем, как обучать модель и решать задачи необходимо вспомнить базовые вещи из линейной алгебры для манипуляций с данными. Ключевые вещи, которые необходимо понимать это **объекты** и **операции.**

Самым простым объектом является просто число, или **скаляр**. Обозначается $x \in \mathbb{R}$. По-простому – это одна ячейка в табличных данных.

Далее существуют совокупности из нескольких скаляров, которые называются **векторы**. Обозначаются $\boldsymbol{x} \in \mathbb{R}^n$, где n – размерность вектора. В табличных данных выделяют векторы-строки и векторы-столбцы.

Поскольку мы можем объединить несколько скаляров в вектор, то, наверное, мы можем объединить несколько векторов одинаковой размерности в новый объект, который называется **матрица**. Обозначается как $\boldsymbol{X} = \left[x_{ij} \right]_{m \times n}$ или $\boldsymbol{X} \in \mathbb{R}^{m \times n}$, где m – количество строк, n – количество столбцов. По сути, матрицы и есть таблицы данных.

Наконец, мы можем продолжать объединять Матрицы одинаковой размерности в новые объекты, которые называются **тензоры**. Трехмерный тензор обозначается как $\boldsymbol{X} = x_{ijk}$. Мы с вами уже рассматривали данные в виде тензоров. Это трехканальное цветное изображение.

В линейной алгебре, по сути, используются те же **операции**, что и в простой школьной алгебре (сложение, вычитание, умножение, деление), но с некоторыми особенностями. Ключевая особенность – нужно внимательно следить за размерностью объектов, над которыми совершаются операции.

- Сложение матриц

$\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{m \times n}$, $\boldsymbol{C} \in \mathbb{R}^{m \times n}$;

Результатом сложения двух матриц, у которых одинаковая размерность, является матрица той же размерности, при этом каждый элемент является суммой соответствующих элементов исходных матриц

$\boldsymbol{C} = \boldsymbol{A} + \boldsymbol{B}$.

- Матрично-скалярное сложение

$\boldsymbol{A} \in \mathbb{R}^{m \times n}, x \in \mathbb{R}$, $\boldsymbol{B} \in \mathbb{R}^{m \times n}$;

При этом мы также можем складывать матрицы и скаляры. В данном случае скаляр добавляется к каждому элементу исходной матрицы.

$$\boldsymbol{B} = \boldsymbol{A} + x.$$

- Broadcasting (сложение матрицы и вектора)

$$\boldsymbol{A} \in \mathbf{R}^{m \times n}, \ \boldsymbol{x} \in \mathbb{R}^{n}, \ \boldsymbol{B} \in \mathbb{R}^{m \times n};$$

Также мы можем интересным образом складывать вектора и матрицы. В данном случае количество столбцов в матрице должно совпадать с размерностью вектора. В данном случае вектор распространяется (от английского to broadcast) по всем строкам матрицы

$$\boldsymbol{B} = \boldsymbol{A} + \boldsymbol{x}.$$

Стоит отметить, что в строгом математическом смысле такой операции нет, однако она реализована в библиотеке NumPy языка Python/

- Умножение Матрицы на Матрицу

$$\boldsymbol{A} = \left[a_{ij}\right]_{m \times n}, \ \boldsymbol{B} = \left[b_{ij}\right]_{n \times p}, \ \boldsymbol{C} = \boldsymbol{AB} = \left[c_{ij}\right]_{m \times p};$$

Умножение матрицы на матрицу самое требовательное к размерностям исходных матриц. Мы можем умножать матрицу на матрицу только в том случае, если количество столбцов первой матрицы совпадает с количеством строк второй матрицы. При этом размерность итоговой матрицы, будет следующей: количество строк – будет равно количеству строк первой матрицы, а количество столбцов – количеству столбцов второй матрицы

При этом каждый элемент итоговой матрицы определяется исходя из следующих соображений: $c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$.

При этом в общем случае $\boldsymbol{AB} \neq \boldsymbol{BA}$.

- Поэлементное умножение (произведение Адамара)

$$\boldsymbol{A} = \left[a_{ij}\right]_{m \times n}, \ \boldsymbol{B} = \left[b_{ij}\right]_{m \times n};$$

Стоит также упомянуть операцию поэлементного умножения, которая вычисляется по аналогии с поэлементным сложением. Разумеется, что размерности исходных матриц должны совпадать

$$\boldsymbol{A} \odot \boldsymbol{B} = \left[a_{ij} b_{ij}\right]_{m \times n}.$$

- Транспонирование Матрицы

$$\boldsymbol{A} \in \mathbb{R}^{m \times n};$$

Иногда возникает необходимость «повернуть» матрицу, т. е. поменять местами столбцы и строки. Такая операция называется транспонированием.

$\boldsymbol{A}^T \in \mathbb{R}^{n \times m}$.

При этом, если мы выполняем операцию транспонирования два раза мы возвращаемся к исходной матрице.

$\boldsymbol{A}^{TT} = \boldsymbol{A}$.

- «Деление Матриц»

А вот деления матриц не существует. Есть своеобразный аналог делению из школьной алгебры: домножение на Обратную Матрицу. Обратная матрица соответствует следующему условию

$\boldsymbol{A} * \boldsymbol{A}^{-1} = \boldsymbol{A}^{-1} * \boldsymbol{A} = \boldsymbol{I}$,

где $\boldsymbol{I}$ - Единичная Матрица.

Математический анализ

Итак, мы вспомнили Объекты и базовые Операции. Однако для задач из реального мира требуется более сложное взаимодействие между объектами. Для этого нам понадобятся знания из математического анализа, которые описывают функции.

Функция в математике – это соответствие между элементами двух множеств, правило, по которому каждому элементу первого множества соответствует один и только один элемент второго множества. Схожая аналогия существуют также и, например, в программировании.

Самый простой пример функции – функция, которая «ничего не делает». Эта функция берет на вход переменную x и возвращает ее. Функции можно представить по-разному, можно в виде математического выражения: $f(x) = x$, а можно в виде графика функции (Рис. 1-15 а). Как видно данная функция выглядит как прямая линия. При этом у функции могут быть дополнительные параметры, например входные данные можно умножить на скаляр и добавить скаляр $f(x) = 0.5 \times x - 1$, график такой функции представлен на (Рис. 1-15 б). График все еще выглядит как прямая линия. Такие функции называются линейными функциями, т. е. выходной аргумент функции пропорционален входному аргументу.

Функции бывают также и нелинейными. Например, в раннее время использования нейронных сетей была достаточно распространена сигмоидная функция, которую также называют логистической, которая выражается уравнением $f(x)=\frac{1}{1+e^{-x}}$ (Рис. 1-15 в). Другим примером нелинейной функции из мира нейронных сетей является функция ReLU (Rectified linear unit – линейный выпрямитель) $f(x)=\max(0,x)$ (Рис. 1-15 г).

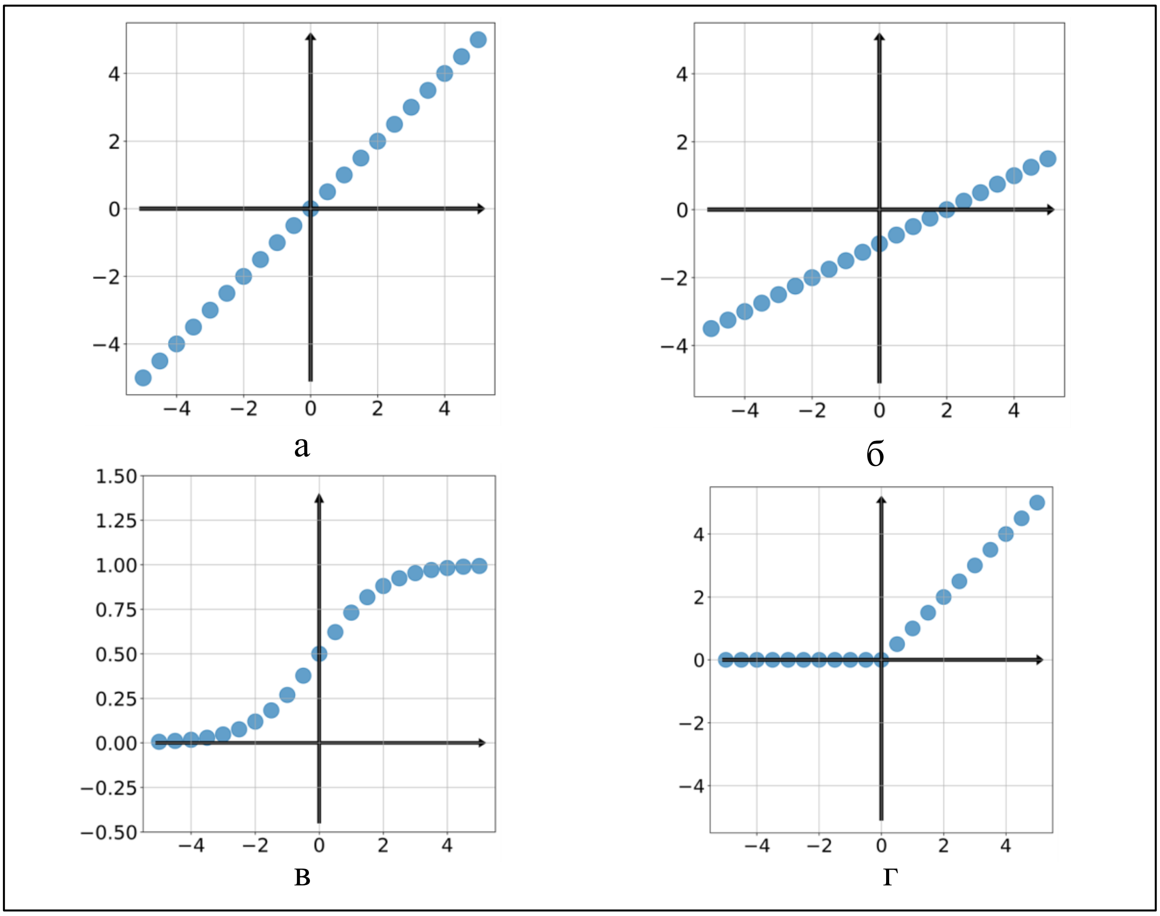

Рис. 1-15 Примеры графиков функций

Для анализа поведения функций часто используют другую функцию, которая называется производной. **Производная** функции в точке – скорость изменения функции в данной точке. Производную можно определить, как предел отношения приращения функции к приращению аргумента.

$$f'(x_0) = \lim_{x \to x_0} \frac{f(x) - f(x_0)}{x - x_0} = \lim_{\Delta x \to 0} \frac{f(x_0 + \Delta x) - f(x_0)}{\Delta x} = \lim_{\Delta x \to 0} \frac{\Delta f(x)}{\Delta x}.$$

На Рис. 1-16 представлены примеры производных для линейной функции (а), логистической функции (б) и ReLU (в). Так видно, что скорость изменения линейной функции – постоянна, логистической функции зависит от области значений, а для функции ReLU меняется «скачкообразно».

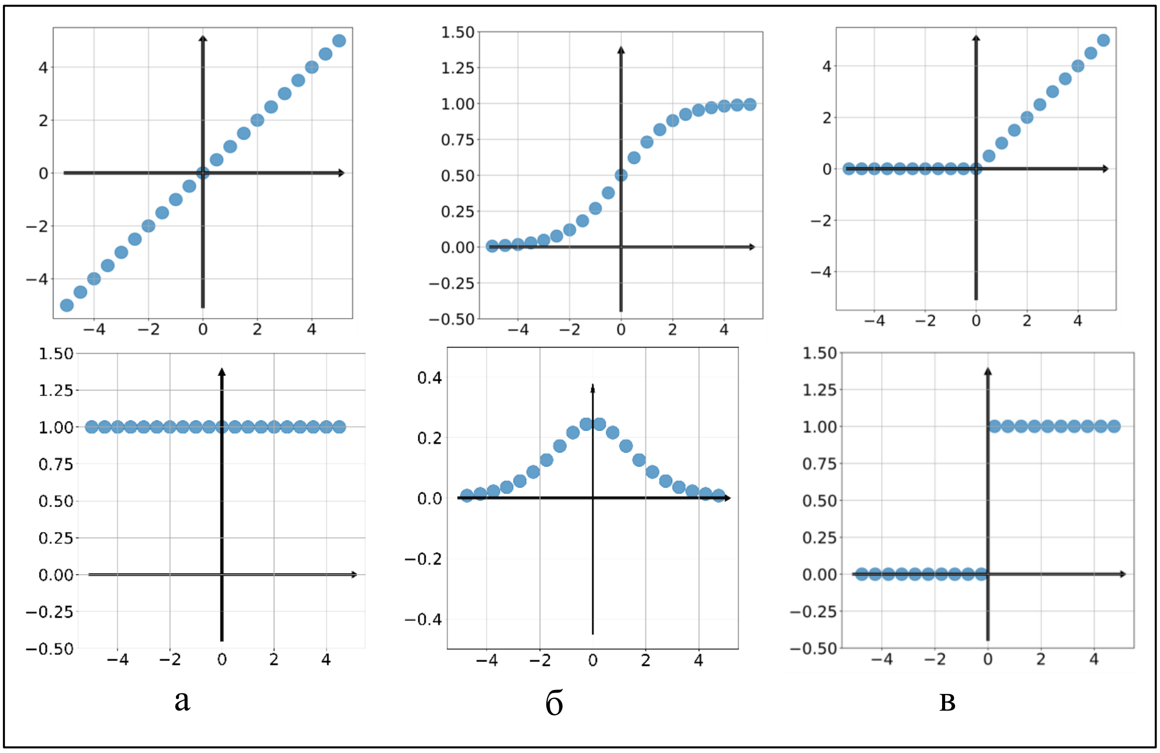

Рис. 1-16 Примеры функций и их производных

Ниже представлены функции и их производные, которых достаточно знать для успешного усвоения материала данного учебного пособия.

Функция	**Производная**
c - константа	0
x	1
x^m	$m \times x^{m-1}$
$\ln(x)$	$\frac{1}{x}$
e^x	e^x
$f(x) \times g(x)$	$f'(x) \times g(x) + f(x) \times g'(x)$
$c \times f(x)$	$c \times f'(x)$

$y=f(t)$ $t=g(x)$	$\frac{\partial y}{\partial x}=\frac{\partial y}{\partial t}\frac{\partial t}{\partial x}$

Отдельно стоит упомянуть последнюю строку в таблице – т. н. цепное правило. Если y это функция от переменной t, а t в свою очередь зависит от переменной x, тогда производная y по переменной t будет равна производной y по переменной t помноженную на производную t по переменной x.

Функции могут зависеть от разных переменных. Например, $f(x,w,b)=wx+b.$ Мы можем посчитать производные по трем переменным x,w,b. Если мы считаем производную $\frac{\partial f}{\partial x}$ то остальные переменные считаются константами. Производные по отдельным переменным называются частными производными. Вектор, составленный из частных производных, называется **градиент**. Для упомянутой выше функции будет выглядеть следующим образом $\nabla f(x,w,b)=\left[\frac{\partial f}{\partial x},\frac{\partial f}{\partial w},\frac{\partial f}{\partial b}\right]=[w,x,1]$

Отдельно рассмотрим производную квадратичной функции $f(x)=x^2.$ Напомним, что мы определили, что в общем случае цель обучения, это минимизация функционала потерь $Q(a,\mathbb{X})\to\min\limits_{a\in\mathbb{A}}.$ В качестве примера функционала потерь достаточно часто используется именно квадратичная функция. Допустим у этого функционала всего 1 параметр w (Рис. 1-17).

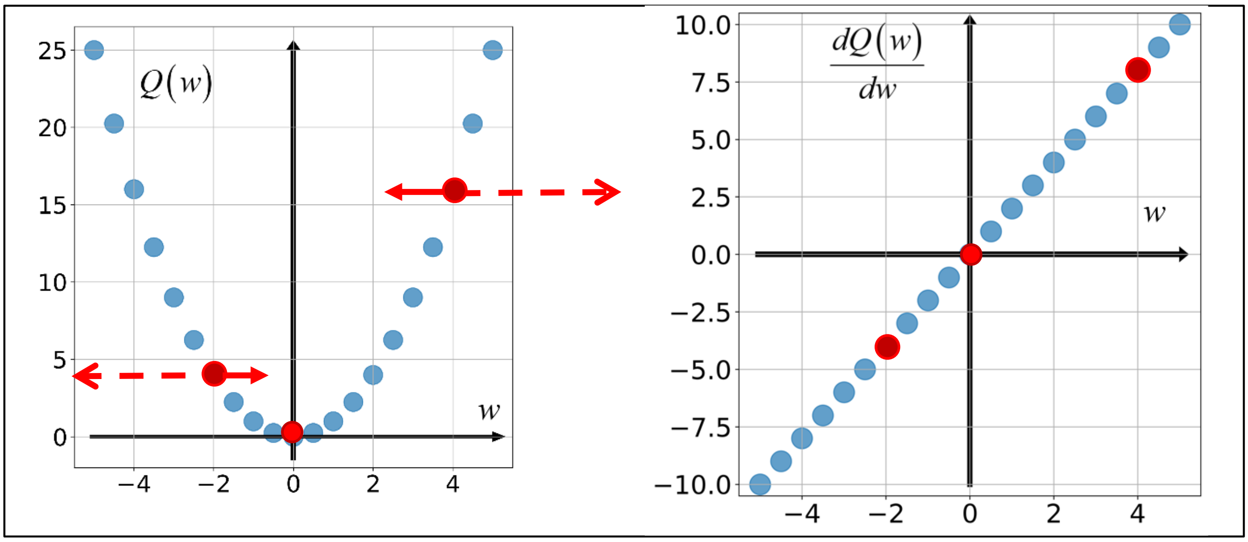

Рис. 1-17 Квадратичная функция и ее производная

Для того чтобы найти минимум этой функции нужно приравнять производную этой функции к нулю. Значение параметра w_0, при котором производная $\frac{dQ(w=w_0)}{dw}$ равна нулю будет соответствовать параметру, который минимизирует исходный функционал. Запомним это.

С другой стороны, допустим мы подобрали значения параметра случайным образом и хотим оценить, насколько хорош этот параметр и можно ли его как-то изменить, используя производную. Допустим $\frac{dQ(w=-2)}{dw}=-4$. Что это значит? В этой точки функция убывает (производная отрицательная), т. е. если продолжать увеличивать параметр, то мы скорее всего придем к минимуму функции. Аналогично рассмотрим $\frac{dQ(w=4)}{dw}=8$. Производная положительная, а значит функция возрастает. Причем возрастает быстрее, чем в точке $w=-2$. Это значит, что если мы уменьшим параметр, то мы приблизимся к минимуму функции. При этом нам нужно сделать больший шаг, поскольку абсолютная величина производной больше. Эта идея изменения параметров в направлении обратной знаку производной на величину пропорциональную значению производной и легла в основу алгоритма градиентного спуска, который мы с вами более подробно рассмотрим в 3 Главе.

Контрольные вопросы

1. Опишите разницу между обучением с учителем и обучением без учителя
2. Опишите разницу между задачами классификации и задачами регрессии
3. Вас попросят создать программу, которая будет определять кошку или собаку по изображению. К какому типу задач машинного обучения относится эта просьба?
4. Почему рекомендуется выполнять проверку модели на валидационных данных (в качестве альтернативы простому использованию всех тренировочных данных)?
5. У вас есть три матрицы A, B, C: A имеет размеры 5 × 4, B имеет размеры 4 × 6, C имеет размеры 3 × 5. Укажите все возможные матрицы, которые можно перемножить между собой.
6. Найдите градиент функции $f\left(x,y,z\right)=yx^2+\ln\left(y\right)+e^{-z}$.
7. Найдите частные производные сложной функции $E=\left(\hat{y}-y\right)^2$, $\hat{y}=wx+b$ по переменным w и b.

Найдите частные производные сложной функции $E=y\times\ln\hat{y}+\left(1-y\right)\times\ln\left(1-\hat{y}\right)$, $\hat{y}=\dfrac{1}{1+e^{-z}}$ по переменной z.