Проектирование интеллектуальных систем

Лекция № 3. Введение в теорию искусственных нейронных сетей

Содержание

Строение биологического нейрона

Искусственные нейронные сети появились в результате попытки смоделировать работу человеческого мозга на компьютере. Базовой структурной и функциональной единицей организации мозга человека является *нейрон* – клетка, которая принимает, обрабатывает и передаёт электрохимические сигналы по нервным путям, образующим коммуникационную систему мозга [3]. С точки зрения информационной парадигмы нейрон является основной единицей обработки информации [2]. Модель нейрона, используемая в искусственных нейронных сетях, не стремится к биологическому правдоподобию, а лишь воспроизводит основные информационные процессы, происходящие в нейроне.

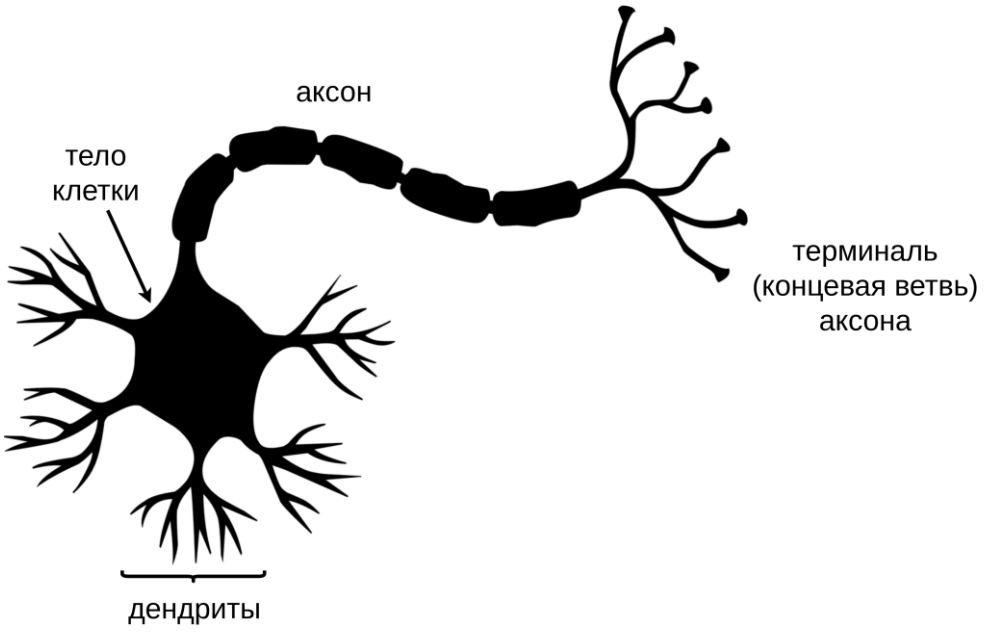


Рис. 1. Структура биологического нейрона.

Из тела клетки биологического нейрона отходит два вида отростков: **дендриты** и **аксон** (рис. 1). По дендритам нейрон получает сигналы от связанных с ним нейронов, а через аксон – передаёт сигнал дальше по сети. Ме-

ста соединения дендритов одного нейрона с аксонами других нейронов называются **синапсами**. В **теле клетки** накапливается заряд, поступающий от входных сигналов, и, как только этот заряд преодолевает некоторое пороговое значение, нейрон **активируется** и передаёт сигнал дальше по аксону. Если заряд не преодолевает пороговое значение, то сигнал дальше не передаётся.

Передача сигнала в синапсах осуществляется с помощью специального вещества – **нейротрансмиттера (нейромедиатора)**. Оно выделяется из специальных «пузырьков» (*везикул*), расположенных в концевой части аксона (*терминали*), в *синаптическую щель*, а затем попадает на *постсинаптическую мембрану* другого нейрона, на которой расположены специальные рецепторы, чувствительные к нейромедиаторам (рис. 2). Таким образом, передача сигналов между нейронами осуществляется *химическим путём*.

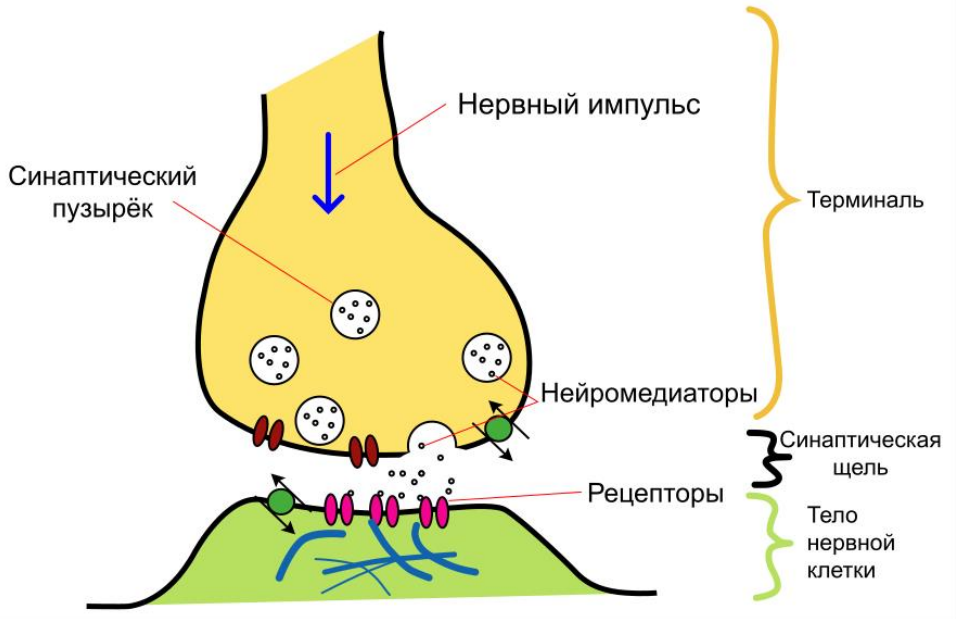


Рис. 2. Строение синапса.

Математическая модель нейрона

Под математической моделью нейрона следует понимать «представление исходной структуры живого нейрона и протекающих в нём процессов в форме математических объектов и отношений между ними, при котором структура нейрона и протекающие в нём процессы превращаются в истинные предположения о них» [2].

Используемая в современных искусственных нейронных сетях математическая модель нейрона была предложена в 1943 г. У. Мак-Каллоком и У. Питтсом. В этой модели искусственный нейрон может быть описан при помощи следующих элементов: входных сигналов, синаптических весов, сумматорной функции, функции активации и выходного значения.

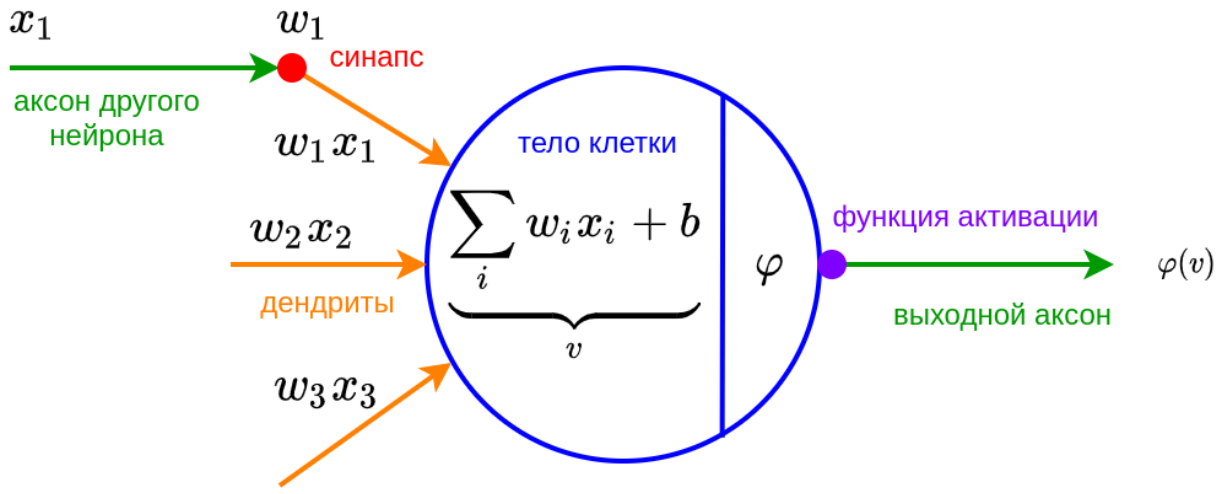


Рис. 3. Математическая модель нейрона.

Входные сигналы можно обозначить как $x_1, \dots, x_m$ – это сигналы, которые поступают в нейрон от рецепторов или других нейронов. Эти сигналы передаются через синапсы, для моделирования чувствительности которых используются синаптические веса $w_1, \dots, w_m$. Синаптические веса представ-

ляют собой действительные числа, которые могут быть как положительными, так и отрицательными, в зависимости от того, является ли этот синапс возбуждающим или тормозящим. Нулевое значение синаптического веса эквивалентно отсутствию связи между нейронами.

При прохождении через синапс, входной сигнал умножается на соответствующий синаптический вес, т. е. вычисляются следующие значения: $x_i \cdot w_i,\ i = 1, \dots, m$. После этого полученные значения передаются в сумматорную функцию, которая вычисляет выражение

$$u = \sum_{i=1}^{m} x_i \cdot w_i.$$

Эта процедура моделирует «аккумуляцию» сигнала в биологическом нейроне. Величина u называется **линейной комбинацией входных воздействий** [4].

Чтобы сформировать выходной сигнал y используется некоторая функция активации φ, аргументом которой является взвешенная сумма (линейная комбинация) входных воздействий: $y = \varphi(u)$.

В классической модели нейрона используется **пороговая функция активации**, которую также называют **функцией Хевисайда**:

$$\varphi(u) = \begin{cases} 1, & \text{если } u > p; \\ 0, & \text{иначе.} \end{cases},$$

где p – некоторое пороговое значение.

Таким образом, нейрон может быть в двух состояниях: активен ($y = 1$) и не активен ($y = 0$).

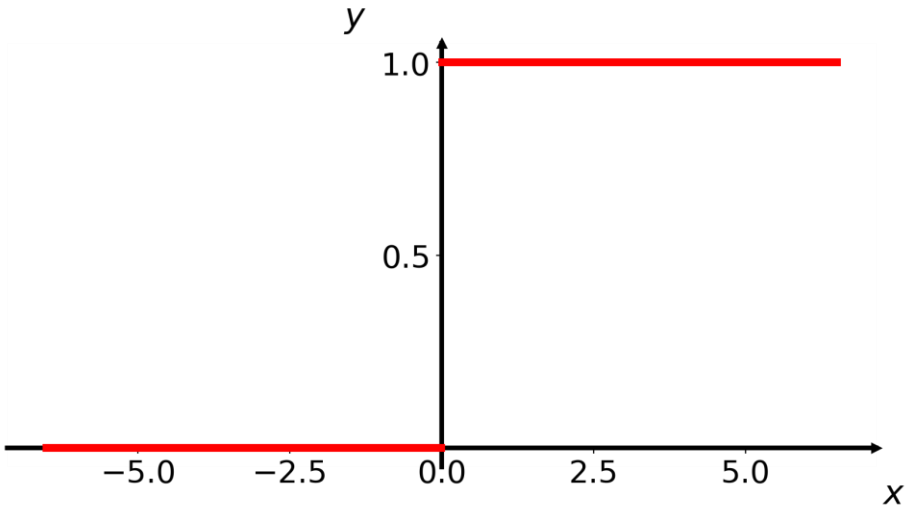


Рис. 4. График функции Хевисайда.

Положение «ступеньки» в пороговой функции регулируется с помощью параметра p. Зачастую вместо того, чтобы выполнять сравнение $u > p$ параметр p переносят в левую часть неравенства ($u - p > 0$) и вводят обозначение $b = -p$. Величина b называется **смещением** (англ. bias). Для удобства введём обозначение $v = u + b$ (величина v называется **постсинаптический потенциал** [4]). Тогда пороговую функцию можно записать в виде

$$\varphi(v) = \begin{cases} 1, & \text{если } v > 0; \\ 0, & \text{иначе.} \end{cases}$$

Такая запись более удобна по нескольким причинам. Во-первых, смещение является более общим понятием и позволяет регулировать положение функций активации, в которых нет «порога» как такового. Во-вторых, введение смещения позволяет работать с ним так же, как и с синаптическими весами, если ввести «фиктивный вход», на который всегда подаётся единичный сигнал. Тогда смещение можно считать синаптическим весом, соответствующим этому входу. Такой «трюк» упрощает программную реализацию искусственного нейрона.

Линейные алгоритмы машинного обучения

Искусственные нейронные сети можно считать одной из первых форм машинного обучения. **Машинное обучение** – это раздел искусственного интеллекта, в рамках которого изучаются методы решения плохоформализуемых задач посредством разработки алгоритмов, способных обучаться на основе опыта (данных).

Классическая формулировка задачи машинного обучения выглядит следующим образом. Прежде всего, выбирается некоторое **пространство объектов** (или генеральная совокупность) $\mathbb{X}$ – это множество всех объектов, относительно которых нужно делать какие-то предсказания. Выбор пространства объектов зависит от существа решаемой задачи. Например, в случае решения задачи автоматической диагностики, пространство объектов – это множество всех возможных пациентов, относительно диагноза которых нужно сделать прогноз. Каждый объект характеризуется некоторым набором характеристик, называемых **признаками**, которые могут быть как количественными, так и качественными.

Также определяется некоторая **целевая переменная** – характеристика объектов, которую нужно предсказать. Для целевой переменной определяется множество возможных значений $\mathbb{Y}$.

Цель состоит в том, чтобы построить отображение $f: \mathbb{X} \to \mathbb{Y}$, позволяющее сопоставить каждому объекту соответствующее ему значение целевой переменной. Для поиска этой зависимости требуется *обучающая выборка*

$$T = \{(\mathbf{x_i}, y_i) \mid \mathbf{x_i} \in \mathbb{X};\ y_i \in \mathbb{Y};\ i = 1, \dots, n\},$$

которая представляет собой множество пар (объект, значение целевой переменной).

Задача определения искомой зависимости обычно сводится к задаче минимизации **целевой функции** (**функции потерь**) на обучающей выборке

$$L(f,T) = \frac{1}{n}\sum_{i=1}^{n} L_i\,(f, \mathbf{x_i}, y_i) \to min.$$

Таким образом, решение сводится к выбору такой функции f^*, для которой значение функции потерь на обучающей выборке минимально. Однако, чтобы решить эту задачу, необходимо сделать предположение о характере искомой зависимости и ограничить пространство поиска некоторым семейством функций $\mathbb{A}$ (например, линейных функций)

$$f^* = \underset{f\in\mathbb{A}}{argmin}\, L(f,T).$$

Чтобы однозначно определить некоторую функцию из выбранного семейства, необходимо сопоставить ей набор параметров. Таким образом, поиск функции сводится к поиску оптимального набора параметров

$$w^* = \underset{w\in\mathbb{W}}{argmin}\, L(f_w,T),$$

где $\mathbb{W}$ – множество возможных значений параметров.

Выбор функции потерь зависит от характера решаемой задачи, который, в свою очередь, определяет множество возможных значений целевой переменной. Если $\mathbb{Y} = \mathbb{R}$ (то есть множество возможных значений совпадает со множеством действительных чисел), то такая задача называется **задачей регрессии**. Если же $\mathbb{Y}$ представляет собой множество дискретных значений (например, $\{0,1\}$), то это **задача классификации**. С помощью описанной выше модели нейрона можно решить задачу бинарной классификации, а сама такая модель эквивалентна *пороговому линейному классификатору*.

Алгоритмы классификации строят разделяющую границу в пространстве признаков таким образом, что объекты разных классов оказываются по раз-

ные стороны от этой границы. Если граница, разделяющая классы, имеет вид прямой (плоскости, гиперплоскости), то такой классификатор называется линейным.

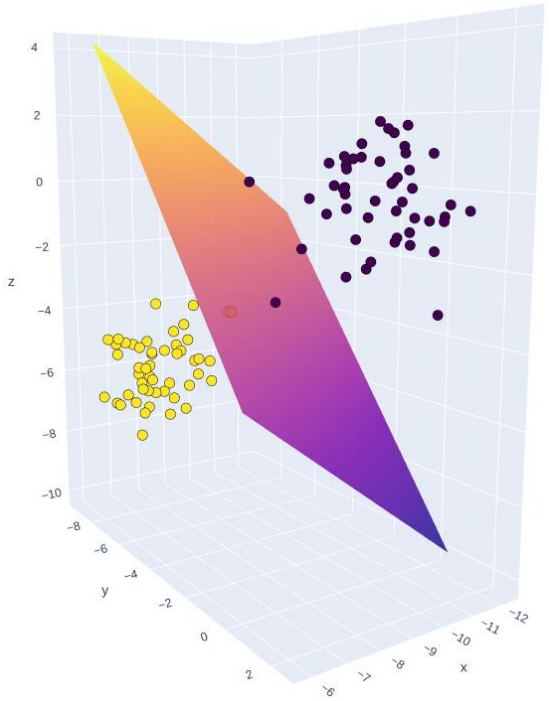


Рис. 5. Линейная разделяющая граница.

Для рассмотренной модели нейрона уравнение разделяющей границы имеет вид

$$w_1 x_1 + w_2 x_2 + \cdots + w_m x_m + b = 0.$$

Это уравнение задаёт гиперплоскость в пространстве признаков. Вектором нормали этой гиперплоскости является вектор синаптических весов $\mathbf{w} = (w_1, \ldots, w_m)$.

По значению функции $f(\mathbf{x}) = w_1 x_1 + w_2 x_2 + \cdots + w_m x_m + b$ можно определить положение точки с координатами $\mathbf{x} = (x_1, \ldots, x_m)$ относительно упомянутой гиперплоскости. Если $f(\mathbf{x}) > 0$, то точка лежит по одну сторону от гиперплоскости, если $f(\mathbf{x}) < 0$ – по другую, если же $f(\mathbf{x}) = 0$, то точка

лежит на гиперплоскости. Знак функции зависит от направления вектора нормали. Чем больше значение $|f(\mathbf{x})|$, тем больше расстояние от точки до гиперплоскости.

Для линейных классификаторов часто вводится понятие **отступ** (англ. margin) – величина, характеризующая «уверенность» классификатора в ответе. Предположим, что метки классов y закодированы числами 1 и -1. Тогда величина отступа для объекта $\mathbf{x}$ может быть определена как

$$M(\mathbf{x}) = y\,((\mathbf{x}, \mathbf{w}) + b),$$

где $(\mathbf{x}, \mathbf{w})$ – скалярное произведение вектора признаков $\mathbf{x}$ на вектор параметров модели $\mathbf{w}$, $y \in \{-1,1\}$ – метка класса объекта.

Величина отступа положительна, если классификатор верно определил класс объекта. В противном случае величина отступа отрицательна. Чем дальше объект находится от разделяющей границы, тем больше величина отступа.

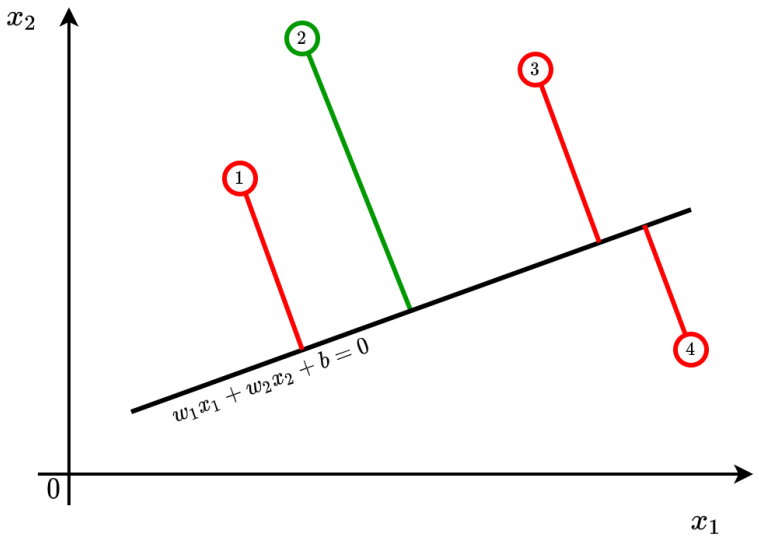


Рис. 6. Отступы для верно и неверно классифицированных объектов.

На основе величины отступа можно определить 1/0 функцию потерь (англ. zero-one loss) следующим образом

$$L(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}[\,M(\mathbf{x_i}) < 0],$$

где n – количество наблюдений в выборке, $[\cdot]$ – индикаторная функция. Значением этой функции является количество ошибок, которые сделал классификатор.

Однослойный персептрон

В 1958 г. Фрэнк Розенблатт ввел понятие персептрона как первой модели обучения с учителем [4]. В простейшем случае персептрон представляет собой нейрон с пороговой функцией активации (т. е. является линейным пороговым классификатором). Ф. Розенблатт предложил алгоритм для обучения персептрона, основанный на принципе коррекции ошибок. Суть этого алгоритма заключается в следующем.

Шаг 0. Инициализировать значения синаптических весов и смещения небольшими случайными числами.

Шаг 1. Выбрать элемент обучающей выборки $(\mathbf{x_i}, y_i)$.

- Подать на вход НС вектор $\mathbf{x_i}$ и вычислить значение $\hat{y}_i$.
- Сравнить значения $\hat{y}_i$ и y_i.
  - Если $\hat{y}_i > y_i$, то **уменьшить** веса всех **активных** входов.
  - Если $\hat{y}_i < y_i$, то **увеличить** веса всех **активных** входов.

Шаг 2. Выбрать следующий элемент обучающей выборки $(i := i + 1)$, пока не обработаны все элементы $(i \leq n)$.

Шаг 3. Проверить НС на правильность работы для всех элементов обучающей выборки.

- Если $\forall\, i \in\ [1, n]\ \ \hat{y}_i = y_i$, то завершить работу алгоритма.
- Иначе перейти к шагу 1.

Идея алгоритма довольно проста. Объекты обучающей выборки перебираются по очереди. Для каждого объекта вычисляется отклик нейронной сети (выходное значение). Если полученный отклик не совпал с желаемым (т. е. если персептрон допустил ошибку), то производится корректировка синаптических весов.

Персептрон может допустить два вида ошибок: выдать 0 вместо 1 или 1 вместо 0. В первом случае необходимо увеличить веса всех активных входов, а во втором – уменьшить. Под активными входами понимаются те входы, на которые был подан единичный сигнал. Веса неактивных входов (т. е. тех, на которые был подан нулевой сигнал) не должны подвергаться корректировке, так как они не оказывают никакого влияния на выходной сигнал, а значит не влияют и на ошибку.

Увеличение весов активных входов приведёт к увеличению значения взвешенной суммы входных сигналов, что, в свою очередь, поспособствует тому, чтобы значение взвешенной суммы преодолело пороговое значение, в результате чего нейрон будет выдавать 1 для данного объекта обучающей выборки. Уменьшение весов активных входов приведёт к противоположному эффекту.

Если данные в обучающей выборке линейно разделимы, то алгоритм завершит свою работу за конечное число шагов.

Алгоритм обучения персептрона можно записать более компактно – с использованием выражения, которое получило название *дельта-правило*.

Дельта-правило позволяет вычислить корректировку, применяемую к синаптическому весу нейрона на каждой итерации обучения. Согласно дельта-правилу, эта корректировка пропорциональна произведению сигнала ошибки на входной сигнал, его вызвавший:

$$\Delta_i = \alpha(y - \hat{y})x_i,$$

$$w_i^{(k+1)} = w_i^{(k)} + \Delta_i,$$

где Δ_i – величина корректировки веса w_i, y – желаемый отклик нейрона, $\hat{y}$ – фактический отклик, x_i – входной сигнал, поданный на i-й вход, k – номер итерации обучения, α – коэффициент скорости обучения (англ. leraning rate).

Фрэнк Розенблатт доказал сходимость предложенного им алгоритма обучения персептрона. Согласно *теореме о сходимости персептрона*, независимо от начальных значений синаптических весов и порядка показа образцов при обучении, персептрон за конечное число шагов научится различать два класса объектов, если только существует такая классификация [3,4]. Иными словами, персептрон способен обучиться всему, что он может представлять. Однако, представлять он может только *линейно разделимые функции*. Классической иллюстрацией этого ограничения является так называемая *проблема XOR*.

Таблица истинности функции XOR.

Точка	**x**	**y**	**x XOR y**
A_0	0	0	0
B_1	0	1	1
B_0	1	0	1
A_1	1	1	0

Проблема XOR заключается в том, что с помощью однослойного персептрона нельзя воспроизвести функцию «исключающее или» (XOR). Это становится очевидным, если рассмотреть геометрическую интерпретацию задачи. В таблице истинности, представленной выше, обозначены точки A_0, A_1, B_0 и B_1. Их можно изобразить на плоскости. Тогда задача моделирования функции XOR сводится к тому, что нужно провести разделящую прямую таким образом, чтобы точки A_0 и A_1 оказались по оду сторону от этой прямой, а точки B_0 и B_1 – по другую. Очевидно, что это сделать невозможно, потому что функция XOR *линейно неразделима*.

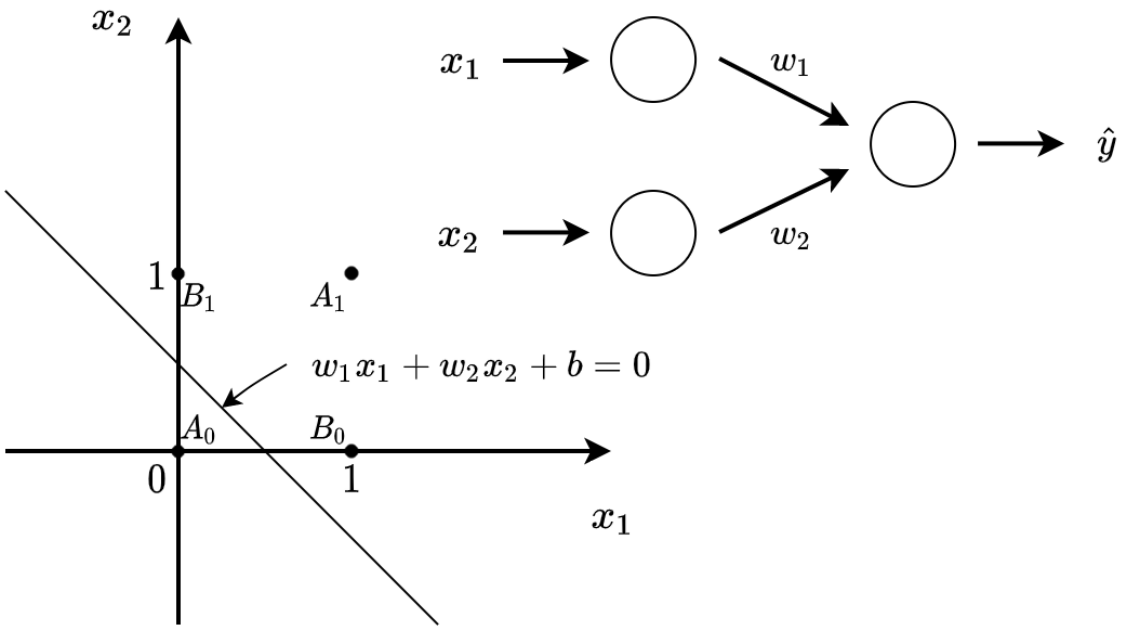


Рис. 7. Геометрическая интерпретация проблемы XOR.

Однако, нужно отметить, что если отобразить данные в *спрямляющее пространство*, то задача может быть решена с помощью однослойного персептрона [1]. Например, если добавить ещё один признак $x \cdot y$, то точки

A_0, A_1, B_0 и B_1 отобразятся в трёхмерное пространство, в котором возможно провести разделяющую плоскость между точками A_0, A_1 и B_0, B_1.

Адаптивный линейный элемент (ADALINE)

Если в качестве функции активации использовать линейную функцию $\varphi(v) = v$, то получим модель, называемую *адаптивный линейный элемент (ADALINE)*. Эта модель была предложена Б. Видроу и М. Хоффом в 1960 г. как развитие модели нейрона Мак-Каллока и Питтса.

Для обучения своей модели Б. Видроу и М. Хофф предложили правило, которое в точности совпадает с дельта-правилом для обучения персептрона:

$$w_i^{(k+1)} = w_i^{(k)} + \alpha(y - \hat{y})x_i.$$

Это правило можно получить, применив алгоритм стохастического градиентного спуска для минимизации квадратичной функции потерь

$$L(\mathbf{w}) = \frac{1}{2n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \to \min_{\mathbf{w}}.$$

Таким образом модель ADALINE эквивалентна *линейной регрессии*, параметры которой настраиваются при помощи стохастического градиентного спуска.

Градиентный спуск: основы

Наиболее часто для обучения искусственных нейронных сетей используется алгоритм градиентного спуска. Он представляет собой численный метод поиска локального минимума функции. Этот алгоритм основан на свойствах градиента функции.

Градиент — вектор, своим направлением указывающий направление наибольшего возрастания некоторой величины φ, значение которой меняется от одной точки пространства к другой (скалярного поля), а по величине (модулю) равный скорости роста этой величины в этом направлении.

$$grad\ \varphi = \nabla\varphi = \left(\frac{\partial\varphi}{\partial x_1}, \dots, \frac{\partial\varphi}{\partial x_m}\right)$$

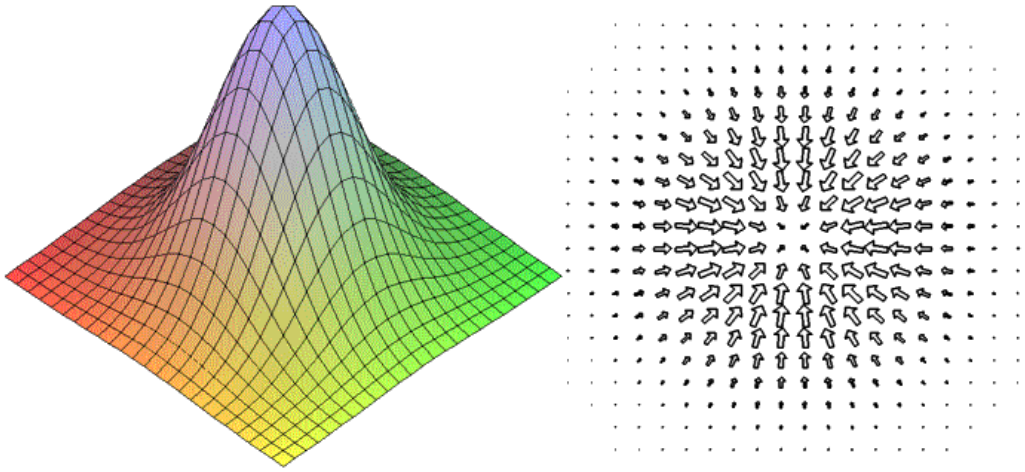


Рис. 8. Градиентное поле гауссианы.

Основная идея алгоритма градиентного спуска заключается в следующем.

Шаг 1. Сначала задаётся некоторое начальное приближение, то есть некоторая точка $\mathbf{x^{(i)}}$ $(i = 0)$ в области определения функции φ, минимум которой требуется найти.

Шаг 2. Затем вычисляется градиент в данной точке $\nabla\varphi(\mathbf{x^{(i)}})$. Он направлен в сторону наискорейшего роста функции. Для поиска минимума нужно двигаться в сторону антиградиента.

Шаг 3. Делается «шаг» в направлении антиградиента и осуществляется переход в точку $\mathbf{x^{(i+1)}}$:

$$\mathbf{x^{(i+1)}} = \mathbf{x^{(i)}} - \alpha \cdot \nabla\varphi(\mathbf{x^{(i)}}),$$

где α – некоторая положительная константа («скорость обучения»).

Шаг 4. Шаги 2-4 повторяются для $i := i + 1$ до сходимости либо до достижения максимального числа итераций.

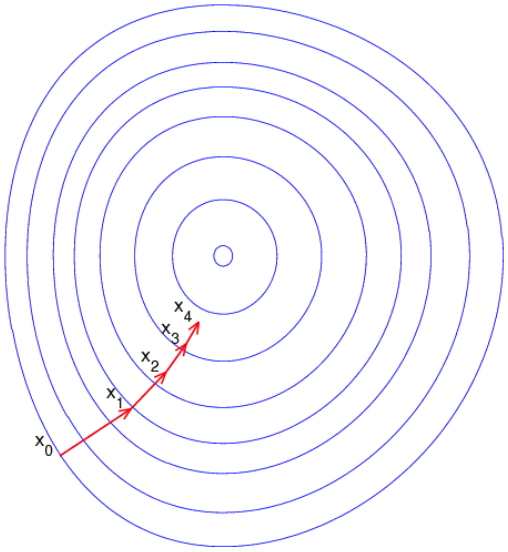


Рис. 9. Направление градиента функции в некоторой точке перпендикулярно линии уровня функции, проходящей через эту точку.

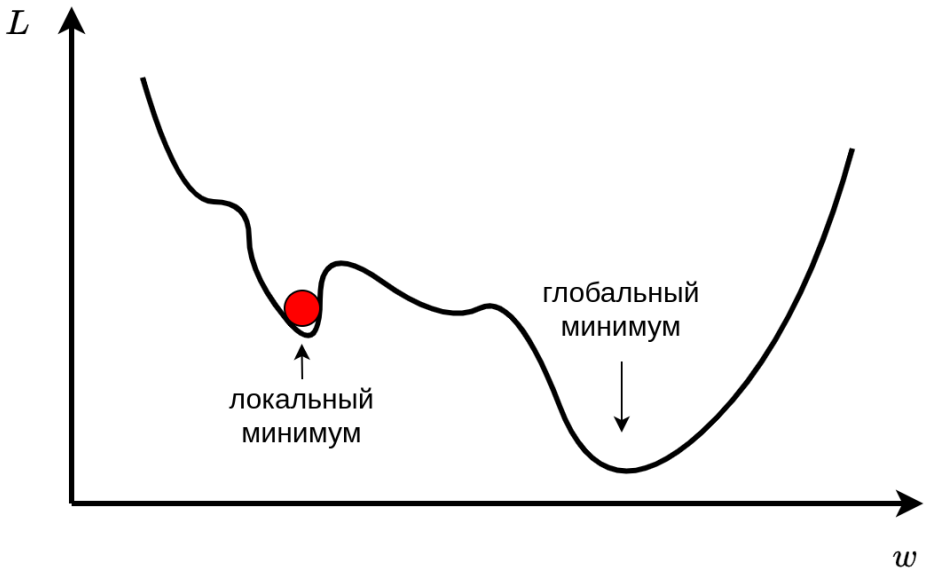
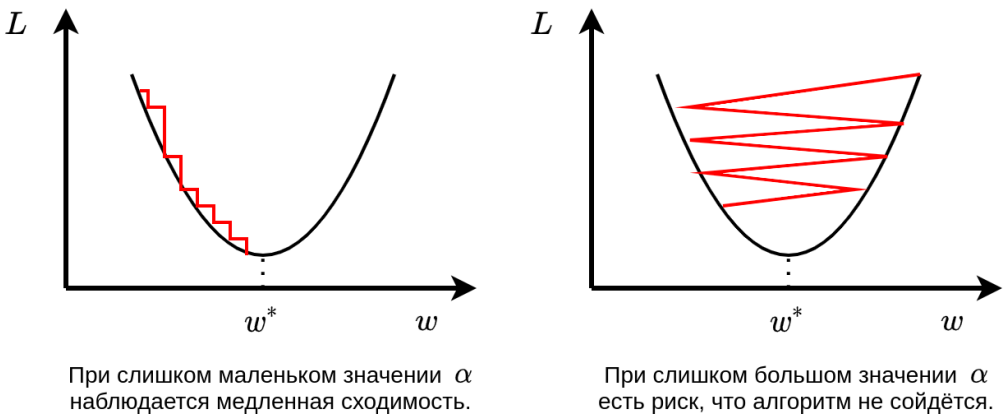


Рис. 10. Градиентный спуск может «застрять» в локальном минимуме.



При слишком маленьком значении α наблюдается медленная сходимость.

При слишком большом значении α есть риск, что алгоритм не сойдётся.

Рис. 11. Влияние параметра скорости обучения на сходимость градиентного спуска.

К недостаткам алгоритма градиентного спуска можно отнести следующее.

1. Требует дифференцируемости оптимизируемой функции.
2. Результат во многом зависит от начального приближения.
3. Чувствителен к выбору значения параметра α.
4. Застревает на «плато».

Теперь можно показать, что правило Видроу-Хоффа есть ничто иное как правило обновления весов при использовании стохастического градиентного спуска.

Минимизируемая функция имеет вид

$$L(\mathbf{w}) = \frac{1}{2n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2.$$

Рассмотрим некоторый синаптический вес w_q $(1 \le q \le m)$ и рассчитаем для него корректировку в соответствии с формулой градиентного спуска.

$$\frac{\partial L}{\partial w_q} = \frac{1}{n}\sum_{i=1}^{n}\frac{\partial L_i}{\partial \hat{y}_i}\frac{\partial \hat{y}_i}{\partial w_q} = -\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)x_{iq},$$

где $x_{iq} - q$-й элемент вектора признаков объекта i.

Так как в стохастическом градиентном спуске градиент считается на основе одного примера, возьмём из формулы выше один элемент суммы и опустим индексы для простоты. Получим

$$\frac{\partial L}{\partial w_q} = -(y - \hat{y})x_q.$$

Подставим это выражение в формулу для градиентного спуска и получим

$$w_q^{(k+1)} = w_q^{(k)} + \alpha(y - \hat{y})x_q,$$

что в точности соответствует дельта-правилу (правилу Видроу-Хоффа).

Вопросы для самопроверки

1. Перечислите основные структурные элементы биологического нейрона.
2. Расскажите, как устроена математическая модель нейрона Маккаллока—Питтса.
3. Сформулируйте общую постановку задачи машинного обучения. Какие основные типы задач вы знаете?
4. Выпишите уравнение разделяющей границы, которую строит линейный классификатор.
5. В чём заключается алгоритм обучения персептрона?
6. В чём состоит «проблема XOR»?

7. Опишите принцип работы алгоритма градиентного спуска.

Список литературы

1. Воронцов К. В. Лекции по искусственным нейронным сетям, 2007. URL: http://www.ccas.ru/voron/download/NeuralNets.pdf (дата обращения: 17.09.2020)
2. НЕЙРОН. Обработка сигналов. Пластичность. Моделирование: Фундаментальное руководство / Ю. И. Александров, К. В. Анохин, Б. Н. Безденежных и др. Тюмень: Издательство Тюменского государственного университета, 2008. 548 с.
3. Уоссермен Ф. Нейрокомпьютерная техника : Теория и практика. М.: Мир, 1992. 240 с.
4. Хайкин С. Нейронные сети: полный курс, 2-е издание.: Пер. с англ. М.: Издательский дом «Вильямс», 2006. 1104 с.