Калькулятор многослойного перцептрона

Техническое руководство и математические основы

Русский

Вольфганг Бёме

wolfgang.boehme@economic-modeling.de

www.economic-modeling.de

2 ноября 2026 г.

Документация на этом сайте разработана при поддержке ИИ Claude (Anthropic).

Содержание

1 Введение **5**
1.1 Обзор 5
1.2 Области применения 5

2 Математические основы **5**
2.1 Архитектура сети 5
2.2 Модель нейрона 6
2.3 Матричная нотация 6

3 Функции активации **7**
3.1 Сигмоидная функция 7
3.2 ReLU (выпрямленная линейная единица) 7
3.3 Функция tanh 7
3.4 Линейная функция 8

4 Прямое распространение (Forward Propagation) **8**
4.1 Алгоритм 8
4.2 Пример вычисления 8

5 Функции потерь **9**
5.1 Среднеквадратичная ошибка (MSE) 9
5.2 Бинарная кросс-энтропия (BCE) 9

6 Обратное распространение ошибки (Backpropagation) **9**
6.1 Основной принцип 10
6.2 Градиент выходного слоя 10
6.3 Градиенты скрытых слоёв 10
6.4 Градиенты весов и смещений 10
6.5 Алгоритм обратного распространения 11

7 Градиентный спуск **11**
7.1 Стандартный градиентный спуск 11
7.2 Пакетный градиентный спуск 11
7.3 Мини-пакетный градиентный спуск 11

8 Алгоритм обучения **12**
8.1 Полный цикл обучения 12
8.2 Реализация в калькуляторе v20 12

9 Нормализация данных **13**
9.1 Мотивация 13
9.2 Min-Max нормализация 13
9.3 Денормализация 13
9.4 Параметры нормализации 13
9.5 Параметры только по обучающим данным 13

10 Инициализация весов **14**
10.1 Случайная инициализация 14
10.2 Инициализация Ксавье/Глоро (рекомендуется) 14
10.3 Реализация в калькуляторе v20 14

11 Гиперпараметры **14**
11.1 Скорость обучения η 14
11.2 Количество эпох 15
11.3 Размер пакета B 15
11.4 Архитектура сети 15

12 Разбиение данных и бэктестинг **15**
12.1 Разбиение на обучение и тест 15
12.2 Значение предыдущего дня как вход (лаг 1) 16
12.3 Метрики 16
12.4 Эталоны сравнения 16
12.5 Проверка экстраполяции 16

13 Использование калькулятора **17**
13.1 Рабочий процесс 17
13.2 Пример: кросс-курсы (UZS) 17

14 Устранение неполадок **18**
14.1 Обучение не сходится 18
14.2 Переобучение 18
14.3 Недообучение 18
14.4 Численная нестабильность 19

15 Математические теоремы **19**
15.1 Теорема об универсальной аппроксимации 19
15.2 Сходимость градиентного спуска 19

16 Детали реализации **19**
16.1 Матричные операции 19
16.2 Анализ сложности 20

17 Визуализации **20**
17.1 Граф сети 20
17.2 Активные нейроны 20
17.3 Кривая потерь 21
17.4 Распределение весов 21

18 Форматы экспорта **21**

19 Возможности расширения **22**
19.1 Планируемые функции 22
19.2 Dropout 22
19.3 Momentum 22
19.4 Adam Optimizer 22

20 Литература **23**
20.1 Основополагающие работы 23
20.2 Учебники 23

21 Приложение **23**
21.1 Сборник формул 23
21.2 Глоссарий 24
21.3 Обозначения 25

22 Заключение **25**

Аннотация

Данное техническое руководство описывает математические основы, алгоритмы и детали реализации калькулятора многослойного перцептрона (MLP). Документ предназначен для пользователей, желающих глубже понять теорию нейронных сетей, а также для разработчиков, которые хотят разобраться в реализации или расширить её. Подробно рассматриваются все используемые формулы, алгоритм обратного распространения ошибки, функции активации, функции потерь и методы оптимизации.

1. Введение

1.1. Обзор

Многослойный перцептрон (MLP) — фундаментальная искусственная нейронная сеть, состоящая из нескольких слоёв нейронов. Данный калькулятор реализует полноценный MLP со следующими характеристиками:

- Настраиваемое количество входных нейронов (1–10)
- Переменное количество скрытых слоёв (2–5)
- Гибкое количество нейронов на скрытый слой (1–20)
- Один выходной нейрон для регрессии или бинарной классификации
- Различные функции активации для каждого слоя
- Обучение на основе обратного распространения ошибки
- Визуализация хода обучения в реальном времени
- Воспроизводимые результаты благодаря начальному значению (seed)
- Разбиение Train/Test (хронологическое или равномерное) с автоматическим бэктестингом и эталонами сравнения
- Ввод и вывод всегда в исходных значениях; нормализация выполняется внутри
- Проверка на экстраполяцию за пределы обучающего диапазона

1.2. Области применения

MLP подходят для широкого круга задач:

- **Регрессия:** прогнозирование непрерывных значений (например, валютные курсы, цены)
- **Бинарная классификация:** отнесение к одному из двух классов (например, спам/не спам)
- **Аппроксимация функций:** обучение сложным нелинейным зависимостям
- **Распознавание образов:** выявление закономерностей в данных

2. Математические основы

2.1. Архитектура сети

Определение 2.1 (Многослойный перцептрон)**.** Многослойный перцептрон — ориентированный ациклический граф, состоящий из L слоёв, где:

- Слой 0: входной слой с n_0 нейронами

- Слой l для $1 \leq l < L-1$: скрытые слои с n_l нейронами
- Слой $L-1$: выходной слой с n_{L-1} нейронами

Архитектура сети описывается вектором:

$$\mathcal{N} = (n_0, n_1, n_2, \ldots, n_{L-1}) \tag{1}$$

Пример: $(7, 16, 8, 1)$ описывает сеть с 7 входными нейронами, двумя скрытыми слоями из 16 и 8 нейронов соответственно и одним выходным нейроном.

2.2. Модель нейрона

Определение 2.2 (Искусственный нейрон)**.** Искусственный нейрон j в слое l вычисляет свой выход в два этапа:

1. Взвешенная сумма (пред-активация):

$$z_j^{(l)} = \sum_{i=1}^{n_{l-1}} w_{ji}^{(l)} \cdot a_i^{(l-1)} + b_j^{(l)} \tag{2}$$

2. Активация (пост-активация):

$$a_j^{(l)} = f^{(l)}(z_j^{(l)}) \tag{3}$$

где:

- $w_{ji}^{(l)}$ — вес от нейрона i слоя $l-1$ к нейрону j слоя l
- $b_j^{(l)}$ — смещение нейрона j в слое l
- $a_i^{(l-1)}$ — активация нейрона i предыдущего слоя
- $f^{(l)}$ — функция активации в слое l

2.3. Матричная нотация

Для эффективных вычислений используем матричную нотацию. Для слоя l:

$$\mathbf{z}^{(l)} = \mathbf{W}^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)} \tag{4}$$

$$\mathbf{a}^{(l)} = f^{(l)}(\mathbf{z}^{(l)}) \tag{5}$$

где:

- $\mathbf{W}^{(l)} \in \mathbb{R}^{n_l \times n_{l-1}}$ — матрица весов
- $\mathbf{b}^{(l)} \in \mathbb{R}^{n_l}$ — вектор смещений
- $\mathbf{a}^{(l)} \in \mathbb{R}^{n_l}$ — вектор активаций
- $f^{(l)}$ применяется поэлементно к вектору

3. Функции активации

Функции активации позволяют сети обучаться нелинейным зависимостям.

3.1. Сигмоидная функция

Сигмоидная функция отображает значения в интервал $(0, 1)$:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{6}$$

Производная:

$$\sigma'(z) = \sigma(z) \cdot (1 - \sigma(z)) \tag{7}$$

Свойства:

- Выход в диапазоне от 0 до 1
- Гладкая и дифференцируемая
- Интерпретируется как вероятность
- Проблема: затухающие градиенты при $|z| \gg 0$

3.2. ReLU (выпрямленная линейная единица)

$$\text{ReLU}(z) = \max(0, z) = \begin{cases} z & \text{если } z > 0 \\ 0 & \text{если } z \leq 0 \end{cases} \tag{8}$$

Производная:

$$\text{ReLU}'(z) = \begin{cases} 1 & \text{если } z > 0 \\ 0 & \text{если } z \leq 0 \end{cases} \tag{9}$$

Свойства:

- Очень эффективна в вычислении
- Нет затухающего градиента при $z > 0$
- Стандарт для глубоких сетей
- Проблема: «мёртвые ReLU» при $z \leq 0$

3.3. Функция tanh

$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} = \frac{e^{2z} - 1}{e^{2z} + 1} \tag{10}$$

Производная:

$$\tanh'(z) = 1 - \tanh^2(z) \tag{11}$$

Свойства:

- Выход в диапазоне от -1 до 1

- Центрирована вокруг 0
- Более сильные градиенты, чем у сигмоиды
- Аналогичная проблема затухающих градиентов, как у сигмоиды

3.4. Линейная функция

$$\text{linear}(z) = z \tag{12}$$

Производная:

$$\text{linear}'(z) = 1 \tag{13}$$

Применение:

- Типично для выходного слоя при регрессии
- Диапазон значений не ограничен

4. Прямое распространение (Forward Propagation)

4.1. Алгоритм

Прямой проход вычисляет выход сети для заданного входа $\mathbf{x}$.

Algorithm 1 Прямое распространение

Require: Вектор входа $\mathbf{x} \in \mathbb{R}^{n_0}$, веса $\{\mathbf{W}^{(l)}\}_{l=1}^{L-1}$, смещения $\{\mathbf{b}^{(l)}\}_{l=1}^{L-1}$
Ensure: Выход $\mathbf{a}^{(L-1)}$

$\mathbf{a}^{(0)} \leftarrow \mathbf{x}$ ▷ Входной слой
for $l = 1$ до $L-1$ **do** ▷ Для каждого слоя
  $\mathbf{z}^{(l)} \leftarrow \mathbf{W}^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}$ ▷ Взвешенная сумма
  $\mathbf{a}^{(l)} \leftarrow f^{(l)}(\mathbf{z}^{(l)})$ ▷ Активация
end for
return $\mathbf{a}^{(L-1)}$ ▷ Выход

4.2. Пример вычисления

Дана сеть с архитектурой $(2, 3, 1)$ и входом $\mathbf{x} = (0{,}5;\ 0{,}8)^T$.
Слой 1 (вход $\rightarrow$ скрытый):

$$\mathbf{W}^{(1)} = \begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{pmatrix}, \quad \mathbf{b}^{(1)} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} \tag{14}$$

$$\mathbf{z}^{(1)} = \mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)} = \begin{pmatrix} w_{11} \cdot 0{,}5 + w_{12} \cdot 0{,}8 + b_1 \\ w_{21} \cdot 0{,}5 + w_{22} \cdot 0{,}8 + b_2 \\ w_{31} \cdot 0{,}5 + w_{32} \cdot 0{,}8 + b_3 \end{pmatrix} \tag{15}$$

$$\mathbf{a}^{(1)} = \text{ReLU}(\mathbf{z}^{(1)}) \tag{16}$$

Слой 2 (скрытый → выход):

$$z^{(2)} = w_1^{(2)} a_1^{(1)} + w_2^{(2)} a_2^{(1)} + w_3^{(2)} a_3^{(1)} + b^{(2)} \tag{17}$$

$$a^{(2)} = \sigma(z^{(2)}) \tag{18}$$

5. Функции потерь

Функция потерь $\mathcal{L}$ измеряет расхождение между прогнозом $\hat{y}$ и истинным значением y.

5.1. Среднеквадратичная ошибка (MSE)

Для задач регрессии:

$$\mathcal{L}_{\text{MSE}}(y, \hat{y}) = \frac{1}{2}(y - \hat{y})^2 \tag{19}$$

Производная:

$$\frac{\partial \mathcal{L}_{\text{MSE}}}{\partial \hat{y}} = -(y - \hat{y}) = \hat{y} - y \tag{20}$$

Для набора данных из N примеров:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \tag{21}$$

5.2. Бинарная кросс-энтропия (BCE)

Для бинарной классификации:

$$\mathcal{L}_{\text{BCE}}(y, \hat{y}) = -[y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})] \tag{22}$$

Производная:

$$\frac{\partial \mathcal{L}_{\text{BCE}}}{\partial \hat{y}} = -\frac{y}{\hat{y}} + \frac{1 - y}{1 - \hat{y}} = \frac{\hat{y} - y}{\hat{y}(1 - \hat{y})} \tag{23}$$

Для набора данных:

$$\mathcal{L}_{\text{BCE}} = -\frac{1}{N} \sum_{i=1}^{N} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)] \tag{24}$$

6. Обратное распространение ошибки (Backpropagation)

Обратное распространение — алгоритм вычисления градиентов функции потерь по всем параметрам сети.

6.1. Основной принцип

Применяем правило цепочки для распространения градиентов от выходного слоя назад по сети.

Теорема 6.1 (Правило цепочки для нейронных сетей)**.** Производная функции потерь по весам:

$$\frac{\partial \mathcal{L}}{\partial w_{ji}^{(l)}} = \frac{\partial \mathcal{L}}{\partial z_j^{(l)}} \cdot \frac{\partial z_j^{(l)}}{\partial w_{ji}^{(l)}} \tag{25}$$

6.2. Градиент выходного слоя

Для выходного слоя $L-1$ с одним нейроном:

$$\delta^{(L-1)} = \frac{\partial \mathcal{L}}{\partial z^{(L-1)}} = \frac{\partial \mathcal{L}}{\partial a^{(L-1)}} \cdot \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \tag{26}$$

При MSE-потерях и сигмоидной активации:

$$\delta^{(L-1)} = (a^{(L-1)} - y) \cdot \sigma'(z^{(L-1)}) = (a^{(L-1)} - y) \cdot a^{(L-1)} \cdot (1 - a^{(L-1)}) \tag{27}$$

6.3. Градиенты скрытых слоёв

Для скрытых слоёв $l < L-1$:

$$\delta_j^{(l)} = \left(\sum_{k=1}^{n_{l+1}} w_{kj}^{(l+1)} \delta_k^{(l+1)} \right) \cdot f'^{(l)}(z_j^{(l)}) \tag{28}$$

В матричной нотации:

$$\boldsymbol{\delta}^{(l)} = \left[(\mathbf{W}^{(l+1)})^T \boldsymbol{\delta}^{(l+1)} \right] \odot f'^{(l)}(\mathbf{z}^{(l)}) \tag{29}$$

где $\odot$ — произведение Адамара (поэлементное умножение).

6.4. Градиенты весов и смещений

Веса:

$$\frac{\partial \mathcal{L}}{\partial w_{ji}^{(l)}} = \delta_j^{(l)} \cdot a_i^{(l-1)} \tag{30}$$

Смещения:

$$\frac{\partial \mathcal{L}}{\partial b_j^{(l)}} = \delta_j^{(l)} \tag{31}$$

В матричной форме:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(l)}} = \boldsymbol{\delta}^{(l)} (\mathbf{a}^{(l-1)})^T \tag{32}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{b}^{(l)}} = \boldsymbol{\delta}^{(l)} \tag{33}$$

6.5. Алгоритм обратного распространения

Algorithm 2 Обратное распространение ошибки

Require: Вход $\mathbf{x}$, цель y, активации $\{\mathbf{a}^{(l)}\}$, пред-активации $\{\mathbf{z}^{(l)}\}$
Ensure: Градиенты $\{\nabla_{\mathbf{W}^{(l)}}\mathcal{L}\}$, $\{\nabla_{\mathbf{b}^{(l)}}\mathcal{L}\}$

// Выходной слой
$\boldsymbol{\delta}^{(L-1)} \leftarrow \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{(L-1)}} \odot f'^{(L-1)}(\mathbf{z}^{(L-1)})$

// Скрытые слои (в обратном порядке)
for $l = L-2$ вниз до 1 **do**
    $\boldsymbol{\delta}^{(l)} \leftarrow [(\mathbf{W}^{(l+1)})^T \boldsymbol{\delta}^{(l+1)}] \odot f'^{(l)}(\mathbf{z}^{(l)})$
end for

// Вычисление градиентов
for $l = 1$ до $L-1$ **do**
    $\nabla_{\mathbf{W}^{(l)}}\mathcal{L} \leftarrow \boldsymbol{\delta}^{(l)}(\mathbf{a}^{(l-1)})^T$
    $\nabla_{\mathbf{b}^{(l)}}\mathcal{L} \leftarrow \boldsymbol{\delta}^{(l)}$
end for
return Градиенты

7. Градиентный спуск

7.1. Стандартный градиентный спуск

Обновление параметров со скоростью обучения η:

$$\mathbf{W}^{(l)} \leftarrow \mathbf{W}^{(l)} - \eta \cdot \nabla_{\mathbf{W}^{(l)}}\mathcal{L} \tag{34}$$

$$\mathbf{b}^{(l)} \leftarrow \mathbf{b}^{(l)} - \eta \cdot \nabla_{\mathbf{b}^{(l)}}\mathcal{L} \tag{35}$$

7.2. Пакетный градиентный спуск

Для набора данных $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$:

$$\nabla_{\mathbf{W}^{(l)}}\mathcal{L}_{\text{total}} = \frac{1}{N}\sum_{i=1}^{N} \nabla_{\mathbf{W}^{(l)}}\mathcal{L}(\mathbf{x}_i, y_i) \tag{36}$$

7.3. Мини-пакетный градиентный спуск

Разбиваем набор данных на пакеты размером B:

$$\nabla_{\mathbf{W}^{(l)}}\mathcal{L}_{\text{batch}} = \frac{1}{B}\sum_{i \in \text{batch}} \nabla_{\mathbf{W}^{(l)}}\mathcal{L}(\mathbf{x}_i, y_i) \tag{37}$$

Преимущества:

- Быстрее сходится, чем пакетный GD
- Меньше требований к памяти, чем пакетный GD
- Регуляризирующий эффект за счёт шума

8. Алгоритм обучения

8.1. Полный цикл обучения

Algorithm 3 Обучение нейронной сети

Require: Набор данных $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, параметры сети, скорость обучения η, эпохи E, размер пакета B
Ensure: Обученные параметры

for эпоха $= 1$ до E **do**
  Перемешать $\mathcal{D}$ ▷ Перемешивание набора данных
  $\mathcal{L}_{\text{epoch}} \leftarrow 0$
  for каждый пакет из $\mathcal{D}$ **do**
    $\nabla_{\mathbf{W}}\mathcal{L} \leftarrow 0$, $\nabla_{\mathbf{b}}\mathcal{L} \leftarrow 0$ ▷ Инициализация градиентов
    for каждого $(\mathbf{x}, y)$ в пакете **do**
      $\hat{y} \leftarrow$ ForwardProp(**x**) ▷ Прямой проход
      $\mathcal{L}_{\text{epoch}} \leftarrow \mathcal{L}_{\text{epoch}} + \mathcal{L}(y, \hat{y})$
      $\nabla_{\mathbf{W}}\mathcal{L} \leftarrow \nabla_{\mathbf{W}}\mathcal{L} +$ BackProp(**x**, y) ▷ Накопление градиентов
    end for
    $\nabla_{\mathbf{W}}\mathcal{L} \leftarrow \nabla_{\mathbf{W}}\mathcal{L}/B$ ▷ Среднее
    $\mathbf{W} \leftarrow \mathbf{W} - \eta \cdot \nabla_{\mathbf{W}}\mathcal{L}$ ▷ Обновление
    $\mathbf{b} \leftarrow \mathbf{b} - \eta \cdot \nabla_{\mathbf{b}}\mathcal{L}$ ▷ Обновление
  end for
  $\mathcal{L}_{\text{epoch}} \leftarrow \mathcal{L}_{\text{epoch}}/N$
  Записать прогресс и потери
end for

8.2. Реализация в калькуляторе v20

Калькулятор использует *стохастический* градиентный спуск: веса обновляются после каждого отдельного примера. Размер батча лишь структурирует цикл и не влияет на результат. Для каждого примера

$$\boldsymbol{\delta}^{(L-1)} = (\mathbf{y} - \hat{\mathbf{y}}) \odot f'^{(L-1)}, \qquad \mathbf{W}^{(l)} \leftarrow \mathbf{W}^{(l)} + \eta\, \boldsymbol{\delta}^{(l)} (\mathbf{a}^{(l-1)})^T, \qquad \mathbf{b}^{(l)} \leftarrow \mathbf{b}^{(l)} + \eta\, \boldsymbol{\delta}^{(l)}. \quad (38)$$

Ошибка передаётся на предыдущий слой уже с обновлёнными весами. Производная f' вычисляется по активации a (напр. $a(1-a)$ для сигмоиды). Градиент на выходе имеет одинаковый вид для MSE и BCE; выбранная функция потерь определяет отображаемый Loss (BCE с ограничением $\hat{y}$ интервалом $[10^{-4}, 1 - 10^{-4}]$). Loss эпохи — среднее по всем обучающим примерам в нормированной шкале.

Перемешивание: если включена опция «Перемешивать порядок обучающих данных в каждой эпохе», порядок в начале каждой эпохи переставляется методом Фишера–Йетса. Генератор для этого инициализируется значением seed $+ 7919$. Без перемешивания при хронологических данных сеть всегда видит последними самые свежие строки.

9. Нормализация данных

9.1. Мотивация

При данных с сильно различающимися масштабами могут возникнуть проблемы:

- Численная нестабильность (переполнение/потеря значимости)
- Медленная сходимость
- Затухающие или взрывающиеся градиенты

9.2. Min-Max нормализация

Для признака j со значениями $x_j^{(1)}, \ldots, x_j^{(N)}$:

$$x_j^{\text{norm}} = \frac{x_j - \min(x_j)}{\max(x_j) - \min(x_j)} \tag{39}$$

Свойства:

- Масштабирует в интервал $[0, 1]$
- Сохраняет относительные расстояния
- Все признаки на одной шкале

9.3. Денормализация

Для прогнозов необходимо обратное преобразование:

$$x_j^{\text{orig}} = x_j^{\text{norm}} \cdot (\max(x_j) - \min(x_j)) + \min(x_j) \tag{40}$$

9.4. Параметры нормализации

Для каждого признака сохраняем:

$$\text{params}_j = \begin{cases} \min_j = \min(x_j^{(1)}, \ldots, x_j^{(N)}) \\ \max_j = \max(x_j^{(1)}, \ldots, x_j^{(N)}) \\ \text{range}_j = \max_j - \min_j \end{cases} \tag{41}$$

9.5. Параметры только по обучающим данным

Минимум и максимум каждого входа и цели определяются *исключительно по обучающим данным* и без изменений применяются к тестовым данным, ручному вводу и пакетным файлам. Поэтому нормированные тестовые значения могут лежать вне $[0, 1]$ — это намеренно и соответствует реальному прогнозу, при котором будущее во время обучения неизвестно. Выход сети денормализуется по параметрам цели; все метрики и отображения — в исходных значениях.

10. Инициализация весов

10.1. Случайная инициализация

Веса инициализируются случайно для нарушения симметрии:

$$w_{ji}^{(l)} \sim \mathcal{U}(-1, 1) \tag{42}$$

где $\mathcal{U}(-1, 1)$ — равномерное распределение на $[-1, 1]$.

10.2. Инициализация Ксавье/Глоро (рекомендуется)

Для лучшей сходимости:

$$w_{ji}^{(l)} \sim \mathcal{N}\left(0, \frac{2}{n_{l-1} + n_l}\right) \tag{43}$$

или равномерно:

$$w_{ji}^{(l)} \sim \mathcal{U}\left(-\sqrt{\frac{6}{n_{l-1} + n_l}}, \sqrt{\frac{6}{n_{l-1} + n_l}}\right) \tag{44}$$

10.3. Реализация в калькуляторе v20

Веса и смещения выбираются равномерно:

$$w_{ji}^{(l)},\, b_j^{(l)} \sim \mathcal{U}\Big(-\sqrt{1/n_{l-1}},\ \sqrt{1/n_{l-1}}\Big)\,. \tag{45}$$

Случайные числа даёт генератор *mulberry32* (32-битное состояние, период 2^{32}), инициализированный значением seed с вкладки «Конфигурация» (по умолчанию 42). Те же seed, данные и настройки дают точно те же начальные веса, тот же порядок перемешивания и тот же результат.

11. Гиперпараметры

11.1. Скорость обучения η

Значение: шаг при градиентном спуске

- Слишком большая: нестабильность, осцилляция, расходимость
- Слишком малая: медленная сходимость, локальные минимумы
- Типичные значения: от $0{,}001$ до $0{,}1$

$$\eta_{\text{opt}} \approx \frac{1}{\lambda_{\max}(\mathbf{H})} \tag{46}$$

где $\lambda_{\max}(\mathbf{H})$ — наибольшее собственное значение матрицы Гессиана.

11.2. Количество эпох

Значение: сколько раз сеть проходит по всему набору данных

- Слишком мало: недообучение, плохое обобщение
- Слишком много: переобучение на тренировочных данных
- Типичные значения: от 1000 до 10000 в зависимости от задачи

11.3. Размер пакета B

Значение: количество примеров на одно обновление градиента

- Малый ($B = 1$): стохастический GD, высокий шум, быстрые обновления
- Большой ($B = N$): пакетный GD, стабильный, медленные обновления
- Средний ($B = 32$–256): хороший компромисс

$$\text{Обновлений за эпоху} = \left\lceil \frac{N}{B} \right\rceil \tag{47}$$

11.4. Архитектура сети

Количество скрытых слоёв:

- 0–1: линейные/простые нелинейные задачи
- 2–3: сложные нелинейные задачи (стандарт)
- ≥ 4: глубокое обучение, очень сложные задачи

Нейроны на слой:

$$n_{\text{hidden}} \approx \frac{n_{\text{input}} + n_{\text{output}}}{2} \tag{48}$$

или по теореме Колмогорова–Нильсена:

$$n_{\text{hidden}} \geq 2 \cdot n_{\text{input}} + 1 \tag{49}$$

12. Разбиение данных и бэктестинг

12.1. Разбиение на обучение и тест

При N строках и доле обучения q (по умолчанию 80 %) хронологическое разбиение использует $N_{\text{train}} = \lfloor qN \rfloor$ обучающих строк (251 строка, 80 % $\to$ 200 обучение, 51 тест).

- **Хронологически:** первые N_{train} строк — обучение, последние — тест. Единственный реалистичный вариант для временных рядов.
- **Равномерно:** каждая k-я строка — тест, $k = \max\bigl(2, \mathrm{round}(1/(1-q))\bigr)$, т. е. при 80 % каждая 5-я строка.

12.2. Значение предыдущего дня как вход (лаг 1)

По желанию значение цели из предыдущей строки добавляется как дополнительный вход $x_t^{\text{lag}} = y_{t-1}$; первая строка отбрасывается. Тогда сеть может воспроизвести как минимум правило «как вчера».

12.3. Метрики

Для n строк с фактическим значением y_i и прогнозом $\hat{y}_i$ (исходные значения), $e_i = \hat{y}_i - y_i$:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|e_i|, \qquad \text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}e_i^2}, \qquad \text{MAPE} = \frac{100\,\%}{n}\sum_{i=1}^{n}\left|\frac{e_i}{y_i}\right|, \tag{50}$$

$$R^2 = 1 - \frac{\sum_i e_i^2}{\sum_i (y_i - \bar{y})^2}, \qquad \text{Bias} = \frac{1}{n}\sum_{i=1}^{n} e_i, \qquad \text{Max} = \max_i |e_i|. \tag{51}$$

$\bar{y}$ — среднее оцениваемых строк; $R^2 < 0$ означает, что прогноз хуже этого постоянного среднего. Для классификации (выход Sigmoid, цель 0/1) добавляется точность с порогом 0,5.

Калькулятор сообщает о *возможном переобучении*, если $\text{RMSE}_{\text{тест}} > 2\,\text{RMSE}_{\text{обуч}}$ (регрессия) или точность на тесте более чем на 10 процентных пунктов ниже, чем на обучении.

12.4. Эталоны сравнения

В тестовом периоде сеть сравнивается с тремя простыми правилами:

$$\hat{y}_t^{\text{сред}} = \bar{y}_{\text{обуч}}, \qquad \hat{y}_t^{\text{посл}} = y_{t_0-1}, \qquad \hat{y}_t^{\text{вчера}} = y_{t-1}, \tag{52}$$

где y_{t_0-1} — последнее обучающее значение перед началом теста t_0. При $r = \text{RMSE}_{\text{MLP}} / \min_k \text{RMSE}_k$ вывод таков: «лучше» при $r < 0{,}95$, «примерно так же» при $0{,}95 \le r \le 1{,}05$, иначе «хуже». Для классификации эталоном служит самый частый класс обучающих данных.

12.5. Проверка экстраполяции

Для каждого входа j обучающий диапазон — $[m_j, M_j]$ (минимум и максимум обучающих данных). Отклонение значения x:

$$o_j(x) = \begin{cases} x - m_j & x < m_j \\ x - M_j & x > M_j \\ 0 & \text{иначе,} \end{cases} \qquad \text{относительно: } \frac{|o_j(x)|}{M_j - m_j}\cdot 100\,\%. \tag{53}$$

Строка считается экстраполяцией, если $o_j \neq 0$ хотя бы для одного j. Каждый вход проверяется отдельно; необычные *комбинации* внутри отдельных диапазонов не распознаются.

13. Использование калькулятора

13.1. Рабочий процесс

Шаг 1: загрузить данные (вкладка «Данные»)

1. Выбрать пример — рекомендуемые настройки применяются, сеть создаётся — или загрузить свой CSV (запятая или точка с запятой, строка заголовков; текстовые столбцы и даты игнорируются).
2. Выбрать целевой столбец, нормализацию, опцию лага и разбиение Train/Test.

Шаг 2: проверить конфигурацию (вкладка «Конфигурация»)

1. Блок рекомендаций: рекомендуемое/текущее по каждому параметру; правило: скрытый 1 = 2–4 $\times$ входов, скрытый 2 $\approx$ вдвое меньше.
2. Выход Linear и MSE для регрессии, Sigmoid и BCE для классификации.
3. Зафиксировать seed, чтобы сравнивать настройки.

Шаг 3: обучение (вкладка «Обучение») — наблюдать за кривой потерь.
Шаг 4: бэктестинг (вкладка «Бэктестинг») — тестовые метрики, эталоны сравнения, экстраполяция.
Шаг 5: прогноз и экспорт — ввод в исходных значениях, пакетные файлы, экспорт CSV и PNG.

13.2. Пример: кросс-курсы (UZS)

Данные: 251 торговый день (синтетические), входы USD/UZS, EUR/USD, GBP/UZS, CNY/UZS, цель EUR/UZS:

$$\text{EUR/UZS}_t = \text{EUR/USD}_t \cdot \text{USD/UZS}_t \cdot (1 + \varepsilon_t), \qquad \varepsilon_t \sim \mathcal{N}(0;\, 0{,}0004^2). \tag{54}$$

GBP/UZS и CNY/UZS не влияют на цель.
Настройки: нормализация включена, архитектура $(4, 12, 6, 1)$, ReLU, ReLU, Linear, MSE, $\eta = 0{,}003$, 2000 эпох, seed 42, хронологическое разбиение 200/51 — 145 параметров.
Результат (тестовый период, 51 день):

Метод	MAE	RMSE	MAPE	R^2
MLP	5,34	6,44	0,037 %	0,9959
Как вчера	50,92	62,12	0,354 %	0,6209
Последнее значение	80,15	100,91	0,558 %	$-0{,}0005$
Среднее обучения	85,88	104,31	0,597 %	$-0{,}0690$

$r = 6{,}44/62{,}12 \approx 0{,}10$: сеть примерно на 90 % лучше лучшего простого правила. На обучении MAE 4,68 и MAPE 0,033 % — переобучения нет. 27 тестовых строк — экстраполяция (USD/UZS до 14,8 % ниже, GBP/UZS до 24,7 % выше обучающего диапазона); их MAE (5,03) не больше, чем у остальных строк (5,68).

Контрпример «Валютные данные (UZS)»: с USD, CNY, GBP, CHF как входами для EUR (архитектура $(4, 16, 8, 1)$, 3000 эпох) тест даёт MAPE 0,557 % и $R^2 = -4{,}24$ — хуже правила «как вчера» (MAPE 0,117 %). С входом-лагом сеть достигает MAPE 0,120 % при $r \approx 0{,}99$, т. е. лишь качества этого правила. Причина: дневные изменения EUR и входов практически не коррелированы.

14. Устранение неполадок

14.1. Обучение не сходится

Симптомы: Потери остаются постоянными или сильно осциллируют
Причины и решения:

- Слишком высокая скорость обучения ⇒ уменьшить до $0{,}001$ или $0{,}0001$
- Данные не нормализованы ⇒ включить нормализацию
- Сеть слишком мала ⇒ добавить нейроны или слои
- Затухающие градиенты ⇒ использовать ReLU вместо Sigmoid

14.2. Переобучение

Симптомы: Низкие потери на обучении, но плохое обобщение
Решения:

- Собрать больше данных обучения
- Использовать меньшую сеть
- Остановиться раньше (меньше эпох)
- Добавить регуляризацию (будущая версия)

14.3. Недообучение

Симптомы: Высокие потери при обучении, сеть не учится
Решения:

- Увеличить сеть (больше слоёв/нейронов)
- Обучать больше эпох
- Увеличить скорость обучения
- Использовать более сложные функции активации

14.4. Численная нестабильность

Симптомы: NaN или бесконечность в потерях/весах
Решения:

- Обязательно нормализовать данные
- Сильно уменьшить скорость обучения
- Реализовать ограничение градиента (gradient clipping)
- При BCE: держать sigmoid-выходы подальше от 0/1

15. Математические теоремы

15.1. Теорема об универсальной аппроксимации

Теорема 15.1 (Цыбенко, 1989; Хорник и др., 1989)**.** Сеть прямого распространения с одним скрытым слоем и конечным числом нейронов может приближённо представить любую непрерывную функцию на компактном подмножестве $\mathbb{R}^n$ с произвольной точностью, при условии что функция активации нелинейна, ограничена и монотонно возрастает.

Формально: для любой непрерывной функции $f : K \to \mathbb{R}$ на компакте $K \subset \mathbb{R}^n$ и $\epsilon > 0$ существует сеть F такая, что:

$$\sup_{x \in K} |f(x) - F(x)| < \epsilon \tag{55}$$

15.2. Сходимость градиентного спуска

Теорема 15.2 (Сходимость при выпуклой функции потерь)**.** Для L-Липшиц-непрерывной выпуклой функции $\mathcal{L}$ градиентный спуск с постоянной скоростью обучения $\eta \leq \frac{1}{L}$ сходится к глобальному минимуму:

$$\lim_{t \to \infty} \mathcal{L}(\mathbf{w}_t) = \mathcal{L}^* \tag{56}$$

Примечание: Для нейронных сетей функция потерь, как правило, не является выпуклой, поэтому гарантируется только сходимость к локальным минимумам.

16. Детали реализации

16.1. Матричные операции

Реализация использует класс матриц со следующими операциями:
Умножение матриц:

$$(\mathbf{AB})_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj} \tag{57}$$

Сложность: $\mathcal{O}(n^3)$

Транспонирование:

$$(\mathbf{A}^T)_{ij} = a_{ji} \tag{58}$$

Произведение Адамара:

$$(\mathbf{A} \odot \mathbf{B})_{ij} = a_{ij} \cdot b_{ij} \tag{59}$$

16.2. Анализ сложности

Прямой проход:

$$\mathcal{O}\left(\sum_{l=1}^{L-1} n_l \cdot n_{l-1}\right) = \mathcal{O}(W) \tag{60}$$

где W — общее число весов.

Обратный проход:

$$\mathcal{O}(W) \tag{61}$$

Обучение (одна эпоха):

$$\mathcal{O}(N \cdot W) \tag{62}$$

где N — количество обучающих примеров.

17. Визуализации

17.1. Граф сети

Визуализация показывает:

- **Нейроны:** круги, с цветовой кодировкой по слоям
- **Веса:** линии между нейронами
- **Цвет:** красный = положительные веса, синий = отрицательные
- **Толщина:** пропорциональна величине веса

17.2. Активные нейроны

Для ввода **x** (текущие значения на вкладке «Прогноз», иначе последняя строка данных) каждому нейрону присваивается статус по $|a_j^{(l)}|$:

$$\begin{aligned} &\text{неактивный: } |a| = 0, && \text{слабый: } 0 < |a| < 0{,}2, \\ &\text{нормальный: } 0{,}2 \le |a| < 0{,}8, && \text{сильный: } |a| \ge 0{,}8. \end{aligned} \tag{63}$$

По умолчанию график показывает только нормальные нейроны, слабые — бледно, сильные — по выбору; неактивные отображаются серой точкой. Связь $i \to j$ рисуется толщиной $|w_{ji}^{(l)} a_i^{(l-1)}|$ (красная — положительный, синяя — отрицательный вклад), если вклад составляет не менее 2 % наибольшего. Так видно, какими путями формируется конкретный прогноз.

17.3. Кривая потерь

Кривая потерь $\mathcal{L}(t)$ показывает:

- Ось Y: значение потерь
- Ось X: номер эпохи
- Идеальный вид: монотонно убывающая и сходящаяся

Интерпретация:

- Крутой спуск в начале: хорошее обучение
- Плато: достигнута сходимость
- Осцилляция: слишком высокая скорость обучения
- Рост: переобучение или численные проблемы

17.4. Распределение весов

Гистограмма всех весов:

- Хорошо обученная: нормальное распределение вокруг 0
- Проблемы: очень малые или очень большие веса
- Мёртвые нейроны: много весов ровно 0

18. Форматы экспорта

Все экспорты — CSV-файлы (UTF-8 с BOM) или PNG-графики.

Файл	**Столбцы**	**Формат**
Веса	Layer, From_Neuron, To_Neuron, Weight, Bias	запятая, точка
Архитектура	Параметр, Значение	запятая, точка
Активации	Слой, Нейрон, Значение_активации	запятая, точка
История обучения	Эпоха, Потери	запятая, точка
Бэктестинг	Строка, Использование, входы, Прогноз, Факт, Ошибка, Ошибка_%	точка с запятой, десятичная запятая
Пакетные прогнозы	Индекс, входы, Прогноз [, Факт, Ошибка, Ошибка_%] [, Класс]	точка с запятой, десятичная запятая
Пакетный файл тестового периода	входы, цель (заголовок с именами столбцов)	запятая, точка

```
Epoch,Loss
1,0.524892
2,0.487321
...
```

19. Возможности расширения

19.1. Планируемые функции

- **Dropout:** регуляризация путём случайного отключения нейронов
- **Momentum:** ускоренная сходимость
- **Adam Optimizer:** адаптивные скорости обучения
- **Batch Normalization:** более стабильное обучение
- **Многовыходная модель:** несколько выходных нейронов
- **Свёрточные слои:** для изображений
- **Рекуррентные слои:** для последовательностей
- **Кросс-валидация:** улучшенная оценка
- **Ранняя остановка:** автоматическая остановка при переобучении
- **Планирование скорости обучения:** адаптация в ходе обучения

19.2. Dropout

$$\tilde{a}_j^{(l)} = \begin{cases} 0 & \text{с вероятностью } p \\ \frac{a_j^{(l)}}{1-p} & \text{с вероятностью } 1-p \end{cases} \tag{64}$$

19.3. Momentum

$$v_t = \gamma v_{t-1} + \eta \nabla_{\mathbf{W}} \mathcal{L} \tag{65}$$

$$\mathbf{W}_t = \mathbf{W}_{t-1} - v_t \tag{66}$$

где $\gamma \in [0, 1]$ — параметр импульса (типично: $\gamma = 0{,}9$).

19.4. Adam Optimizer

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla_{\mathbf{W}} \mathcal{L} \tag{67}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)(\nabla_{\mathbf{W}} \mathcal{L})^2 \tag{68}$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \tag{69}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t} \tag{70}$$

$$\mathbf{W}_t = \mathbf{W}_{t-1} - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t \tag{71}$$

Типичные значения: $\beta_1 = 0{,}9$, $\beta_2 = 0{,}999$, $\epsilon = 10^{-8}$

20. Литература

20.1. Основополагающие работы

1. Rosenblatt, F. (1958). «The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain»
2. Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). «Learning representations by back-propagating errors»
3. Cybenko, G. (1989). «Approximation by superpositions of a sigmoidal function»
4. Hornik, K., Stinchcombe, M., & White, H. (1989). «Multilayer feedforward networks are universal approximators»
5. LeCun, Y., Bengio, Y., & Hinton, G. (2015). «Deep learning»

20.2. Учебники

1. Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.
2. Nielsen, M. A. (2015). *Neural Networks and Deep Learning*. Determination Press.
3. Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

21. Приложение

21.1. Сборник формул

Функции активации:

$$\sigma(z) = \frac{1}{1+e^{-z}}, \quad \sigma'(z) = \sigma(z)(1-\sigma(z)) \tag{72}$$

$$\text{ReLU}(z) = \max(0, z), \quad \text{ReLU}'(z) = \mathbb{1}_{z>0} \tag{73}$$

$$\tanh(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}, \quad \tanh'(z) = 1 - \tanh^2(z) \tag{74}$$

Функции потерь:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{2}(y-\hat{y})^2, \quad \frac{\partial \mathcal{L}_{\text{MSE}}}{\partial \hat{y}} = \hat{y} - y \tag{75}$$

$$\mathcal{L}_{\text{BCE}} = -[y \log \hat{y} + (1-y)\log(1-\hat{y})], \quad \frac{\partial \mathcal{L}_{\text{BCE}}}{\partial \hat{y}} = \frac{\hat{y}-y}{\hat{y}(1-\hat{y})} \tag{76}$$

Обратное распространение:

$$\boldsymbol{\delta}^{(L-1)} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{(L-1)}} \odot f'^{(L-1)}(\mathbf{z}^{(L-1)}) \tag{77}$$

$$\boldsymbol{\delta}^{(l)} = [(\mathbf{W}^{(l+1)})^T \boldsymbol{\delta}^{(l+1)}] \odot f'^{(l)}(\mathbf{z}^{(l)}) \tag{78}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(l)}} = \boldsymbol{\delta}^{(l)} (\mathbf{a}^{(l-1)})^T \tag{79}$$

Градиентный спуск:

$$\mathbf{W}^{(l)} \leftarrow \mathbf{W}^{(l)} - \eta \cdot \frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(l)}} \tag{80}$$

Нормализация:

$$x_{\text{norm}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{81}$$

$$x_{\text{orig}} = x_{\text{norm}} \cdot (x_{\max} - x_{\min}) + x_{\min} \tag{82}$$

21.2. Глоссарий

Activation Function (функция активации) Нелинейная функция, применяемая к пред-активациям

Backpropagation (обратное распространение) Алгоритм вычисления градиентов

Batch (пакет) Подмножество обучающего набора данных

Bias (смещение) Аддитивный параметр нейрона (порог)

Epoch (эпоха) Один полный проход по обучающему набору данных

Forward Pass (прямой проход) Вычисление выхода сети для входных данных

Gradient (градиент) Вектор частных производных

Hidden Layer (скрытый слой) Слой между входным и выходным

Learning Rate (скорость обучения) Размер шага при обновлении параметров

Loss Function (функция потерь) Функция измерения ошибки

Neuron (нейрон) Вычислительная единица сети

Overfitting (переобучение) Модель слишком подстраивается под обучающие данные

Weight (вес) Параметр, взвешивающий связи между нейронами

21.3. Обозначения

Символ	Значение
L	Количество слоёв
n_l	Количество нейронов в слое l
$w_{ji}^{(l)}$	Вес от нейрона i (слой $l-1$) к j (слой l)
$b_j^{(l)}$	Смещение нейрона j в слое l
$z_j^{(l)}$	Пред-активация нейрона j в слое l
$a_j^{(l)}$	Активация нейрона j в слое l
$\mathbf{W}^{(l)}$	Матрица весов для слоя l
$\mathbf{b}^{(l)}$	Вектор смещений для слоя l
$f^{(l)}$	Функция активации в слое l
$\mathcal{L}$	Функция потерь
η	Скорость обучения
$\delta_j^{(l)}$	Сигнал ошибки нейрона j в слое l
$\nabla_{\mathbf{W}}\mathcal{L}$	Градиент потерь по весам
$\odot$	Произведение Адамара (поэлементное умножение)
$\mathbb{1}$	Индикаторная функция

22. Заключение

Данное руководство представляет всестороннее математическое и практическое введение в многослойные перцептроны. Реализация в калькуляторе следует описанным здесь алгоритмам и позволяет применять теорию непосредственно на практике.

По вопросам, предложениям по улучшению или сообщениям об ошибках обращайтесь к разработчикам или воспользуйтесь системой обратной связи в калькуляторе.

Удачи в экспериментах с нейронными сетями!