



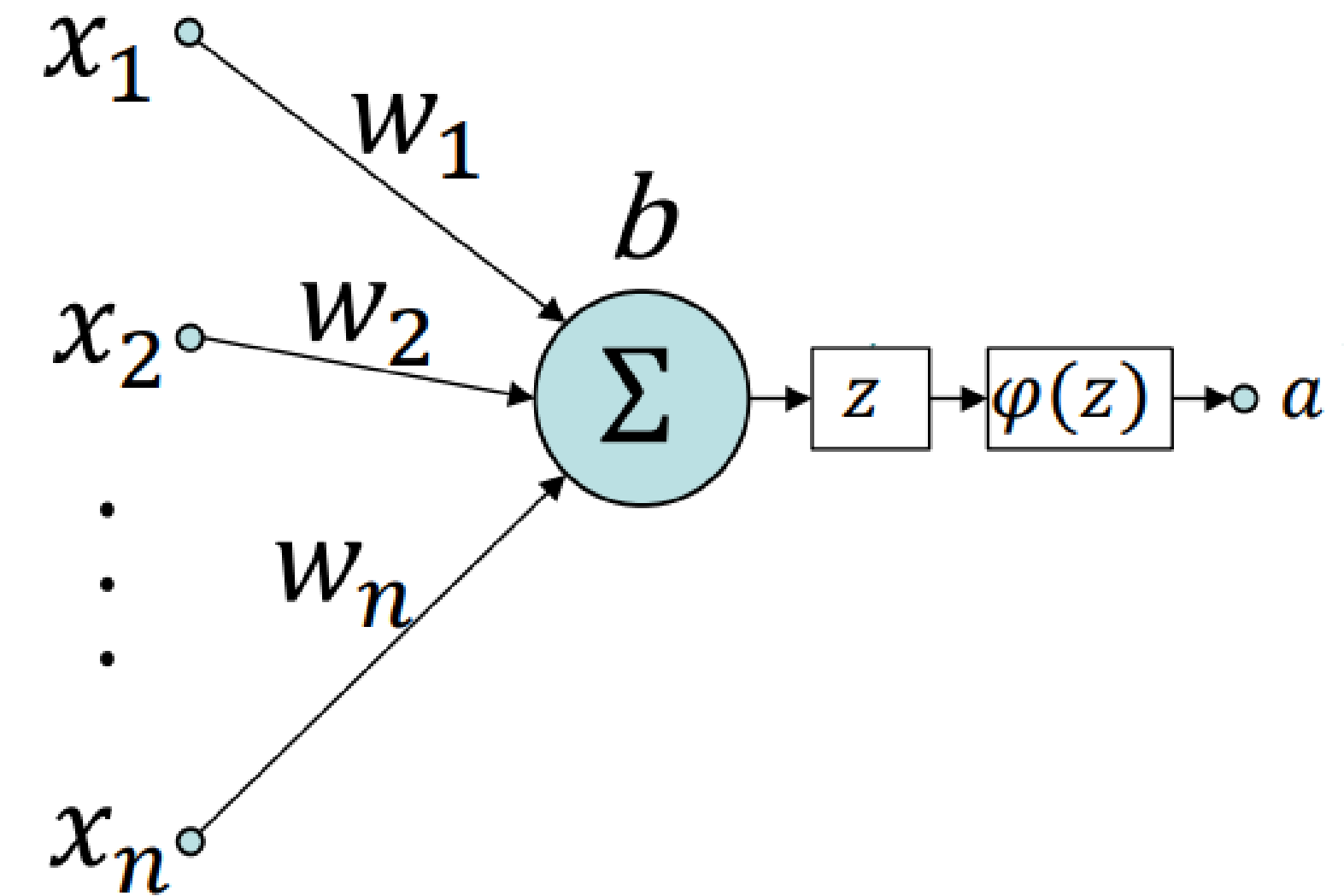
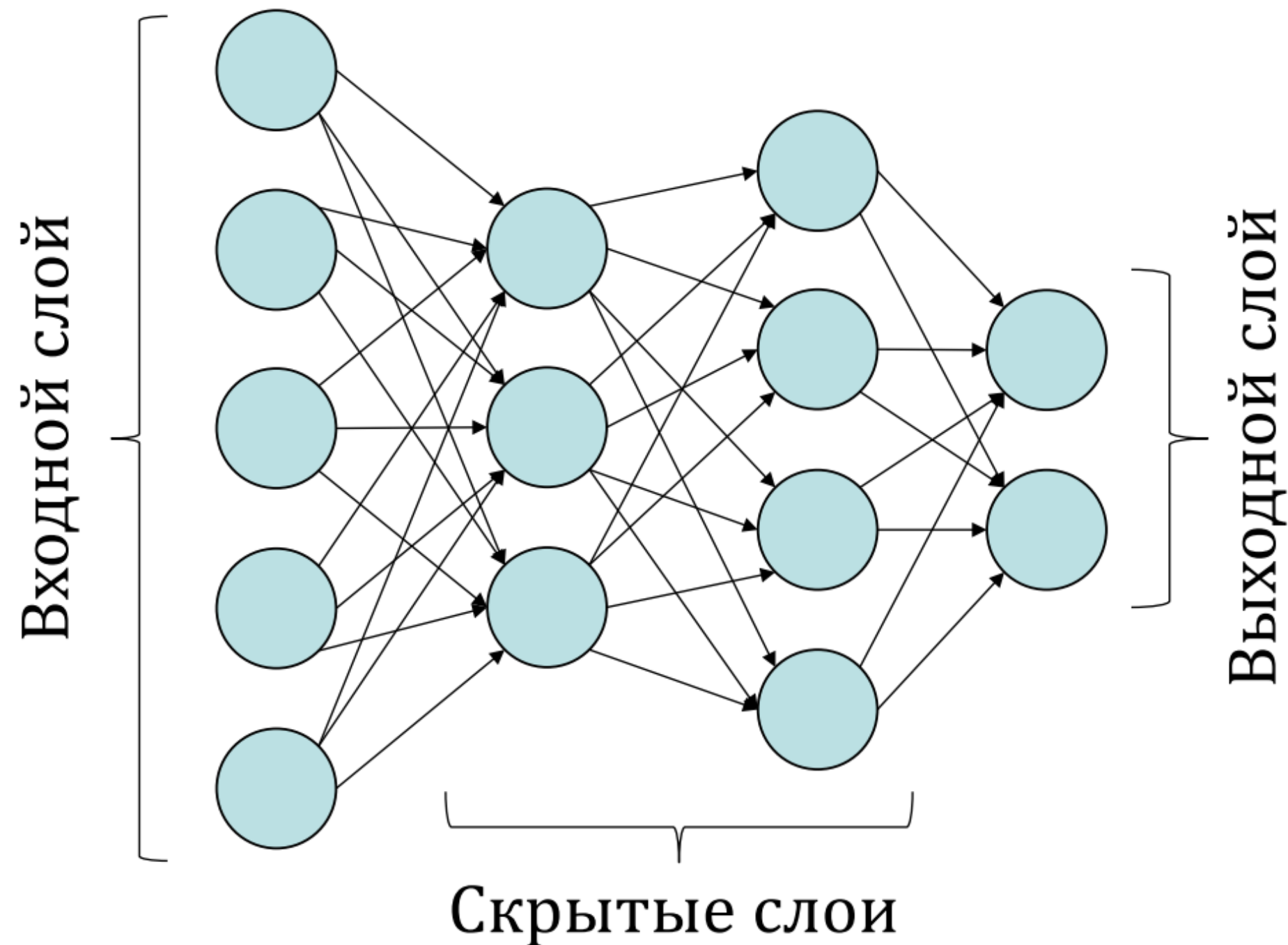
Глубокие сети прямого распространения. Автоматическое дифференцирование

Корнет Мария Евгеньевна
старший преподаватель кафедры
Инженерная кибернетика

План

- ✓ Многослойный персептрон
- ✓ Обучение
- ✓ Сведение задачи обучения к дифференцированию
- ✓ Дифференцирование сложной функции
- ✓ Обратное распространение ошибки

Многослойный персептрон



$x = (x_1, x_2, \dots, x_n)$ – входные сигналы: $x_i \in \{0,1\}$

$w = (w_1, w_2, \dots, w_n)$ – синаптические веса: $w_i \in \mathbb{R}$

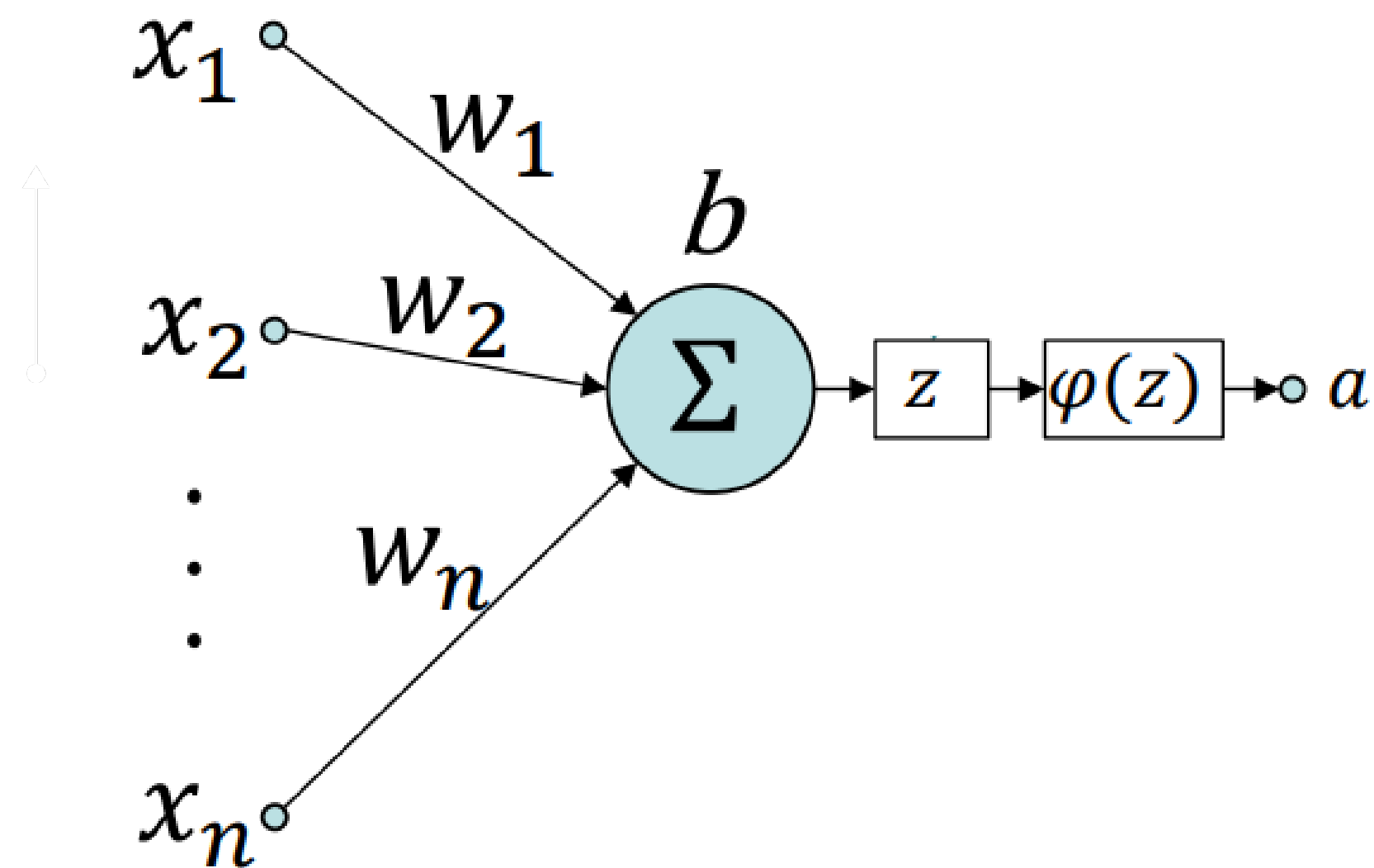
b – смещение: $b \in \mathbb{R}$

$\varphi: \mathbb{R} \rightarrow \mathbb{R}$ – функция активации

z – активационный потенциал:

$$z = b + \sum_{i=1}^n w_i x_i$$

Нейрон – линейная модель



Линейная регрессия:

$$a(x) = b + \sum_{i=1}^n w_i x_i$$

Линейная классификация:

$$a(x) = th(b + \sum_{i=1}^n w_i x_i)$$

Логистическая регрессия:

$$a(x) = \sigma(b + \sum_{i=1}^n w_i x_i)$$

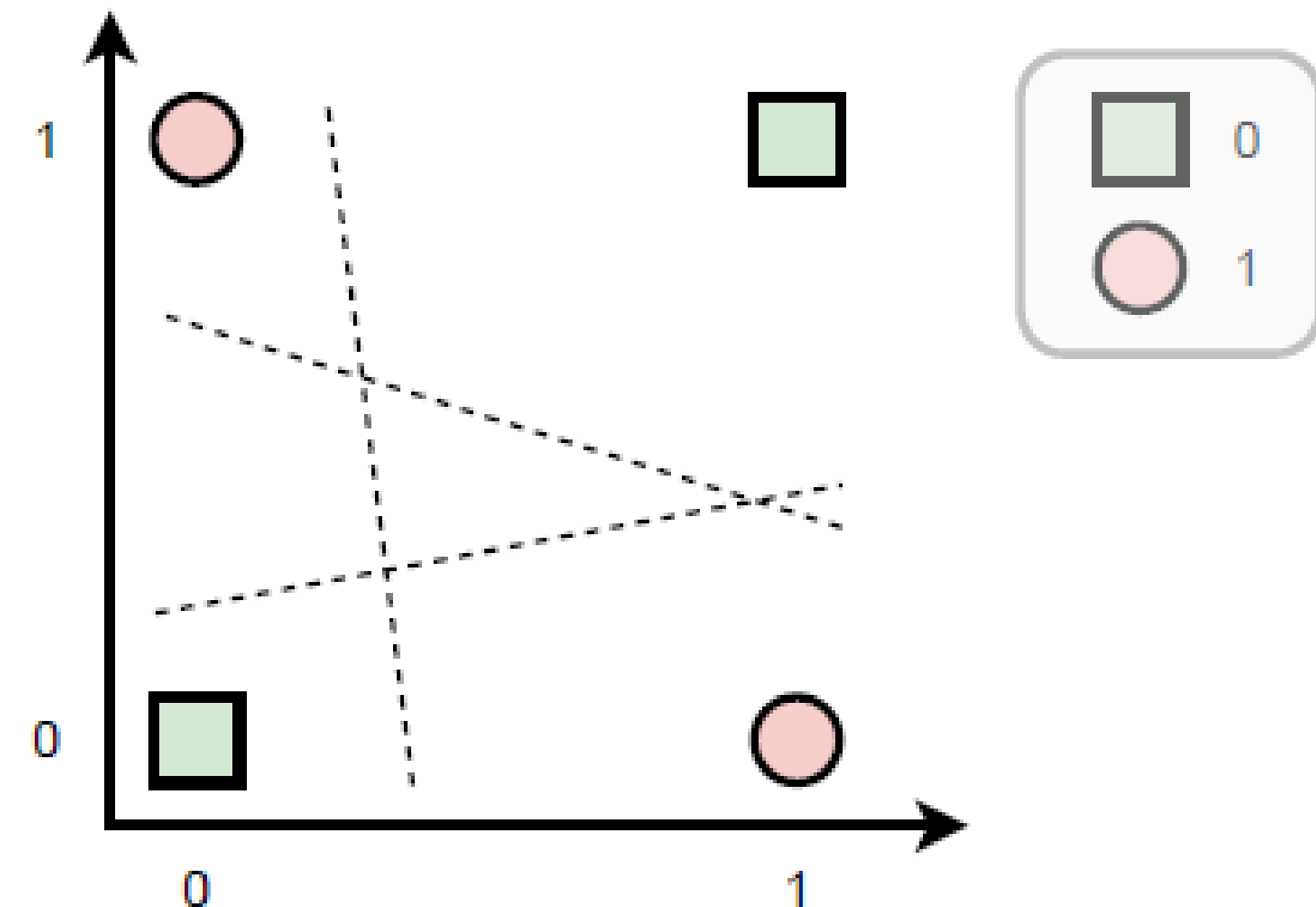
Ограничения линейных моделей

$$f([0,0]) = 0, f([0,1]) = 1, f([1,0]) = 1, f([1,1]) = 0$$

Линейная модель не способна моделировать нелинейную зависимость между признаками!

1 способ: каждая разделяющая линия может быть представлена отдельным нейроном

2 способ: добавить входной нелинейный признак



Простая сеть

Стандартные блоки нейронной сети состоят из:

1: Линейное преобразование:

Математическая операция (вроде взвешенной суммы), где веса и смещения являются параметрами:

$$W, b: x \rightarrow xW + b \quad (W \in \mathbb{R}^{d \times k}, x \in \mathbb{R}^d, b \in \mathbb{R}^k)$$

Обучаемые параметры: матрица W и вектор b

2: Фиксированная нелинейность

Простая, заранее заданная нелинейная функция (например, ReLU, сигмоида, и тд).

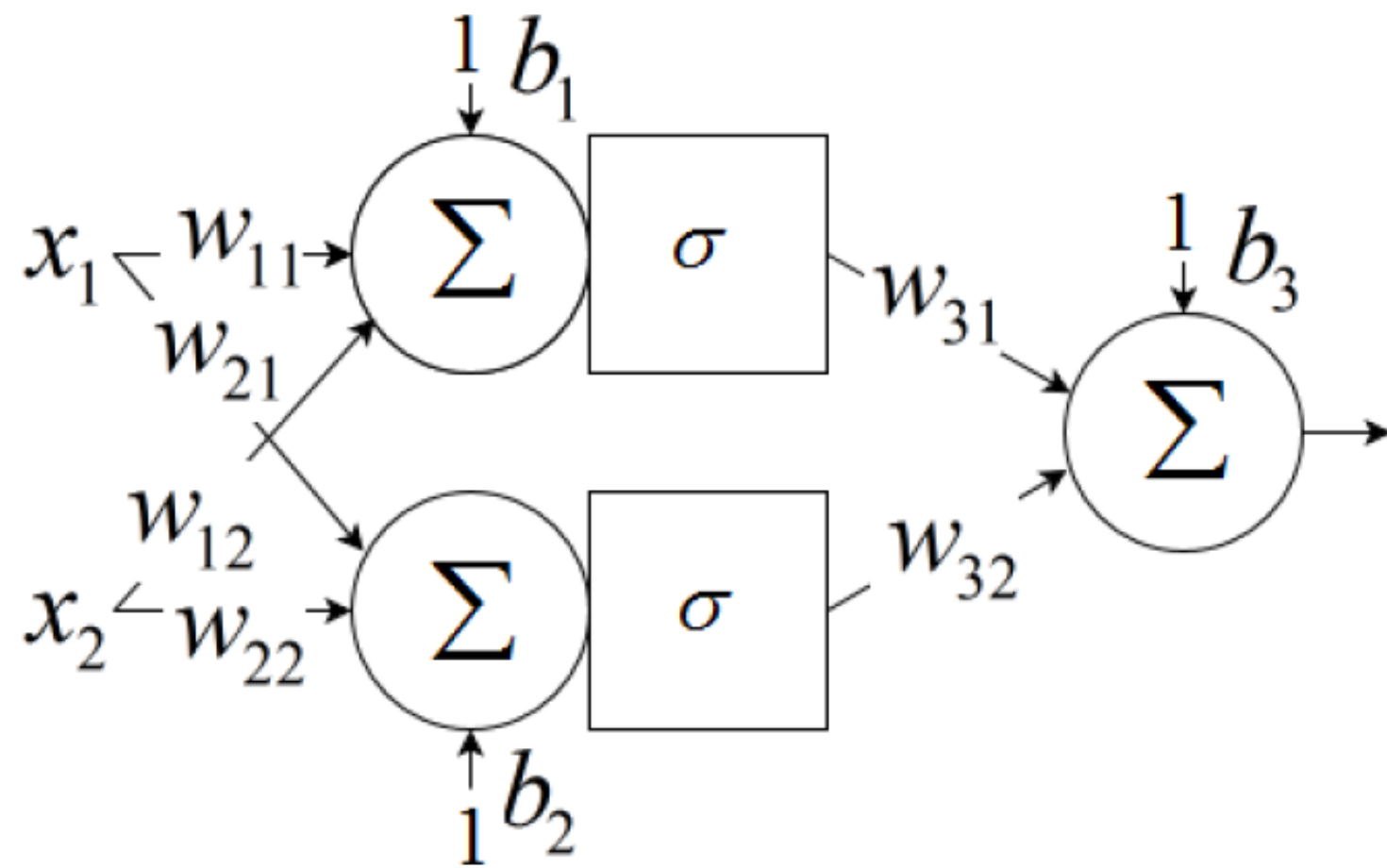
$$\text{ReLU}(x) = \max(0, x)$$

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



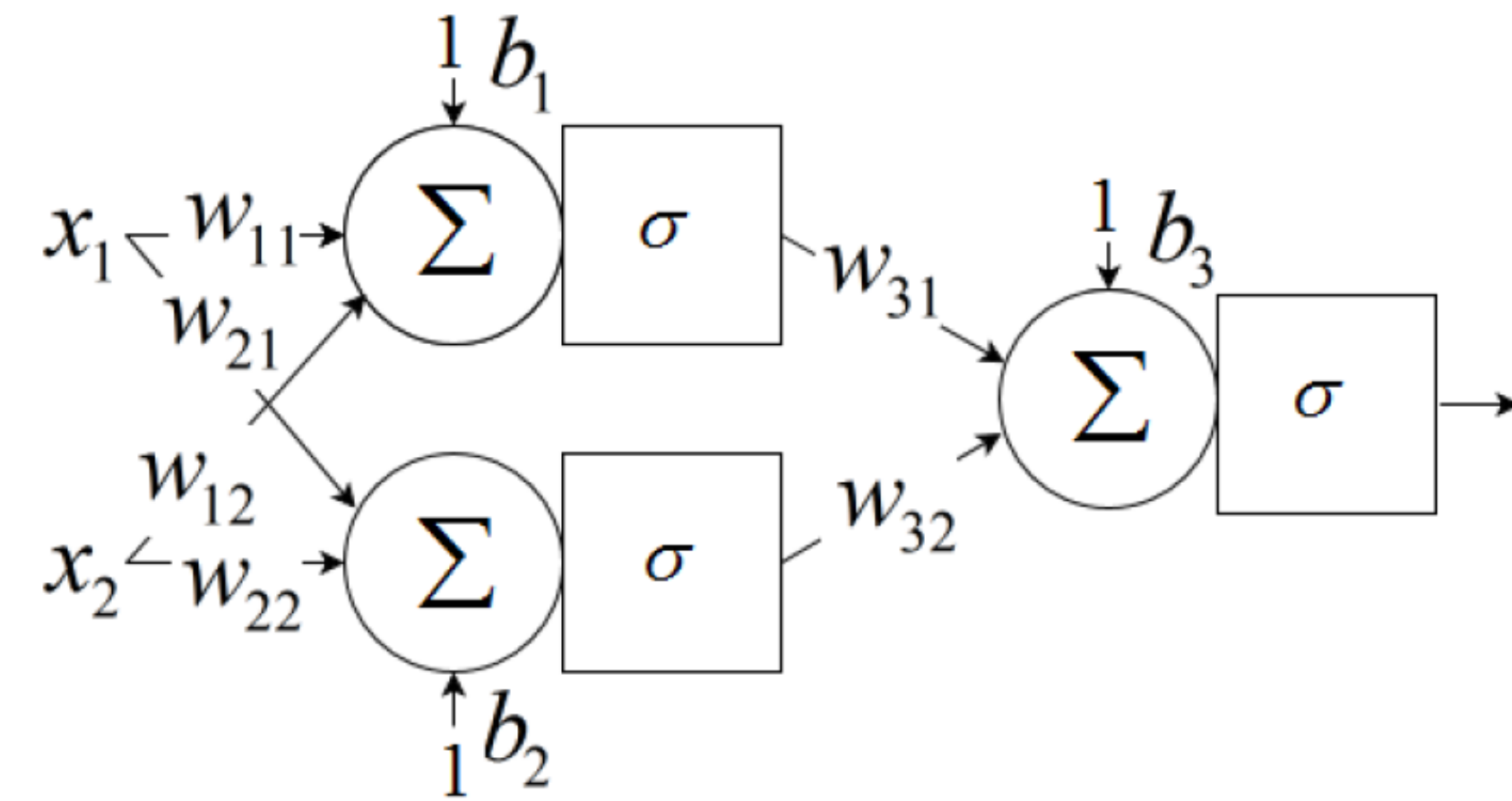
Двухслойная сеть

Регрессия:



$$a = b_3 + w_{31}\sigma(b_1 + w_{11}x_1 + w_{12}x_2) + w_{32}\sigma(b_2 + w_{21}x_1 + w_{22}x_2)$$

Классификация:



$$a = \sigma(b_3 + w_{31}\sigma(b_1 + w_{11}x_1 + w_{12}x_2) + w_{32}\sigma(b_2 + w_{21}x_1 + w_{22}x_2))$$

Матричное представление

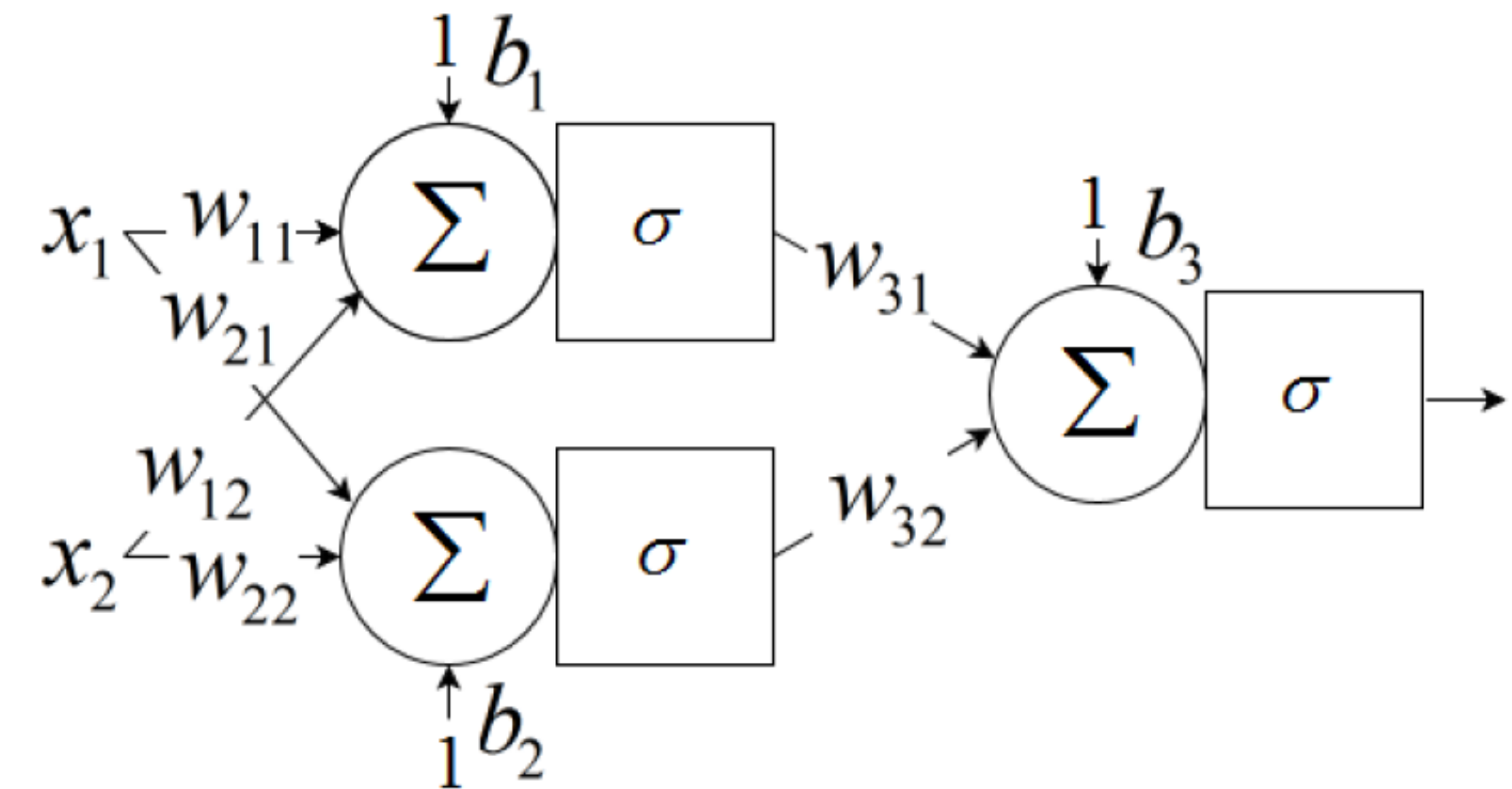
В матричном виде каждый слой – умножение на матрицу W , сдвиг вектором b и применение функции активации σ

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad W = \begin{bmatrix} w_1 & w_2 \end{bmatrix}, \quad b$$

$$z = Wx + b$$

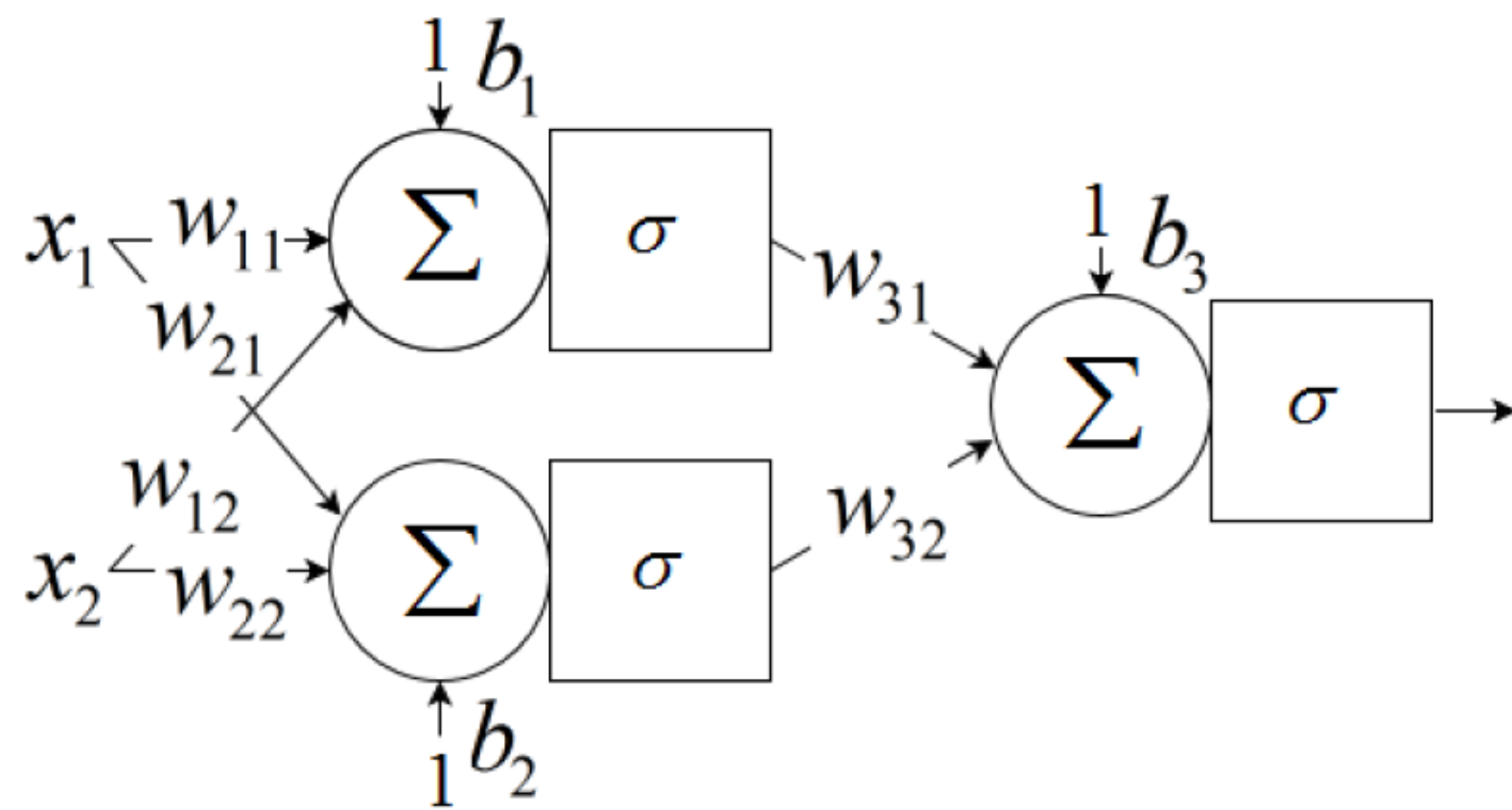
$$\tilde{x} = \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}, \quad \tilde{W} = \begin{bmatrix} w_1 & w_2 & b \end{bmatrix}$$

$$z = \tilde{W} \tilde{x}$$



$$a = \sigma(b_3 + w_{31}\sigma(b_1 + w_{11}x_1 + w_{12}x_2) + w_{32}\sigma(b_2 + w_{21}x_1 + w_{22}x_2))$$

Матричное представление



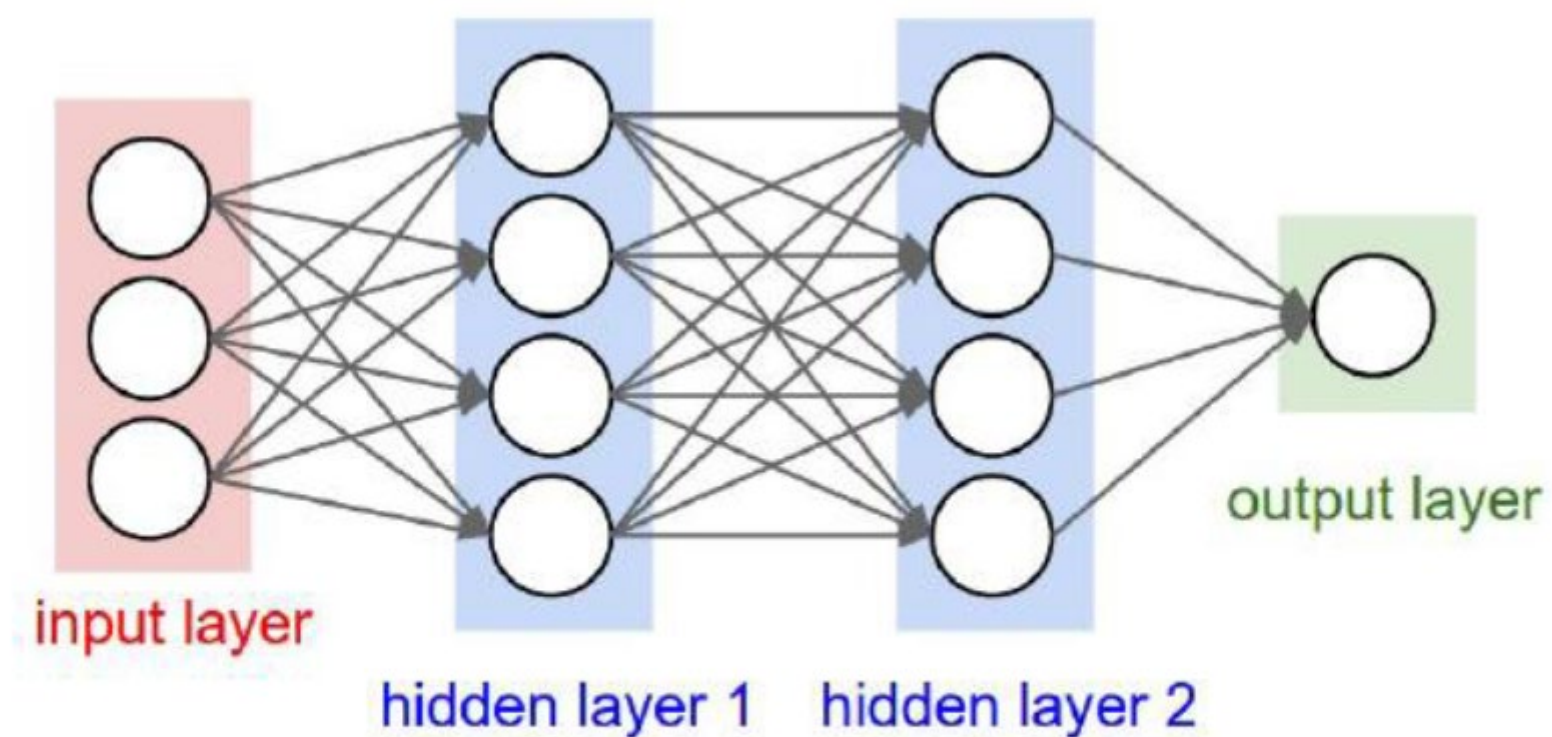
$$a = \sigma(b_3 + w_{31}\sigma(b_1 + w_{11}x_1 + w_{12}x_2) + w_{32}\sigma(b_2 + w_{21}x_1 + w_{22}x_2))$$

$$= \sigma \left(\begin{bmatrix} w_{31} & w_{32} & b_3 \end{bmatrix} \sigma \left(\begin{bmatrix} w_{11} & w_{12} & b_1 \\ w_{21} & w_{22} & b_2 \\ 1 & & \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ 1 \end{pmatrix} \right) \right)$$

Композиция функций

Нейронная сеть представляет собой последовательное преобразование признакового пространства

По сути, это просто некоторая *композиция функций*:



k – количество слоёв,

W_k – матрица весов слоя,

φ_k – нелинейная функция активации

$$\varphi_k(W_k \cdot \dots \cdot \varphi_2(W_2 \cdot \varphi_1(W_1 \cdot x)))$$

Композиция функций

Составная функция – функция, которая состоит из элементарных функций (для которых мы умеем считать производную).

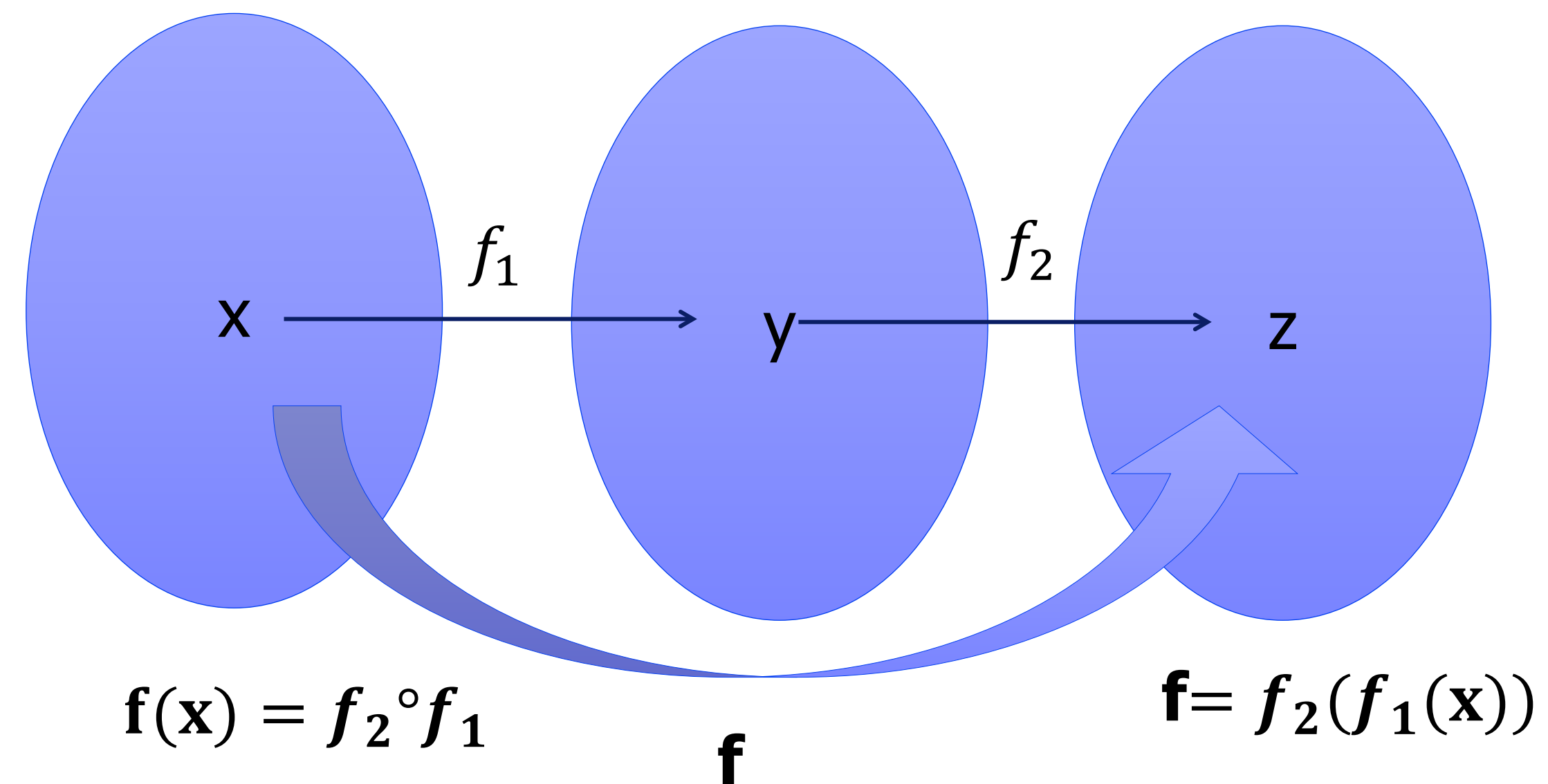
Сложная функция – функция, значения которой отличны от скаляра (матрица, вектор и т.д.).

Композиция функций - одна функция применяется к результату другой:

$$f(x) = (f_2 \circ f_1)(x)$$

$$f_2 \circ f_1 \text{ - композиция двух функций} \quad (f_2 \circ f_1)(x) = f_2(f_1(x))$$

$$f(x) = (f_l \circ f_{l-1} \circ \dots \circ f_2 \circ f_1)(x)$$



Сеть глубокого распространения

Глубокая НС – последовательное преобразование признакового пространства:


$$f_k(W_k \cdot \dots \cdot f_2(W_2 \cdot f_1(W_1 \cdot x)))$$

Вычисление составной функции можно представить в виде направленного ациклического графа:

- Много слоев
- Много данных
- Вычислительные мощности

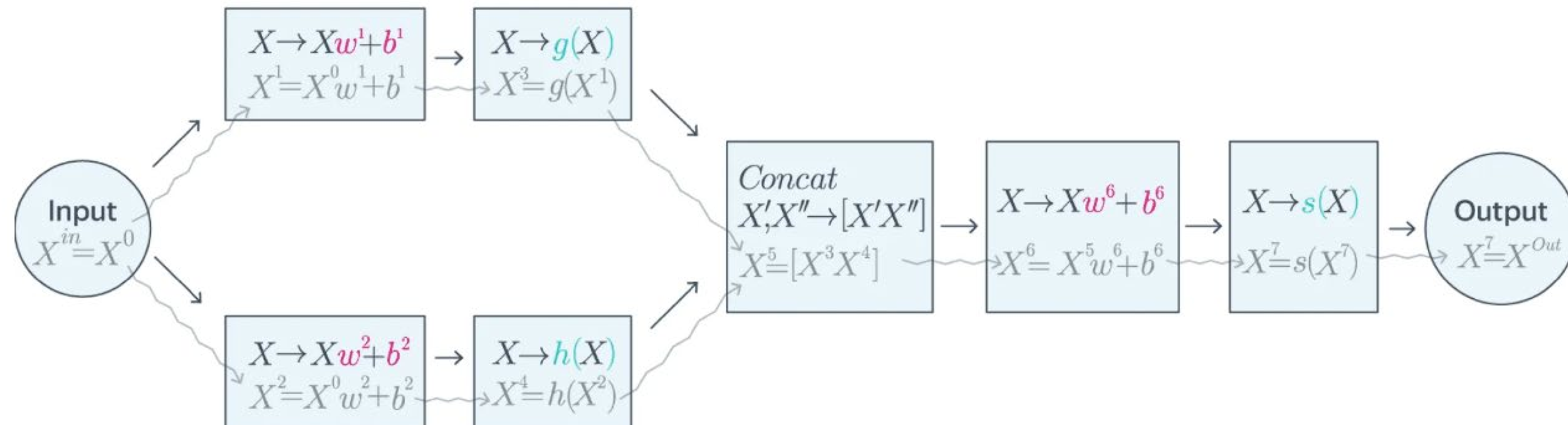
Граф вычислений

Нейронная сеть представляет собой некоторую *композицию функций*, с которой может быть ассоциирован ациклический граф вычислений.

Вершины графа – переменные (входные, внутренние, выходные)

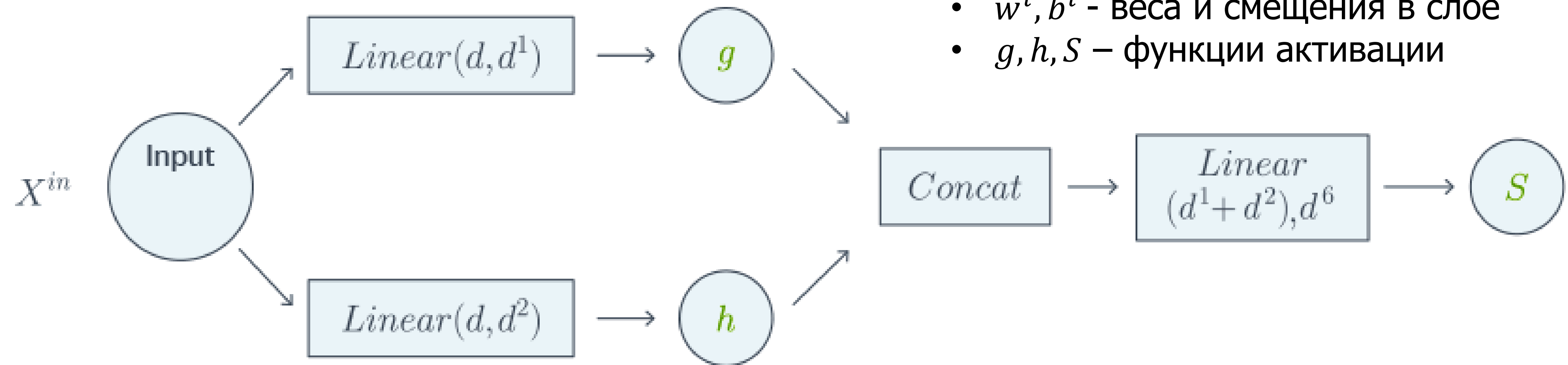
Рёбра – зависимости

Слой – операция (линейная комбинация, нелинейная операции и т.д.)



<https://education.yandex.ru/handbook/ml/article/pervoe-znakomstvo-s-polnosvyaznymi-nejrosetyami>

Вычислительный граф



Обозначения в графе:

- $N \times D$ - размер X входной матрицы признаков
- d^1, d^2, d^6 - линейные преобразования
- w^i, b^i - веса и смещения в слое
- g, h, S - функции активации

Обучение

- **Обучаемая функция:** $f(x_1, x_2, \dots, x_n, a_1, a_2, \dots, a_m): \mathbb{R}^{n+m} \rightarrow \mathbb{R}^k$, n – число признаков, m – число обучаемых параметров, k – число целевых признаков (ответов);
- **Набор данных:** Обучающая выборка $\{(x_i, y_i)\}$
- **Функция потерь** (ошибки): $E(\widehat{y}_i, y_i)$, где \widehat{y}_i - предсказанный вектор для i – объекта, а y_i - вектор реальных значений. Например, MSE: $\frac{1}{|D| \cdot k} \sum_{i=1}^{|D|} \sum_{j=1}^k (y_{i,j} - \widehat{y}_{i,j})^2$
- **Режим обучения:** Переменные x зафиксированы функцией ошибки, параметры a – изменяются в процессе минимизации
- **Режим предсказания:** Переменные x меняются, параметры a – зафиксированы в процессе обучения

Обучение градиентными методами

Пусть входной вектор $x \in \mathbb{R}^n$, веса $W \in \mathbb{R}^{m \times n}$, смещения $b \in \mathbb{R}^m$, активация σ :

$$\begin{aligned} z &= Wx + b \\ a &= \sigma(z) \end{aligned}$$

Каждый слой повторяет эту операцию. Последний слой даёт выход (логиты/прогноз/оценку).

Цель: минимизировать функцию потерь $\mathcal{L}(y, \hat{y})$, где:

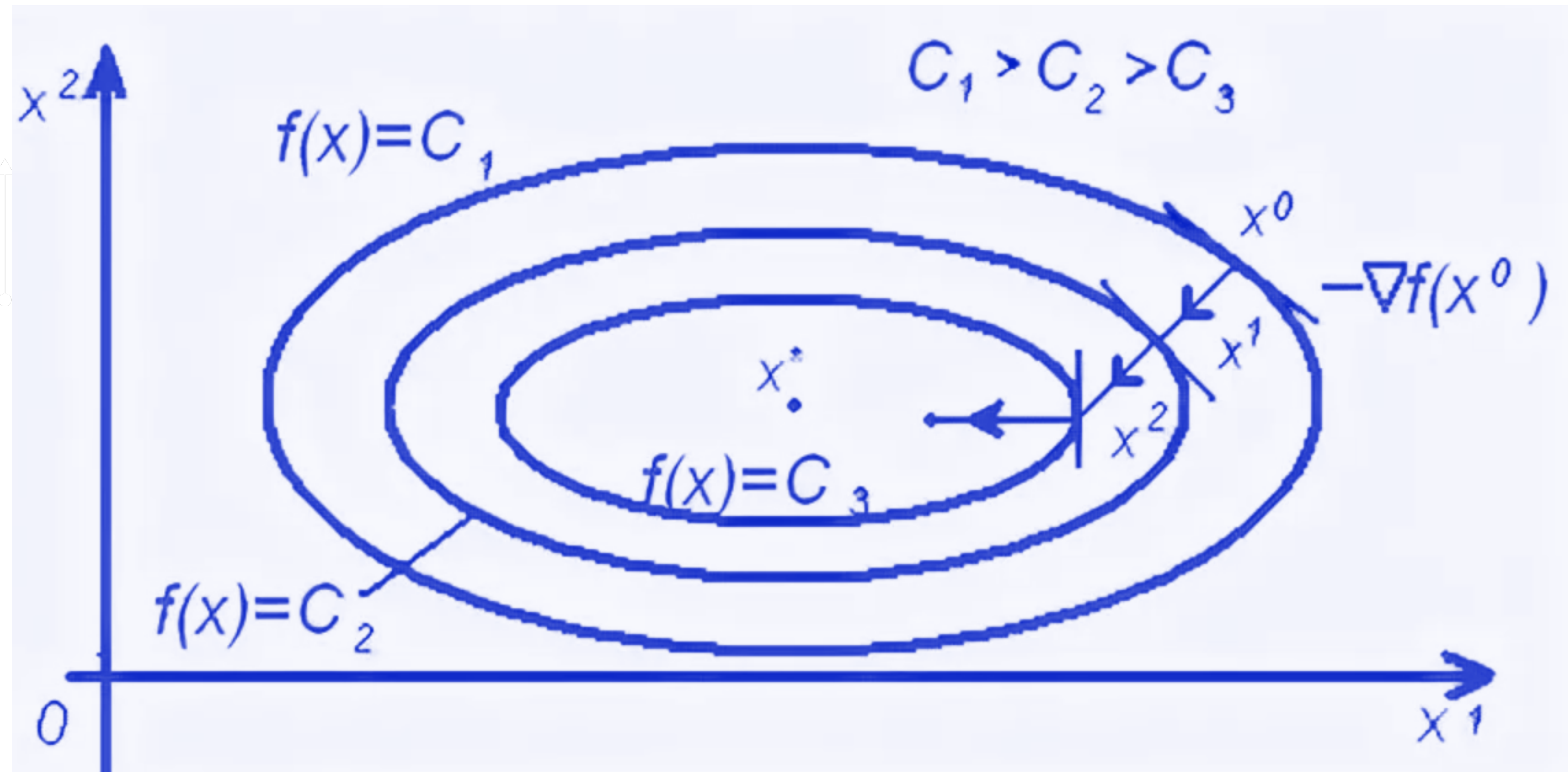
- y — истинная метка,
- \hat{y} — предсказание сети.

Мы обновляем параметры с помощью градиентного спуска:

$$\theta \leftarrow \theta - \eta \cdot \frac{\partial \mathcal{L}}{\partial \theta}$$

Для этого нужно вычислить производную по всем параметрам: $W_1, b_1, W_2, b_2, \dots$

Градиентный спуск



Градиент и Якобиан

- Если функция $f : \mathbb{R}^n \rightarrow \mathbb{R}$ (вектор \rightarrow скаляр), то производная — это градиент ∇f размерности n .
- Если функция $F : \mathbb{R}^n \rightarrow \mathbb{R}^m$ (вектор \rightarrow вектор), то производная — это Якобиан J размерности $m \times n$:

Матрица Якоби:

$$F'(x) = \frac{\partial F(x)}{\partial x} = \begin{pmatrix} \frac{\partial F_1(x)}{\partial x_1} & \dots & \frac{\partial F_1(x)}{\partial x_n} \\ \dots & \dots & \dots \\ \frac{\partial F_m(x)}{\partial x_1} & \dots & \frac{\partial F_m(x)}{\partial x_n} \end{pmatrix}$$

Дифференцируемые функции

Если функция $f: \mathbb{R}^n \rightarrow \mathbb{R}^m$ в точке $x \in \mathbb{R}^n$ имеет частные производные по всем независимым переменным x_1, x_2, \dots, x_n , то из этих производных можно составить матрицу $\left(\frac{\partial f_i(x)}{\partial x_j} \right)$ типа $m \times n$, где i - номер строки матрицы, а j — номер столбца.

Матрица Якоби:

$$f'(x) = \frac{\partial f(x)}{\partial x} = \begin{pmatrix} \frac{\partial f_1(x)}{\partial x_1} & \dots & \frac{\partial f_1(x)}{\partial x_n} \\ \dots & \dots & \dots \\ \frac{\partial f_m(x)}{\partial x_1} & \dots & \frac{\partial f_m(x)}{\partial x_n} \end{pmatrix}$$

Производная сложной функции

Композицию $(g \circ f)(x) = g(f(x))$ двух функций $f: \mathbb{R}^n \rightarrow \mathbb{R}^m$ и $g: \mathbb{R}^m \rightarrow \mathbb{R}^k$ часто задают в виде $z = g(y), y = f(x)$, вводя дополнительный набор переменных $y \in \mathbb{R}^m$.

Тогда матрица Якоби в координатной форме (**цепное правило**):

$$\frac{\partial z_i}{\partial x_j} = \sum_{s=1}^m \frac{\partial g_i}{\partial y_s} \frac{\partial f_s}{\partial x_j} = \frac{\partial g_i}{\partial y_1} \frac{\partial f_1}{\partial x_j} + \dots + \frac{\partial g_i}{\partial y_m} \frac{\partial f_m}{\partial x_j}$$

Полная производная

В случае если f – сложная функция одного аргумента x :



Допустим, у нас:

$$f(x) = h(g(x))$$

Тогда:

$$\frac{df}{dx} = \frac{dh}{dg} \cdot \frac{dg}{dx}$$

– полная производная сложной функции

Производная сложной функции

1. *Символьное дифференцирование* (например, SymPy):

- громоздкие формулы,
- не масштабируется,

2. *Численное дифференцирование*:

- грубая аппроксимация:

$$\frac{\partial f}{\partial x} \approx \frac{f(x + \varepsilon) - f(x)}{\varepsilon}$$

3. *Автоматическое дифференцирование*:

- + правило вычисления производной сложной функции,
- + вычислительный граф,
- + экономия памяти

Автоматическое дифференцирование

1. *Прямой метод* (forward mode): Вычисляет производные параллельно со значениями функции, двигаясь от входов к выходу, используя правила дифференцирования на каждом элементарном шаге.
2. *Обратный* (reverse mode): Основан на запоминании промежуточных результатов прямого прохода и применении цепного правила в обратном направлении за один проход.


Обратное распространение ошибки (backpropagation)

Если,

$$f(x) = g_m(g_{m-1}(\dots(g_1(x))\dots)), \text{ то } \frac{\partial f}{\partial x} = \frac{\partial g_m}{\partial g_{m-1}} \frac{\partial g_{m-1}}{\partial g_{m-2}} \dots \frac{\partial g_2}{\partial g_1} \frac{\partial g_1}{\partial x}$$

1. Совершить прямой проход (forward mode): вычислить и запомнить все промежуточные значения и производные.
2. Вычислить все градиенты методом автоматического дифференцирования.
3. С помощью полученных градиентов совершить шаг (стохастический градиентный спуск).

Прохождение градиента через слой

- 
- X_i — вход в i -й слой (вектор длины n).
 - Y_i — выход i -го слоя (вектор длины m).
 - A_i — матрица весов слоя ($n \times m$).
 - b_i — вектор смещений (bias) длины m .
 - $f(\cdot)$ — функция активации (применяется покомпонентно).

Прямой проход:

1. Линейное преобразование (умножение входа на матрицу весов):

$$M_i = X_i \cdot A_i$$

2. Смещение (bias):

$$S_i = M_i + b_i$$

3. Активация:

$$Y_i = f(S_i)$$

Прохождение градиента через слой (обратный)

Через активацию:

Выход слоя:

$$Y_i = f(S_i)$$

Тогда:

$$dS_i = f'(S_i) \odot dY_i$$

- dY_i — градиент, пришедший от следующего слоя.
- $f'(S_i)$ — производная активации (поэлементно).
- \odot — поэлементное умножение (произведение Адамара).

- X_i — вход в i -й слой (вектор длины n).
- Y_i — выход i -го слоя (вектор длины m).
- A_i — матрица весов слоя ($n \times m$).
- b_i — вектор смещений (bias) длины m .
- $f(\cdot)$ — функция активации (применяется покомпонентно).

Прохождение градиента через слой (обратный)

Через сложение:

$$S_i = M_i + b_i$$

Отсюда:

$$dM_i = dS_i, \quad db_i = dS_i$$

Прохождение градиента через слой (обратный)

Через линейное преобразование:


$$M_i = X_i \cdot A_i$$

Тогда:

$$dX_i = dM_i \cdot A_i^T$$

$$dA_i = X_i^T \cdot dM_i$$

Прохождение градиента через нейрон


$$y = \sum_i x_i$$

Производная по каждому x_i :

$$\frac{\partial y}{\partial x_i} = 1$$

Производная:

$$dx_i = dy$$

Прохождение градиента через нейрон



$$y = \prod_i x_i$$

Частная производная по одному входу x_i :

$$\frac{\partial y}{\partial x_i} = \prod_{j \neq i} x_j$$

Поэтому градиент для x_i :

$$dx_i = dy \cdot \prod_{j \neq i} x_j$$

Прохождение градиента через нейрон



Тогда:

$$y = f(x)$$

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

Значит:

$$dx = dy \cdot f'(x)$$

Пример: если $y = \sin(x)$, то $f'(x) = \cos(x)$, и:

$$dx = dy \cdot \cos(x)$$

Пример

$$F = \frac{\sin(x + y)}{x \cdot y}$$



Пример

$$F = \frac{\sin(x + y)}{x \cdot y}$$



Пример



Пример



Пример





**Спасибо
за внимание!**

