ИСКУССТВЕННЫЕ НЕЙРОННЫЕ СЕТИ

Лекция 2

Перцептрон и линейные нейронные модели

Взвешенная сумма, смещение, функции активации, геометрия классификации, правило обучения, теорема сходимости и пределы однослойного перцептрона

Преподаватель: Аршын Алтыбай

Кафедра компьютерной инженерии,
Алматинский технологический университет, Алматы, Казахстан

Учебный год: 2026

> **Связь с курсом.** На первой лекции введена функция нейрона $a = \varphi(w^\top x + b)$ и рассмотрены семейства нейросетевых архитектур. Теперь мы подробно изучаем самую простую обучаемую модель — перцептрон: от геометрии разделяющей гиперплоскости до реализации, сходимости и строгих ограничений. Эти понятия понадобятся при переходе к многослойному перцептрону в лекции 3 [6, 7].

1. Цели, результаты обучения и план

Цель: объяснить, как линейный искусственный нейрон превращается в бинарный классификатор, как меняются его параметры на основе ошибок и почему класс задач, решаемых одним перцептроном, принципиально ограничен.

По завершении лекции студент должен:

1. вычислять предактивацию $z = w^\top x + b$ и результат разных пороговых/гладких активаций;
2. интерпретировать w и b как нормаль и положение разделяющей гиперплоскости;
3. проверять линейную разделимость конечной двумерной выборки;
4. выводить правило коррекции параметров перцептрона и выполнять одну эпоху вручную;
5. формулировать и понимать условия теоремы сходимости перцептрона;

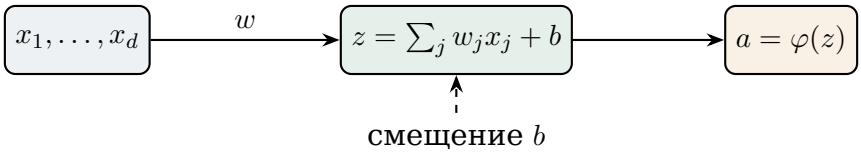
6. доказывать невозможность реализации XOR одним линейным классификатором;
7. различать классический перцептрон, линейную регрессию и логистическую модель;
8. реализовывать обучение бинарного классификатора на NumPy и анализировать ошибки.

2. Модель искусственного нейрона: скалярное произведение и смещение

Пусть $x = (x_1, \dots, x_d)^\top \in \mathbb{R}^d$ — вектор измеренных признаков объекта, $w = (w_1, \dots, w_d)^\top$ — обучаемые веса, $b \in \mathbb{R}$ — смещение. Линейная часть модели — аффинная функция

$$z = w^\top x + b = \sum_{j=1}^{d} w_j x_j + b. \tag{1}$$

Выход определяется функцией φ: $a = \varphi(z)$. Важное терминологическое различие: выражение с b — *аффинное*, а строго линейное отображение не имеет свободного члена. Однако в литературе «линейной моделью» часто называют и аффинную модель [6, 3].



2.1 Смысл весов и смещения

Знак и величина w_j характеризуют влияние признака x_j на величину z при фиксированных остальных переменных. Если $w_j > 0$, увеличение x_j повышает z; если $w_j < 0$ — уменьшает. Сравнивать абсолютные величины весов без учёта масштаба признаков нельзя: изменение единицы измерения одного признака меняет соответствующий вес. Смещение b позволяет сдвинуть границу решения относительно начала координат.

> **Частая ошибка** Больший по модулю коэффициент не всегда означает «более важный» признак: масштабы x_j и корреляции между признаками могут существенно менять интерпретацию.

2.2 Ручное вычисление

Пусть $x = (2, -1, 3)$, $w = (0.5, -2, 0.25)$, $b = -1$. Тогда

$$z = 0.5 \cdot 2 + (-2) \cdot (-1) + 0.25 \cdot 3 - 1 = 1 + 2 + 0.75 - 1 = 2.75.$$

При пороговой активации выход равен 1; при ReLU — 2.75; для сигмоиды — $\sigma(2.75) \approx 0.940$. Эти выходы имеют разный смысл: пороговая функция сразу присваивает класс, а логистическая — выдаёт непрерывную величину, которую при дополнительных предположениях можно трактовать как вероятность.

3. Классическая модель перцептрона

3.1 Два варианта кодировки классов

Часто целевая метка задаётся как $t \in \{0, 1\}$, но для математического анализа удобно использовать $y \in \{-1, +1\}$. Связь: $y = 2t - 1$, $t = (y + 1)/2$.

При метках $\{-1, +1\}$ перцептрон определяется как

$$\hat{y} = \operatorname{sign}(w^\top x + b), \qquad \operatorname{sign}(z) = \begin{cases} +1, & z \geq 0, \\ -1, & z < 0. \end{cases} \tag{2}$$

Соглашение для точки $z = 0$ является техническим и должно быть явно закреплено. Для меток $\{0, 1\}$ обычно пишут $\hat{t} = \mathbb{I}\{w^\top x + b \geq 0\}$ [1, 2].

3.2 Как устроено принятие решения

Алгоритм вычисления предсказания:

1. получить d числовых признаков;
2. умножить каждый признак на свой вес;
3. сложить произведения и прибавить b;
4. сравнить z с нулём;
5. вывести класс $+1$ или -1.

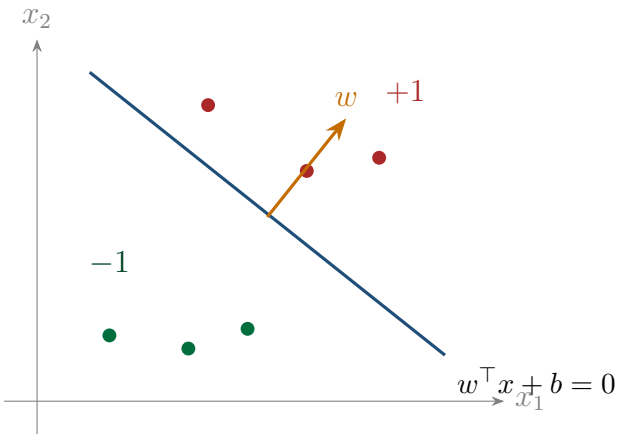
Это *inference*. При обучении модель дополнительно сравнивает $\hat{y}$ и y и изменяет параметры.

4. Геометрия: гиперплоскость, нормаль и расстояние

Набор точек, для которых предактивация равна нулю,

$$H = \{x \in \mathbb{R}^d : w^\top x + b = 0\}, \tag{3}$$

образует гиперплоскость при $w \neq 0$. Для $d = 2$ — прямую; для $d = 3$ — плоскость. Вектор w перпендикулярен этой поверхности, потому что для любых $x, x' \in H$ имеем $w^\top(x - x') = 0$.



Знаковое расстояние до гиперплоскости:

$$d_s(x, H) = \frac{w^\top x + b}{\|w\|_2}, \tag{4}$$

а обычное расстояние равно $|w^\top x + b|/\|w\|_2$. Число $y(w^\top x + b)$ называют *функциональным отступом*, а $y(w^\top x+b)/\|w\|_2$ — *геометрическим отступом*. Понятие отступа помогает понять устойчивость классификации и теорему сходимости.

4.1 Влияние изменения параметров

Изменение b при фиксированном w сдвигает границу параллельно самой себе. Изменение направления w поворачивает её. Умножение и w, и b на число $c > 0$ не меняет предсказания порогового классификатора, хотя z умножается на c. При $c < 0$ классы меняются местами.

5. Линейная разделимость и логические функции

Линейно разделимая выборка — это конечный набор $\{(x_i, y_i)\}_{i=1}^{n}$, $y_i \in \{-1, +1\}$, для которого существуют w, b такие, что

$$y_i(w^\top x_i + b) > 0 \quad \text{для всех } i. \tag{5}$$

Если неравенства выполняются, каждая точка лежит по правильную сторону разделяющей поверхности. Нулевой функциональный отступ означает точку на границе и не удовлетворяет строгому разделению.

5.1 AND и OR: явные решения

Для булевых входов $x_1, x_2 \in \{0, 1\}$ функцию AND реализуют веса $w = (1, 1)$, $b = -1.5$ с порогом $z \geq 0$. OR реализуется $w = (1, 1)$, $b = -0.5$. Проверка:

x_1	x_2	z_{AND}	$\hat{t}_{\text{AND}}$	z_{OR}	$\hat{t}_{\text{OR}}$
0	0	-1.5	0	-0.5	0
0	1	-0.5	0	0.5	1
1	0	-0.5	0	0.5	1
1	1	0.5	1	1.5	1

5.2 Почему XOR невозможно реализовать одним перцептроном

XOR принимает 1 на $(1, 0)$ и $(0, 1)$ и 0 на $(0, 0)$ и $(1, 1)$. Предположим, существуют веса w_1, w_2 и b, реализующие XOR с порогом $z \geq 0$. Тогда должны выполняться

$$\begin{aligned} b &< 0 && \text{для } (0,0), \\ w_1 + b &\geq 0 && \text{для } (1,0), \\ w_2 + b &\geq 0 && \text{для } (0,1), \\ w_1 + w_2 + b &< 0 && \text{для } (1,1). \end{aligned}$$

Из двух средних неравенств следует $w_1 + w_2 + 2b \geq 0$, то есть $w_1 + w_2 + b \geq -b > 0$, что противоречит последнему неравенству. Следовательно, **одного линейного порогового нейрона для XOR недостаточно**. Это фундаментальная мотивация нелинейных многослойных сетей [6, 7].

Ограничение Нельзя сделать XOR линейно разделимой лишь перестановкой координат или изменением масштаба при обратимом аффинном преобразовании исходного пространства. Потребуется нелинейное преобразование признаков либо многослойная нелинейная сеть.

6. Обучение перцептрона: от ошибки к обновлению

6.1 Правило обучения в кодировке $\{-1, +1\}$

Для обучающей пары (x_i, y_i) классический перцептрон проверяет знак $y_i(w^\top x_i + b)$. Если он неположителен, параметры обновляются:

$$w \leftarrow w + \eta y_i x_i, \qquad b \leftarrow b + \eta y_i, \tag{6}$$

где $\eta > 0$ — скорость обучения. При положительном отступе параметры не меняются. Для $y_i = +1$ мы увеличиваем оценку z_i; для $y_i = -1$ — уменьшаем её. Этот простой алгоритм подробно разбирается в учебных материалах Траска и Нильсена [1, 7].

6.2 Вывод через перцептронную функцию потерь

Пусть $\tilde{x} = (x^\top, 1)^\top$ и $\tilde{w} = (w^\top, b)^\top$. Определим для одного объекта

$$L_i(\tilde{w}) = \max\{0, -y_i \tilde{w}^\top \tilde{x}_i\}. \tag{7}$$

Если отступ отрицателен, градиент активной линейной ветви $\nabla L_i = -y_i \tilde{x}_i$, поэтому шаг субградиентного спуска $\tilde{w} \leftarrow \tilde{w} - \eta \nabla L_i$ совпадает с правилом (6). В точке нулевого отступа функция недифференцируема, но допустимо выбрать субградиент $-y_i \tilde{x}_i$; это соответствует обновлению на границе.

6.3 Псевдокод

Алгоритм. Инициализировать $w = 0, b = 0$; для каждой эпохи и каждого объекта вычислить $m_i = y_i(w^\top x_i + b)$. Если $m_i \le 0$, выполнить $w \leftarrow w + \eta y_i x_i$, $b \leftarrow b + \eta y_i$. Остановиться после эпохи без обновлений либо по лимиту эпох. Перемешивание данных может менять траекторию и итоговый разделитель.

7. Численный пример: одна эпоха вручную

Рассмотрим логическую функцию AND с обучающими метками в $\{-1, +1\}$ в порядке $(0,0) \mapsto -1$, $(0,1) \mapsto -1$, $(1,0) \mapsto -1$, $(1,1) \mapsto +1$. Возьмём $\eta = 1$, $w = (0,0)$, $b = 0$ и правило обновления при $m \le 0$.

1. $(0,0), y = -1$: $m = 0$, поэтому w не изменяется, $b \leftarrow -1$.
2. $(0,1), y = -1$: $z = -1$, $m = 1 > 0$, обновления нет.
3. $(1,0), y = -1$: $z = -1$, $m = 1 > 0$, обновления нет.
4. $(1,1), y = +1$: $z = -1$, $m = -1$, обновляем $w \leftarrow (1,1)$, $b \leftarrow 0$.

После этой эпохи параметры $w = (1,1), b = 0$, но обучение **ещё не завершено**: например, объект $(0,1)$ классифицируется как $+1$. Этот пример показывает, почему важно различать *одну эпоху* и *сходимость*.

8. Теорема сходимости перцептрона

8.1 Формулировка

Используем расширенные векторы $\tilde{x}_i = (x_i^\top, 1)^\top$ и $\tilde{w} = (w^\top, b)^\top$. Предположим, что

1. данные линейно разделимы, то есть существует единичный вектор u и $\gamma > 0$, для которых $y_i u^\top \tilde{x}_i \ge \gamma$ для всех i;
2. $\|\tilde{x}_i\|_2 \le R$ для всех i;
3. $\tilde{w}_0 = 0$ и при ошибке выполняется обновление $\tilde{w} \leftarrow \tilde{w} + y_i \tilde{x}_i$ (здесь $\eta = 1$).

Тогда число обновлений M не превосходит $(R/\gamma)^2$. Теорема гарантирует *конечное число коррекций* для линейно разделимой конечной выборки, но не единственность результата и не хорошее качество на новых данных [6, 3].

8.2 Доказательство по шагам

После M ошибочных обновлений имеем

$$u^\top \tilde{w}_M = \sum_{k=1}^{M} y_{i_k} u^\top \tilde{x}_{i_k} \geq M\gamma.$$

С другой стороны, для ошибочно классифицированного объекта $y_i \tilde{w}^\top \tilde{x}_i \leq 0$, поэтому

$$\begin{aligned} \|\tilde{w} + y_i \tilde{x}_i\|_2^2 &= \|\tilde{w}\|_2^2 + 2 y_i \tilde{w}^\top \tilde{x}_i + \|\tilde{x}_i\|_2^2 \\ &\leq \|\tilde{w}\|_2^2 + R^2. \end{aligned}$$

По индукции $\|\tilde{w}_M\|_2^2 \leq M R^2$. Неравенство Коши–Буняковского даёт

$$M\gamma \leq u^\top \tilde{w}_M \leq \|u\|_2 \|\tilde{w}_M\|_2 \leq R\sqrt{M}.$$

Для $M > 0$ получаем $\sqrt{M} \leq R/\gamma$, значит

$$\boxed{M \leq \frac{R^2}{\gamma^2}}. \tag{8}$$

Математический смысл: большой отступ γ относительно R позволяет получить более сильную верхнюю границу числа обновлений. Если данные неразделимы, предпосылка нарушается и данный вывод неприменим.

8.3 Что теорема не утверждает

Она не обещает, что алгоритм найдёт разделитель с максимальным отступом. Она не гарантирует устойчивость к выбросам или высокое качество на неизвестных данных. Если классы пересекаются, возможны повторяющиеся обновления без остановки, поэтому практическая программа должна иметь предел числа эпох.

9. Полная реализация перцептрона на NumPy

Ниже класс поддерживает метки $\{-1, +1\}$ и возвращает историю числа обновлений. Код намеренно не использует готовый классификатор, чтобы показать сам механизм обучения [1, 5].

Пример 1. Перцептрон без готовых ML-библиотек

```
import numpy as np

```

Пример 1 (продолжение)

```
class SimplePerceptron:
    def __init__(self, lr=1.0, epochs=100):
        self.lr = lr
        self.epochs = epochs

    def fit(self, X, y):
        X = np.asarray(X, dtype=float)
        y = np.asarray(y, dtype=int)
        assert set(np.unique(y)).issubset({-1, 1})
        self.w = np.zeros(X.shape[1], dtype=float)
        self.b = 0.0
        self.updates_ = []
        for epoch in range(self.epochs):
            mistakes = 0
            for xi, yi in zip(X, y):
                margin = yi * (np.dot(self.w, xi) + self.b)
                if margin <= 0:
                    self.w += self.lr * yi * xi
                    self.b += self.lr * yi
                    mistakes += 1
            self.updates_.append(mistakes)
            if mistakes == 0:
                break
        return self

    def predict(self, X):
        X = np.asarray(X, dtype=float)
        scores = X @ self.w + self.b
        return np.where(scores >= 0, 1, -1)

X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
y_and = np.array([-1, -1, -1, 1])
model = SimplePerceptron(lr=1.0, epochs=100).fit(X, y_and)
print('predictions:', model.predict(X).tolist())
print('correct:', bool(np.all(model.predict(X) == y_and)))
print('epochs:', len(model.updates_))
```

Ожидаемый вывод

```
predictions: [-1, -1, -1, 1]
correct: True
```

Важно: число эпох и итоговые веса зависят от порядка входных объектов и соглашения о нулевом отступе. Для диагностирования полезно строить график `updates_`, фиксировать генератор случайных чисел при перемешивании и записывать параметры эксперимента.

10. Эксперимент: AND, OR и XOR

Пример 2. Сравнение трёх логических задач

```
X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
tasks = {
    'AND': np.array([-1, -1, -1, 1]),
    'OR':  np.array([-1, 1, 1, 1]),
    'XOR': np.array([-1, 1, 1, -1]),
}
for name, y in tasks.items():
    m = SimplePerceptron(lr=1.0, epochs=30).fit(X, y)
    accuracy = (m.predict(X) == y).mean()
    print(name, 'accuracy:', accuracy,
          'last updates:', m.updates_[-1])
```

Ожидается, что AND и OR на обучающих данных достигнут точности 1.0 и пройдут эпоху без обновлений. Для XOR полного разделения не произойдёт: ограничение доказано алгебраически выше. Последние обновления и оценка точности для XOR зависят от порядка обхода и числа эпох; их не следует представлять как универсальные константы.

11. Перцептрон, линейная регрессия и логистическая регрессия

Эти модели используют одну и ту же аффинную часть $z = w^\top x + b$, но решают разные задачи или обучаются по-разному.

Характеристика	Перцептрон	Линейная регрессия	Логистическая регрессия
Целевой выход	Метка класса ± 1	Непрерывное число	Вероятность класса 1
Выход модели	Порог $\operatorname{sign}(z)$	z	$\sigma(z)$
Критерий обучения	Ошибки/перцепт потеря	Обычно MSE	Обычно log loss
Граница решения	Линейная	Не основной объект	Линейная при пороге 0.5
Вероятностный смысл	Не предусмотрен	Не обязательно	Есть при статистической модели

11.1 Почему логистическая модель дифференцируема

Логистическая функция

$$\sigma(z) = \frac{1}{1+e^{-z}}, \qquad \sigma'(z) = \sigma(z)(1-\sigma(z))$$

гладкая. Для бинарной кросс-энтропии

$$L = -\big[t \log p + (1-t)\log(1-p)\big], \qquad p = \sigma(w^\top x + b),$$

цепное правило даёт $\partial L/\partial w = (p-t)x$ и $\partial L/\partial b = p-t$. В отличие от порогового выхода здесь удобно применять градиентные методы. Подробный вывод этих формул станет основой лекций о функциях потерь и обратном распространении ошибки [7, 6].

12. Масштабирование признаков и устойчивость

Пусть один вход — возраст человека в годах (20–70), другой — нормализованная доля (0–1). При равных начальных весах вклад первого признака в z может доминировать только из-за масштаба. Стандартизация выполняется по статистикам *только обучающей* выборки:

$$x'_j = \frac{x_j - \mu_j}{s_j}, \qquad s_j > 0. \tag{9}$$

Здесь μ_j и s_j — среднее и стандартное отклонение соответствующего признака на train. Те же значения применяют к validation, test и новым данным. Пересчитывать их отдельно на test — типичный источник утечки данных. Однако при наличии константного признака следует отдельно обработать $s_j = 0$ [3].

> **Практическое замечание** Стандартизация не изменяет принципиальную линейную разделимость при обратимом диагональном масштабировании. Она может влиять на скорость обучения и численные свойства алгоритма.

13. Граница модели: шум, выбросы и перекрытие классов

Два объекта могут иметь одинаковый вектор признаков и разные метки из-за ошибки разметки или неполной наблюдаемости. Тогда строгая линейная разделимость невозможна при любом w, b. Даже если обучающая выборка разделима, новый объект может попасть не в свой класс. Поэтому необходимо исследовать ошибки на независимой выборке и сравнивать перцептрон с разумным baseline [3, 4].

Среди практических ограничений: чувствительность к выбросам; зависимость итогового разделителя от порядка примеров; отсутствие калиброванных вероятностей; неспособность представить сложную нелинейную границу без преобразования признаков.

14. Как преодолеть XOR: нелинейные признаки

Введём дополнительный признак $u = x_1 x_2$. Для XOR на булевых точках правильное решение задаёт

$$z = x_1 + x_2 - 2x_1 x_2 - \tfrac{1}{2}. \tag{10}$$

Проверим: для $(0,0)$ имеем -0.5; для $(1,0)$ и $(0,1)$ — 0.5; для $(1,1)$ — -0.5. Мы получили линейный классификатор *в расширенном пространстве* (x_1, x_2, u), но преобразование $u = x_1 x_2$ является нелинейным по исходным признакам.

> **Переход к лекции 3.** Многослойная нейронная сеть способна самостоятельно формировать нелинейные промежуточные представления. Именно это, а не одно лишь увеличение числа весов, делает нелинейные многослойные модели выразительнее одного перцептрона.

15. Практическая задача: двумерный классификато

Постройте набор из 200 точек двух линейно разделимых классов, например с центрами $(-2,-2)$ и $(2,2)$ и небольшим шумом. Разделите выборку на train/test, обучите свой перцептрон и изобразите границу $w_1x_1+w_2x_2+b=0$ вместе с точками. Затем усилите шум и проанализируйте изменения ошибок. Обязательно сравните train и test accuracy, чтобы не путать запоминание обучающих примеров с обобщением [3].

Пример 3. Генерация данных и оценка на независимом тесте

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix

rng = np.random.default_rng(42)
X_neg = rng.normal(loc=(-2., -2.), scale=0.7, size=(100, 2))
X_pos = rng.normal(loc=( 2.,  2.), scale=0.7, size=(100, 2))
X = np.vstack([X_neg, X_pos])
y = np.array([-1] * 100 + [1] * 100)
Xtr, Xte, ytr, yte = train_test_split(
    X, y, test_size=0.25, stratify=y, random_state=42)
model = SimplePerceptron(lr=1.0, epochs=100).fit(Xtr, ytr)
yp = model.predict(Xte)
print('accuracy:', accuracy_score(yte, yp))
print('confusion matrix:\n', confusion_matrix(yte, yp))
```

Нужно отдельно сформулировать, как поведёт себя алгоритм, если добавить повторяющиеся точки с противоречивыми метками. Опирайтесь не на предположение о «плохом оптимизаторе», а на нарушение условия существования разделяющей гиперплоскости.

16. Типичные ошибки при реализации

1. **Несогласованная кодировка меток.** Правило $w \leftarrow w+\eta y x$ корректно для $y \in \{-1,+1\}$, но не в таком виде для $y \in \{0,1\}$.
2. **Забыто смещение.** Без b граница обязана проходить через начало координат.
3. **Неверное направление поправки.** Для метки -1 надо уменьшать значение z, а не увеличивать.
4. **Отсутствует ограничение числа эпох.** На XOR или шумных неразделимых данных программа может не завершаться.
5. **Смешаны обучение и тестирование.** Сообщать о точности только на обучающих данных недостаточно.
6. **Неправильная вероятностная интерпретация.** Выход ± 1 и расстояние до границы не являются автоматически вероятностями.
7. **Нормализация на всех данных.** Это даёт модели информацию о тестовом распределении.

17. Итоги лекции

- Перцептрон вычисляет знак аффинной формы $w^\top x + b$.
- Его геометрия — две полуплоскости (в общем случае полупространства), разделённые гиперплоскостью.
- Простейшее обучение изменяет веса и смещение только при неправильной или граничной классификации.
- Для линейно разделимой выборки число ошибочных обновлений конечно и оценивается через R/γ.
- XOR не линейно разделим в исходном двумерном пространстве, но может разделяться после нелинейного преобразования признаков.
- Перцептрон, линейная регрессия и логистическая регрессия разделяют аффинную основу, но отличаются выходами и обучающими критериями.
- Следующая тема — многослойный перцептрон и обучение нелинейным представлениям.

18. Контрольные вопросы

1. Что такое предактивация? Чем она отличается от выхода нейрона?
2. Почему отображение $w^\top x + b$ с $b \neq 0$ строго говоря не линейное?
3. Как влияет b на расположение разделяющей прямой?
4. Запишите уравнение границы решения для $d = 2$ и $d = 3$.
5. Почему w является нормалью к разделяющей гиперплоскости?
6. Что называют функциональным и геометрическим отступами?
7. Когда конечная выборка линейно разделима?
8. Постройте перцептрон для AND и OR, проверьте все строки таблицы истинности.
9. Докажите, что XOR не реализуется одним линейным пороговым нейроном.
10. Чем кодировка $\{0, 1\}$ отличается от $\{-1, +1\}$?
11. Сформулируйте правило обновления весов при ошибке.
12. Выведите это правило из перцептронной функции потерь.
13. Что такое эпоха? Почему одна эпоха может быть недостаточна?
14. Какие предпосылки нужны для теоремы сходимости?
15. Как получить оценку $M \leq (R/\gamma)^2$?
16. Что не гарантирует теорема сходимости?
17. Чем перцептрон отличается от логистической регрессии?
18. Почему sigmoid удобнее пороговой функции для градиентного обучения?
19. Почему масштабирование признаков выполняют по train-выборке?
20. Каким нелинейным признаком можно превратить XOR в линейно разделимую задачу?

19. Самостоятельная работа

Задание A (вычислительное). Для $x = (1, 2, -1)$, $w = (-0.2, 0.7, 0.4)$, $b = -0.3$ вычислите z и результат пороговой и сигмоидальной активаций. Проверьте ответ в NumPy.

Задание B (геометрическое). Для $w = (2, -1)$, $b = -3$ изобразите границу решения и найдите знаковое расстояние до неё для точек $(0, 0)$, $(2, 1)$, $(3, 3)$.

Задание C (алгоритмическое). Измените исходный порядок обучающих примеров в программе для AND. Сравните число обновлений и итоговые параметры. Объясните наблюдение.

Задание D (программное). Сравните самостоятельно написанный перцептрон и логистическую регрессию из scikit-learn на одной двумерной выборке. Сохраните матрицы ошибок и прокомментируйте отличия.

Задание E (углублённое). Повторите доказательство теоремы сходимости, выписав отдельно применение неравенства Коши–Буняковского и роль условия $\gamma > 0$.

20. Библиография и дальнейшее чтение

Как и в первой лекции, приведены семь основных источников. Для углубления материалов этой лекции особенно полезны [1, 2, 7, 6]; для практической реализации и инженерного сравнения — [3, 4, 5].

References

[1] Trask A. W. *Grokking Deep Learning*. Manning Publications, 2019. Русское издание: «Грокаем глубокое обучение». https://www.manning.com/books/grokking-deep-learning.

[2] Rashid T. *Make Your Own Neural Network*. 2016. Русское издание: «Глубокое обучение. Лёгкая теория и практика на Python» (название может варьироваться по изданию). https://makeyourownneuralnetwork.blogspot.com/.

[3] Géron A. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. 3rd ed., O'Reilly Media, 2022. Русское издание: «Прикладное машинное обучение с помощью Scikit-Learn, Keras и TensorFlow». https://www.oreilly.com/library/view/hands-on-machine-learning/9781098125967/.

[4] Chollet F. *Deep Learning with Python*. 2nd ed., Manning Publications, 2021. Русское издание: «Глубокое обучение на Python». https://www.manning.com/books/deep-learning-with-python-second-edition.

[5] Stevens E., Antiga L., Viehmann T. *Deep Learning with PyTorch*. Manning Publications, 2020. Русское издание известно как «Глубокое обучение с PyTorch»; формулировка «Сублинейное программирование» не является частью оригинального названия. https://www.manning.com/books/deep-learning-with-pytorch.

[6] Goodfellow I., Bengio Y., Courville A. *Deep Learning*. MIT Press, 2016. Русское издание: «Глубокое обучение». https://www.deeplearningbook.org/.

[7] Nielsen M. A. *Neural Networks and Deep Learning*. 2015. Бесплатный онлайн-учебник: «Нейронные сети и глубокое обучение». http://neuralnetworksanddeeplearning.com/.

Рекомендуемая последовательность чтения: сначала Траск [1] и Рашид [2] для понимания вычислений «с нуля»; затем Нильсен [7] для математики обратного распространения и Жерон [3] для рабочего ML-процесса. Для последующего углубления — Шолле [4], Стивенс и соавторы [5], Гудфеллоу и соавторы [6].