

Введение в глубокое обучение

План лекции

- Ограничения линейных моделей
- Модель глубокого обучения
- Вычислительные возможности нейросетей
- Слои моделей
- Функция активации
- Обратное распространение ошибки
- Реализации функции активации
- Эвристики стохастического градиентного спуска
- Нормировка признаков
- Регуляризация (прореживание)

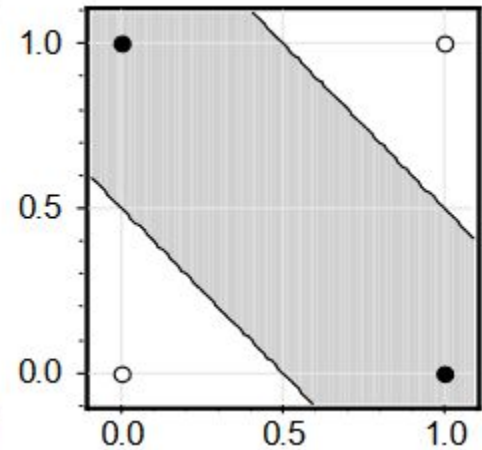
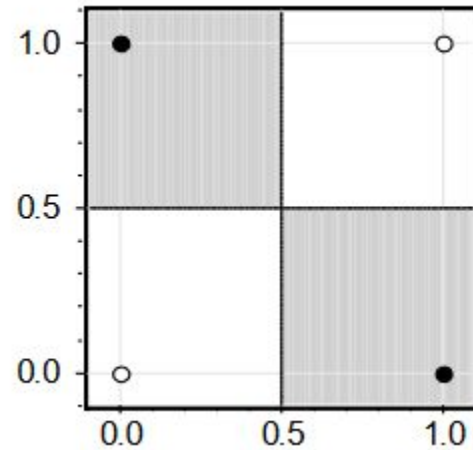
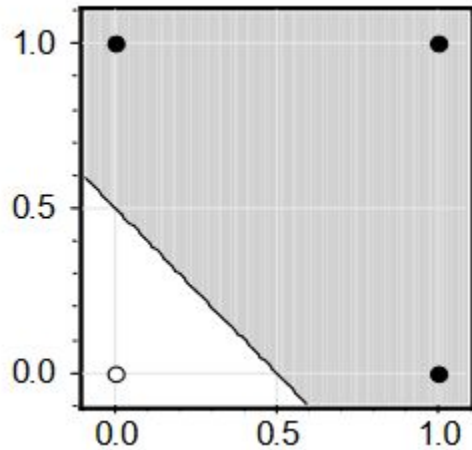
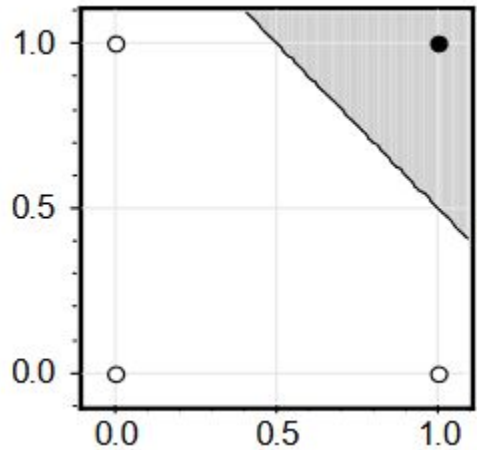
Ограничения линейных моделей

- Работают только с линейными зависимостями
- Сами не конструируют высокоабстрактные признаки
 - текстовые данные
 - графические данные

Задача «исключающего ИЛИ»

$$x^1 \vee x^2 = [x^1 + x^2 - \frac{1}{2} > 0]$$

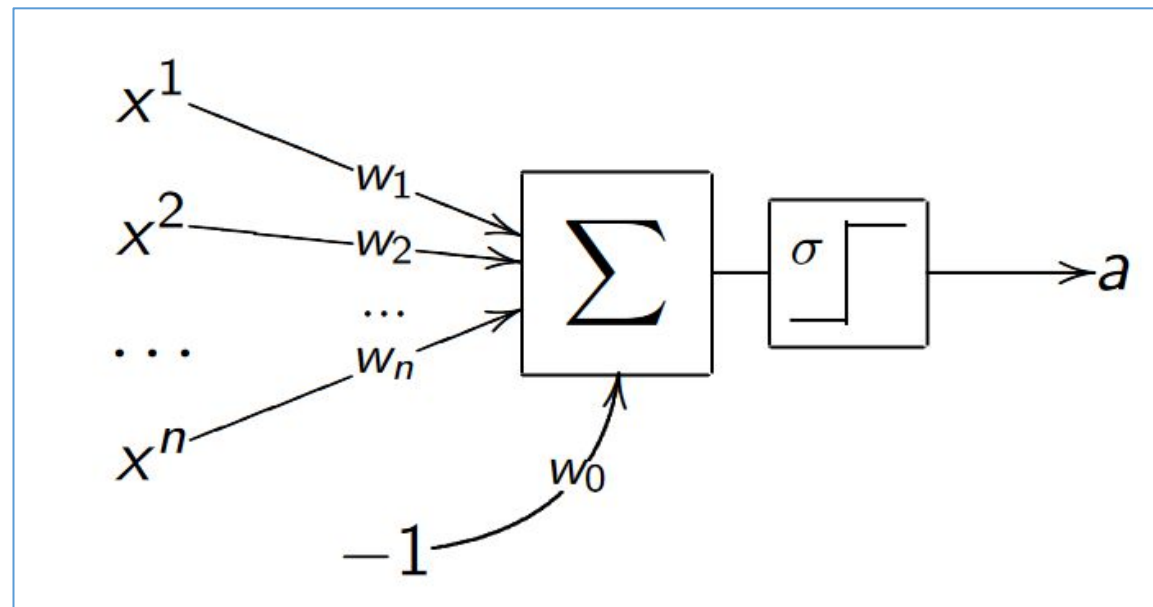
$$x^1 \oplus x^2 = [x^1 + x^2 - 2x^1x^2 - \frac{1}{2} > 0]$$



$$x^1 \wedge x^2 = [x^1 + x^2 - \frac{3}{2} > 0]$$

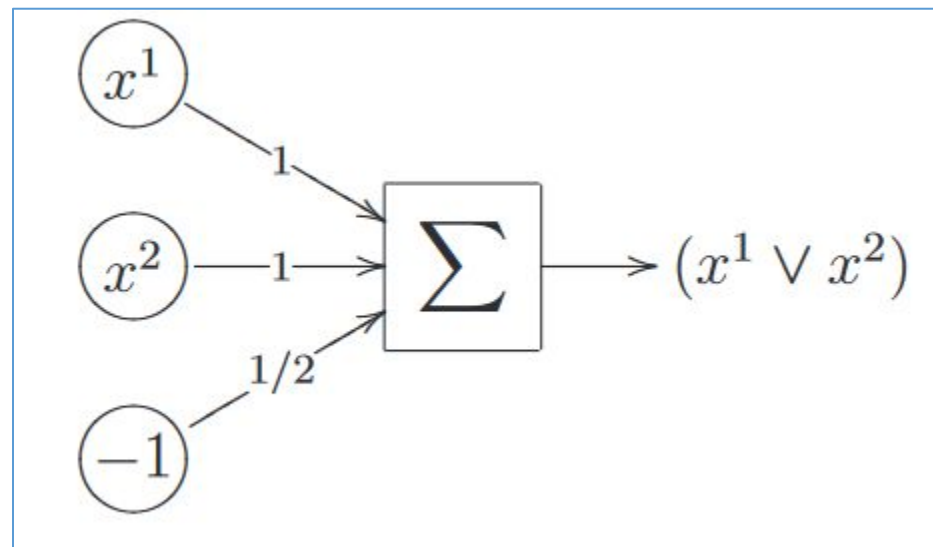
Задача «исключающего ИЛИ»

$$a(x, w) = \sigma(\langle w, x \rangle) = \sigma\left(\sum_{j=1}^n w_j f_j(x) - w_0\right)$$

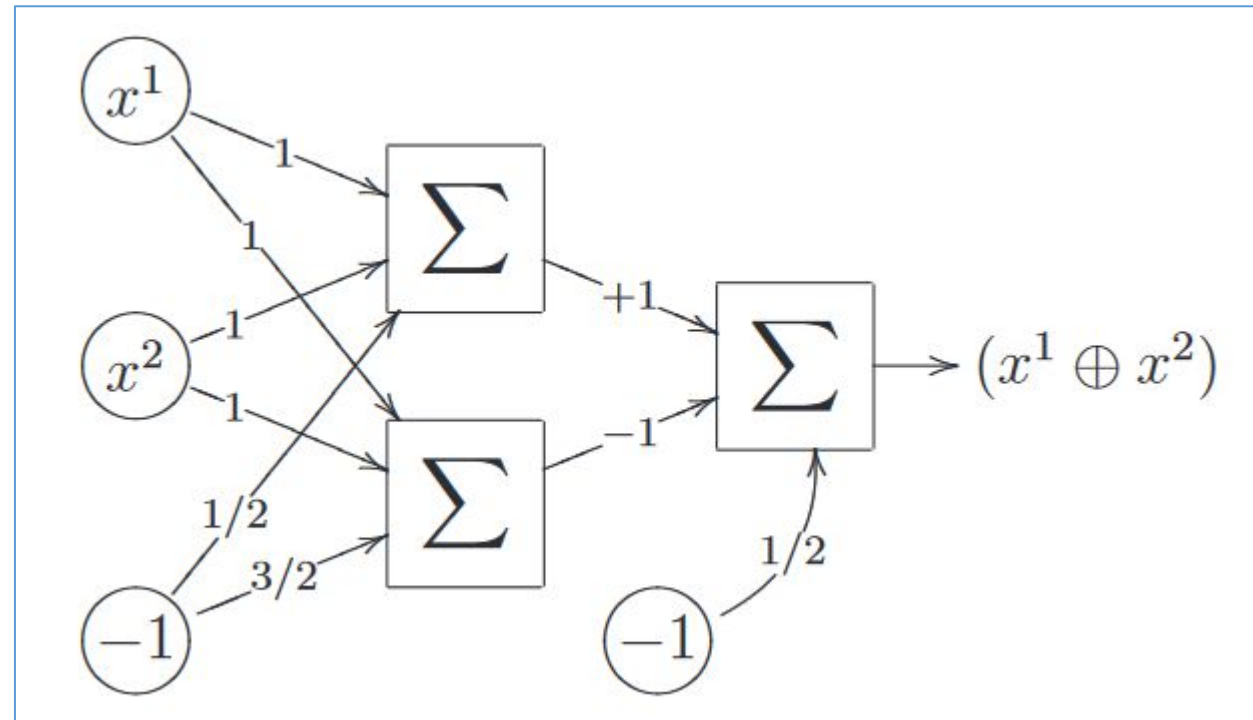
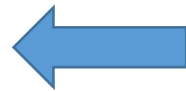
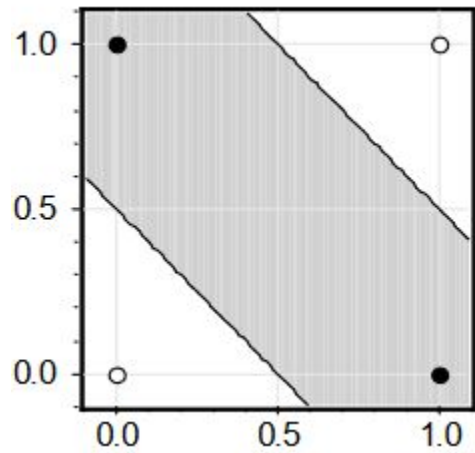


Задача «исключающего ИЛИ»

$$x^1 \vee x^2 = [x^1 + x^2 - \frac{1}{2} > 0]$$



Задача «исключающего ИЛИ»



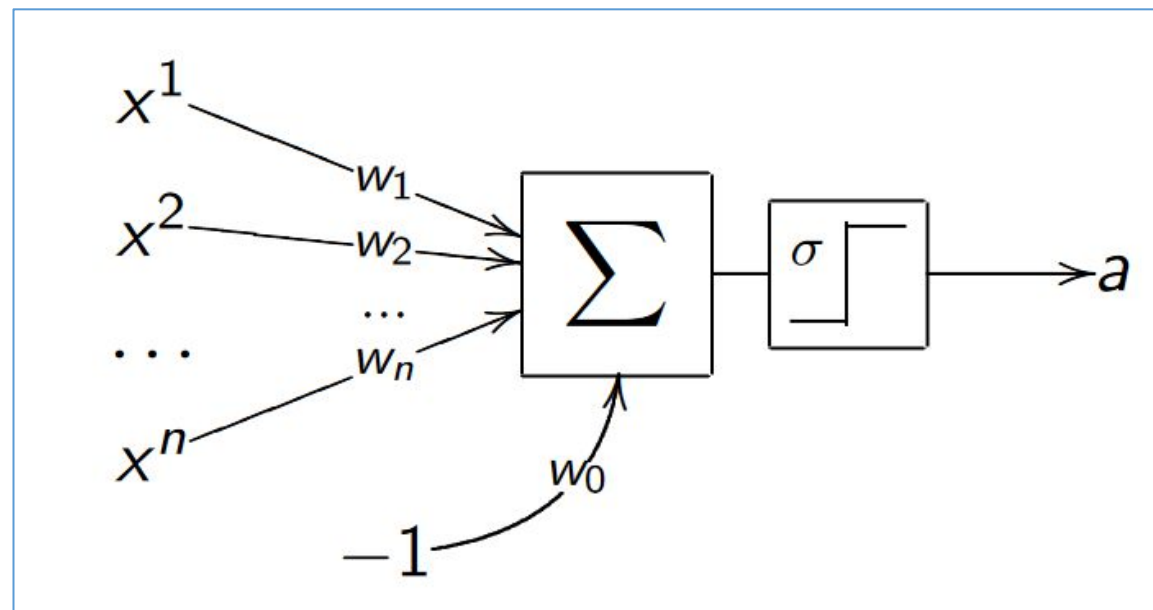
Задача «исключающего ИЛИ»

Таким образом два решения:

- конструирование нового признака на основе исходных (*сложно*)
- построение композиции моделей

Новый подход

$$a(x, w) = \sigma(\langle w, x \rangle) = \sigma\left(\sum_{j=1}^n w_j f_j(x) - w_0\right)$$

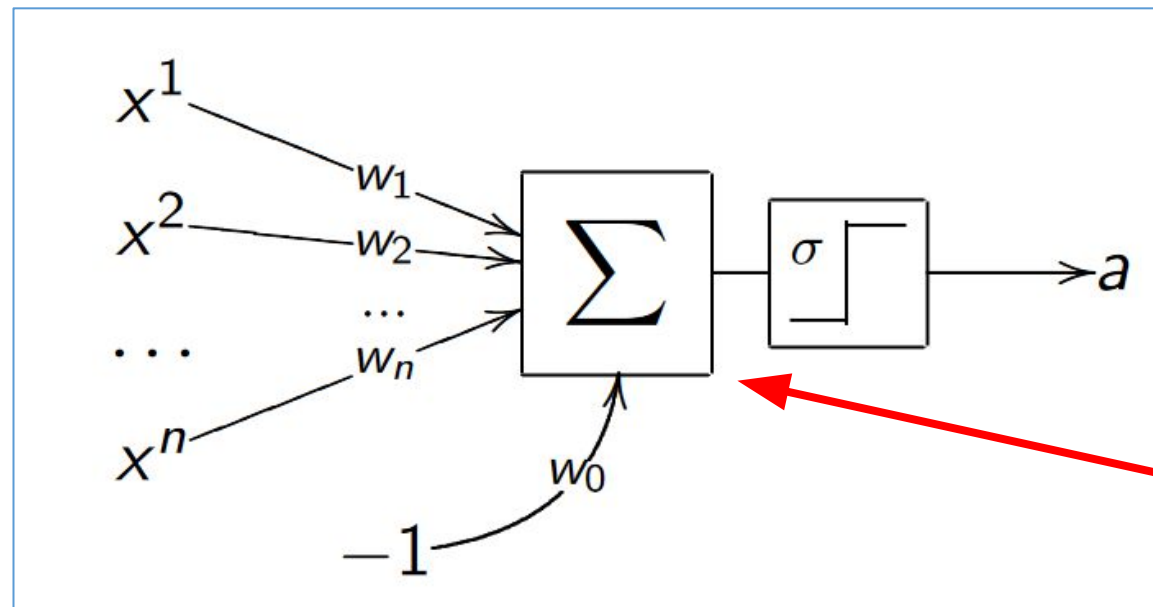


Новый подход

$$a(x, w) = \sigma(\langle w, x \rangle) = \sigma\left(\sum_{j=1}^n w_j f_j(x) - w_0\right)$$



перцептрон
(МакКаллок и Питтс, 1943
год)



искусственный
нейрон

Модель глубокого обучения (нейросеть)

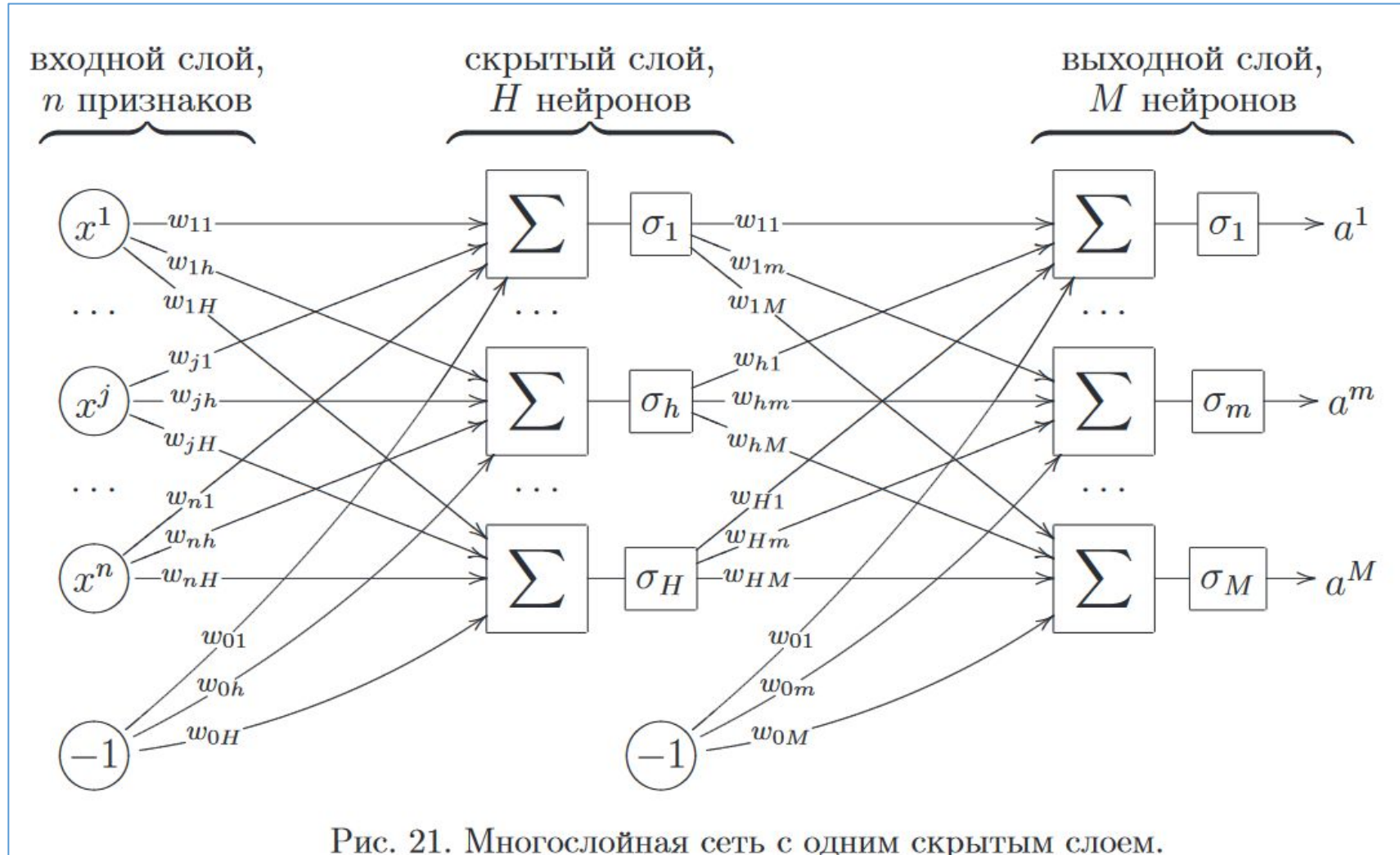
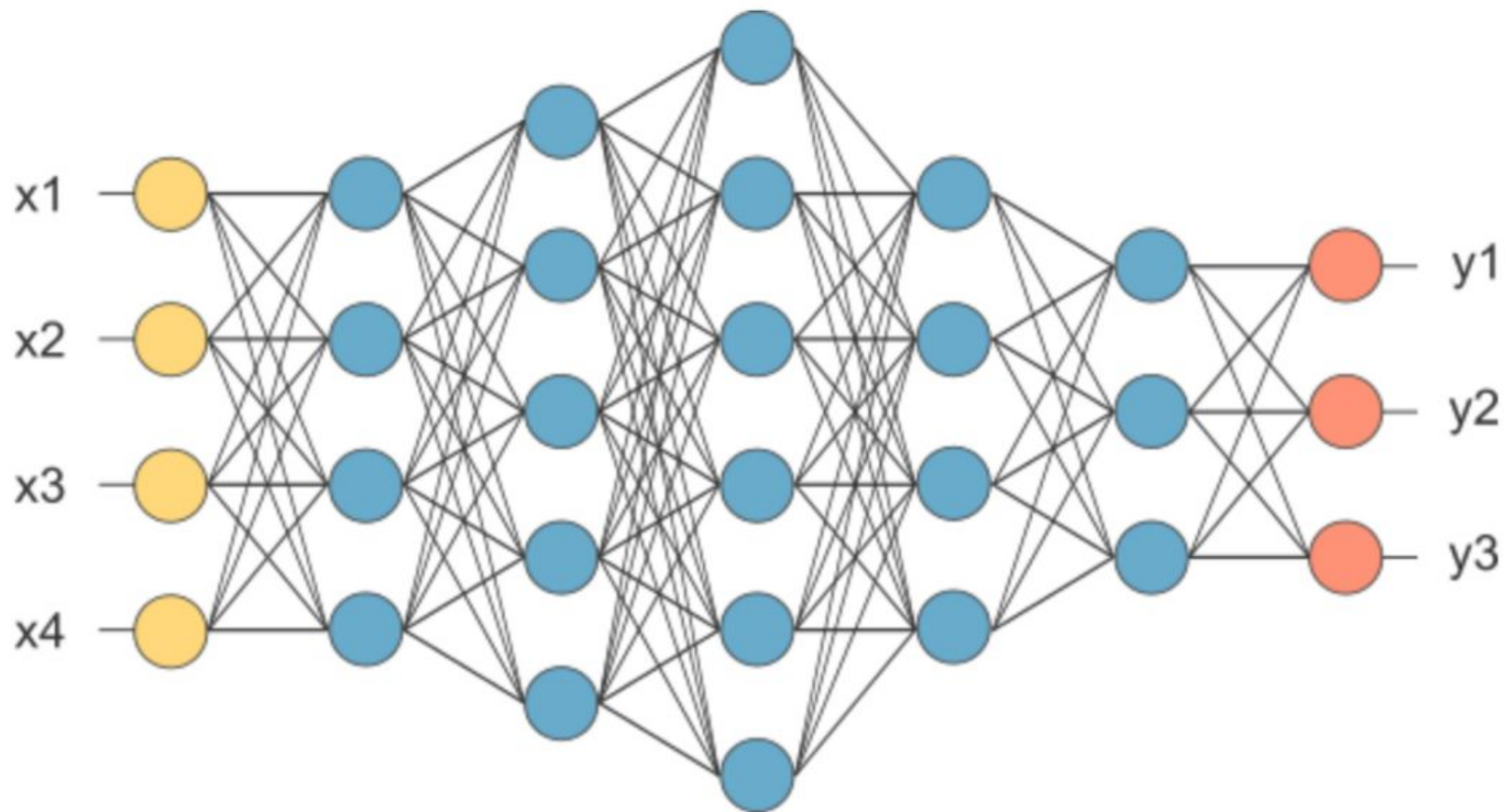


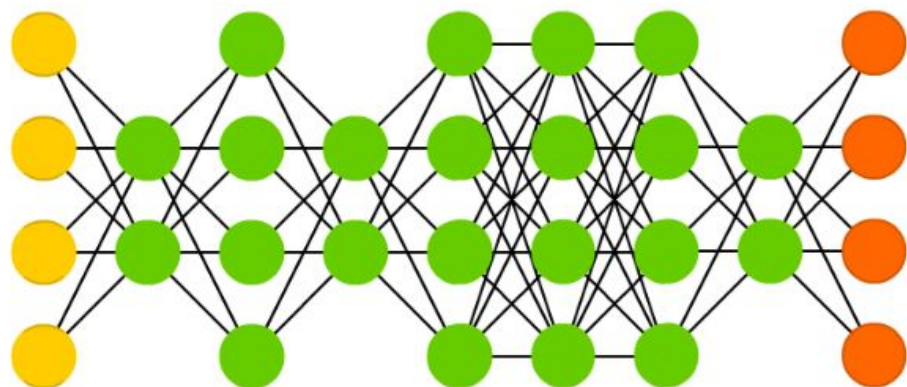
Рис. 21. Многослойная сеть с одним скрытым слоем.

Модель глубокого обучения (нейросеть)

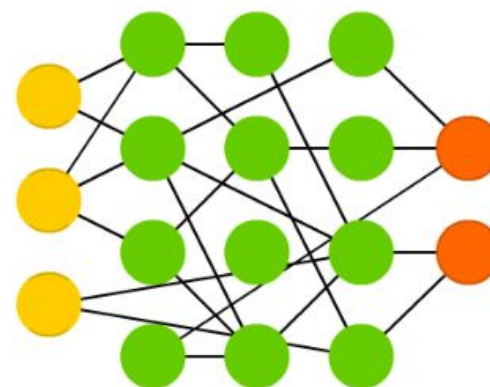





Модель глубокого обучения (нейросеть)

Полносвязная сеть



Неполносвязная сеть



-  Входной слой
-  Скрытый слой
-  Выходной слой

Вычислительные возможности нейросетей

1. Теорема Вейерштрасса – Стоуна
2. Теорема Колмогорова – Арнольда (13 проблема Гильберта)
3. Универсальная теорема аппроксимации (теорема Цыбенко)
Двухслойная сеть может аппроксимировать любую непрерывную функцию многих переменных с любой точностью при достаточном количестве скрытых нейронов
4. Обобщенная аппроксимационная теорема (теорема Горбаня)
С помощью линейных операций и одной нелинейной функции активации можно приблизить любую непрерывную функцию с любой желаемой точностью

Вычислительные возможности нейросетей

Несколько замечаний:

- двух слоёв достаточно для аппроксимации практически всех «математических» функций
- нейросети обучаются преобразованию признаков
- глубина сети позволяет распознавать и конструировать высокоабстрактные признаки

Слои модели

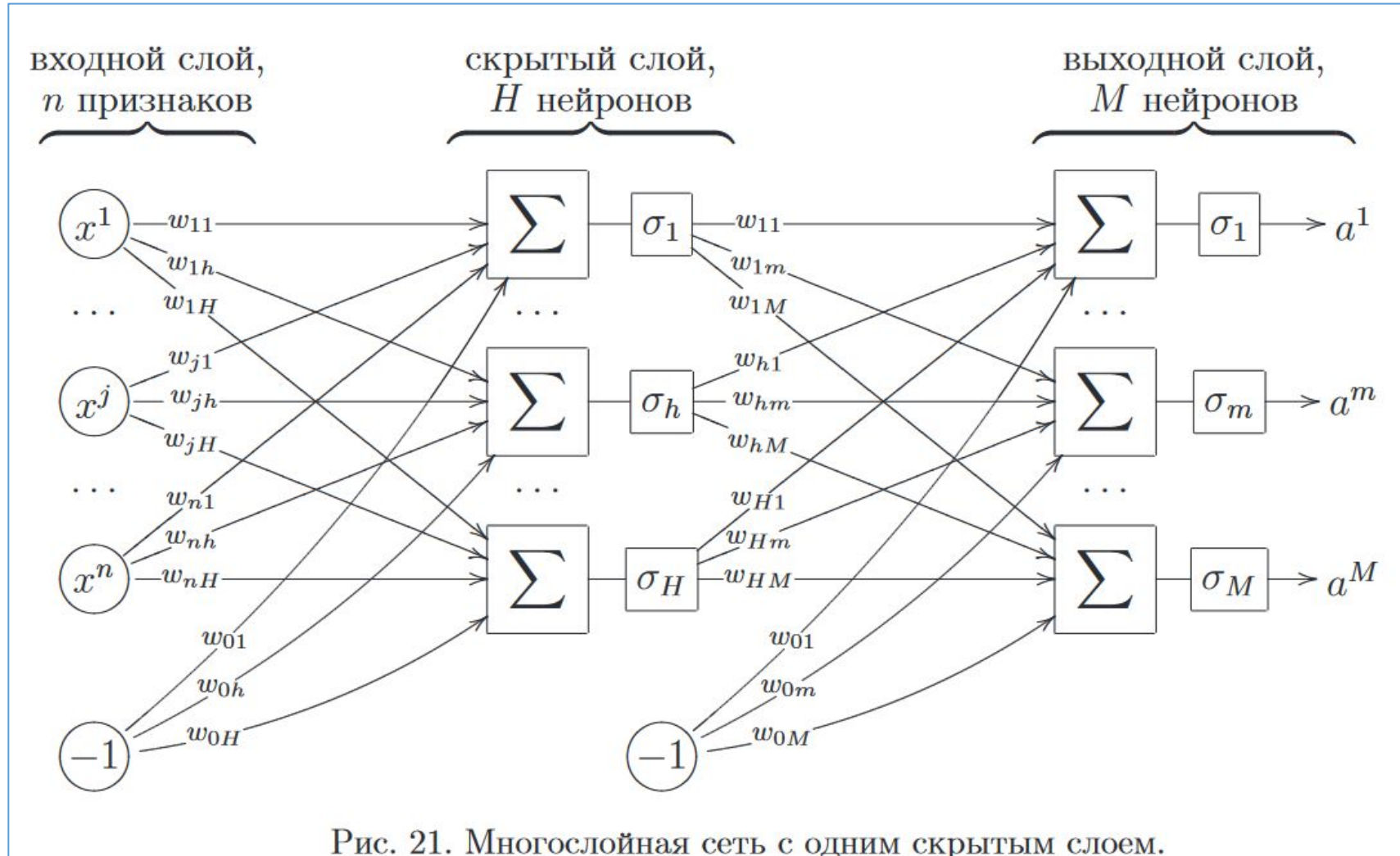
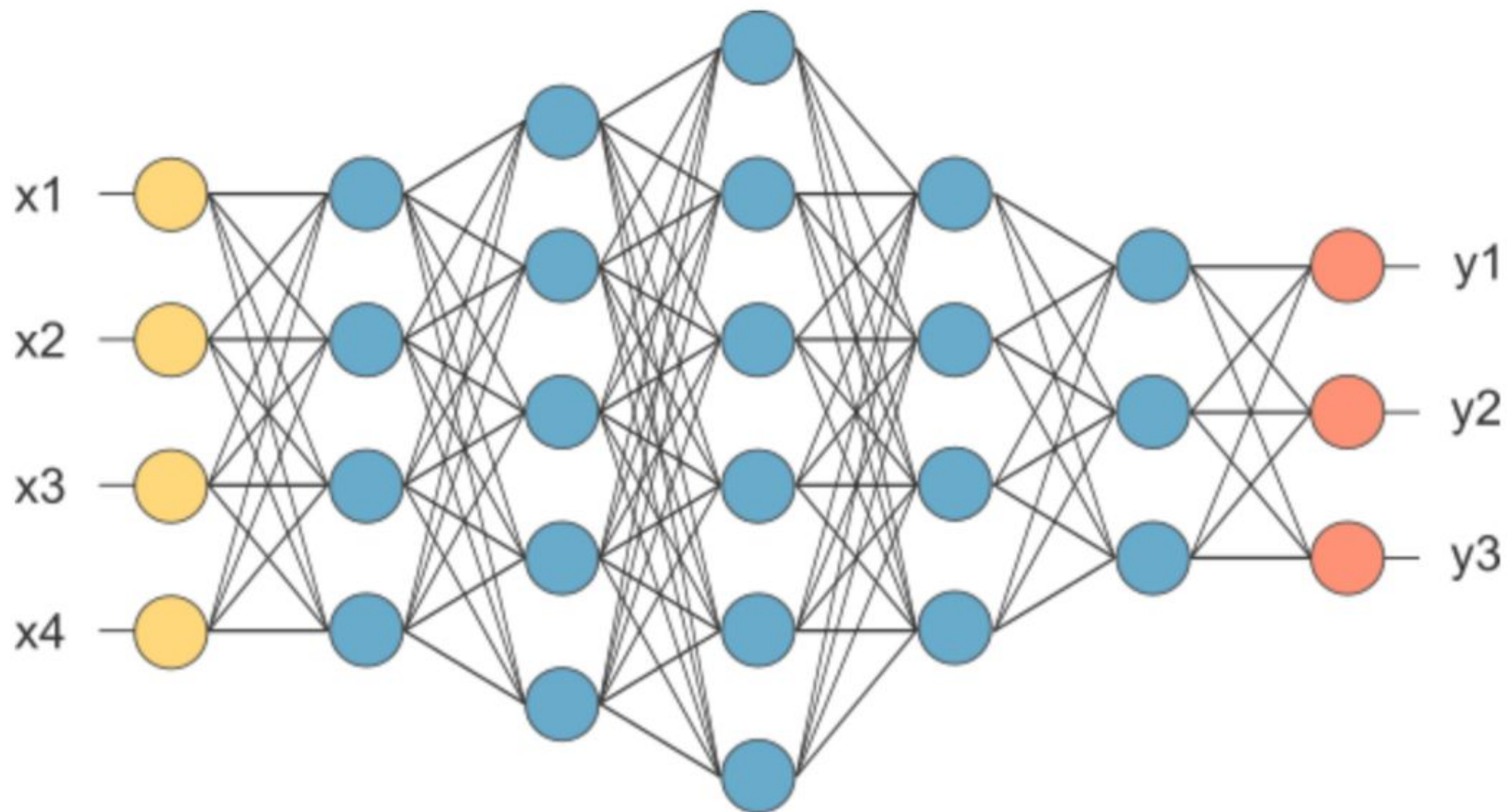


Рис. 21. Многослойная сеть с одним скрытым слоем.

Слой модели

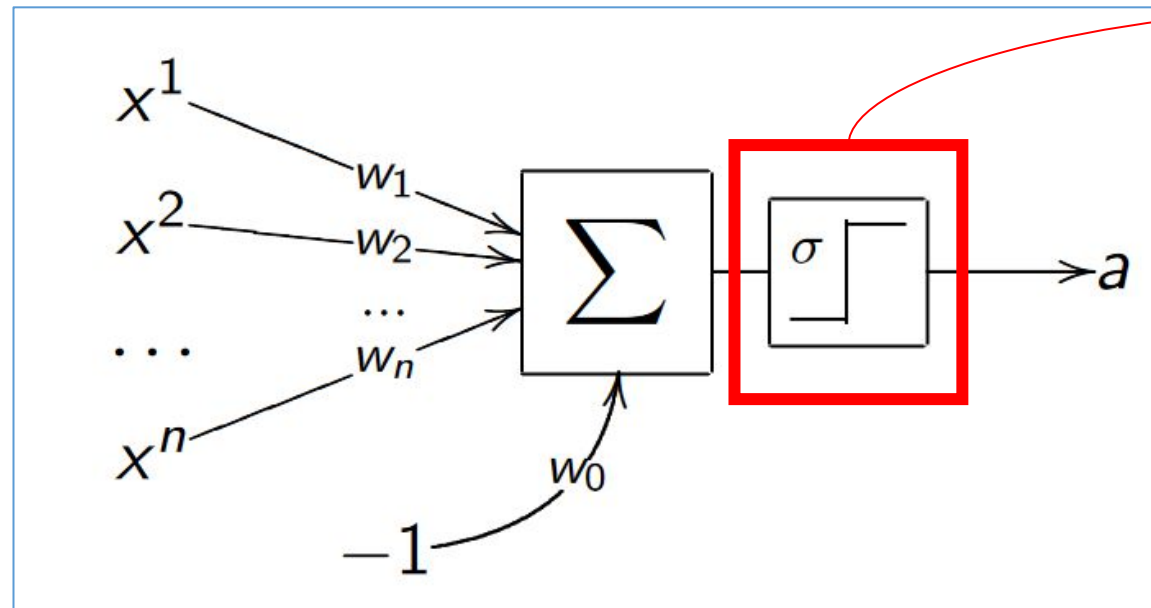


Слои модели

- модель состоит из взаимосвязанных слоёв (*Layers*)
- самый простой и распространённый слой – плотный (*Dense*), но есть и другие ...
- первый слой – *входной*, последний – *выходной*
- *выходной* слой, по сути, это линейная модель
- *скрытые слои* – все слои кроме последнего
- слои хранят параметры (веса) модели

Функция активации

$$a(x, w) = \sigma(\langle w, x \rangle) = \sigma\left(\sum_{j=1}^n w_j f_j(x) - w_0\right)$$



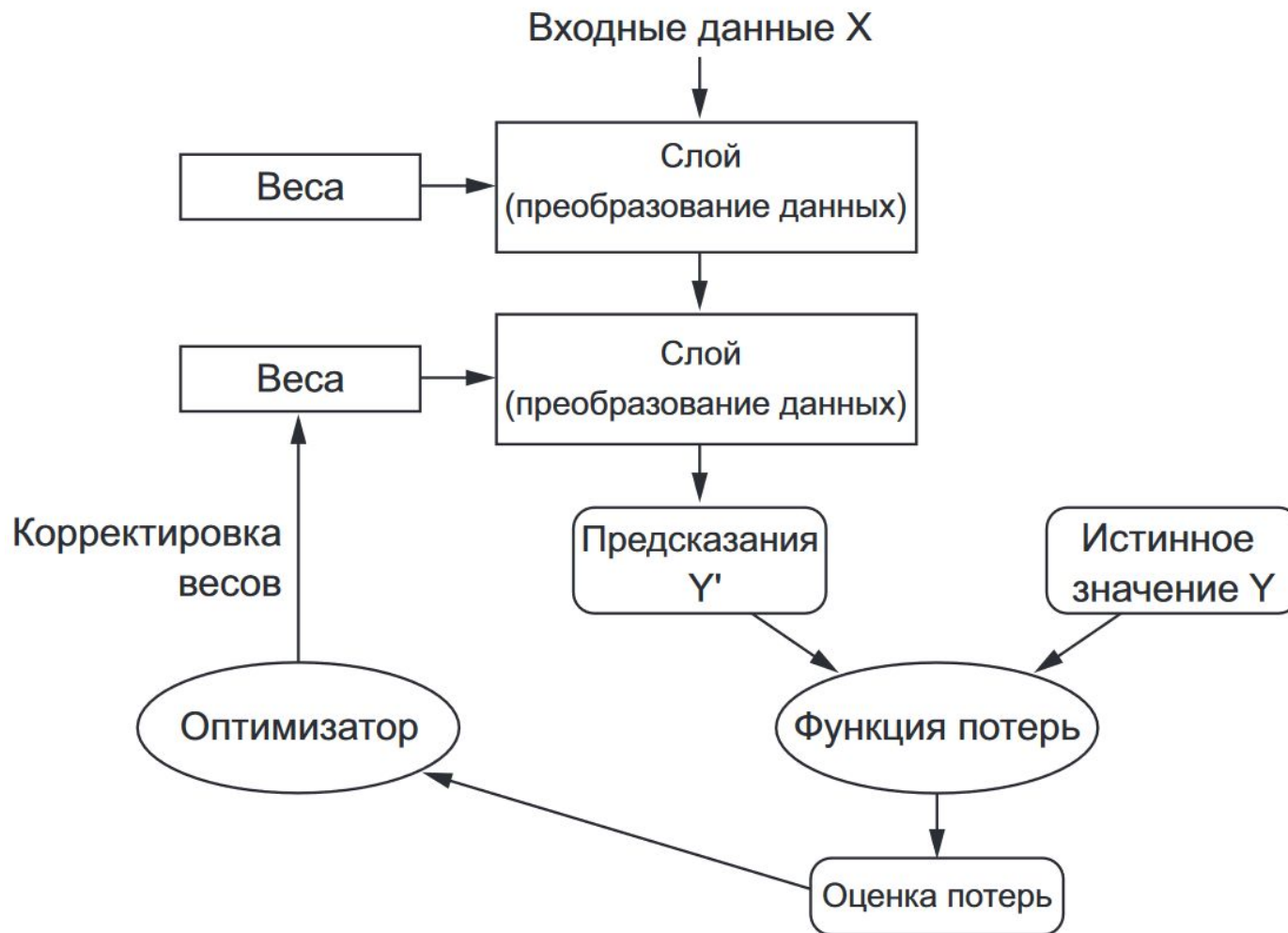
функция активации

Функция активации

Несколько замечаний:

- применяется после линейного преобразования признаков
- *отвечает* за нелинейность
- главное требование – дифференцируемость
- может быть реализована различными функциями

Обучение модели



Обучение модели

Напоминание:

- функция потерь численно определяет, на сколько решена задача
- оптимизатор корректирует параметры модели в сторону ОПТИМАЛЬНЫХ
 - у линейных моделей оптимизатор использовал метод SGD

Обучение модели

Каким методом будем обучать нейросеть?



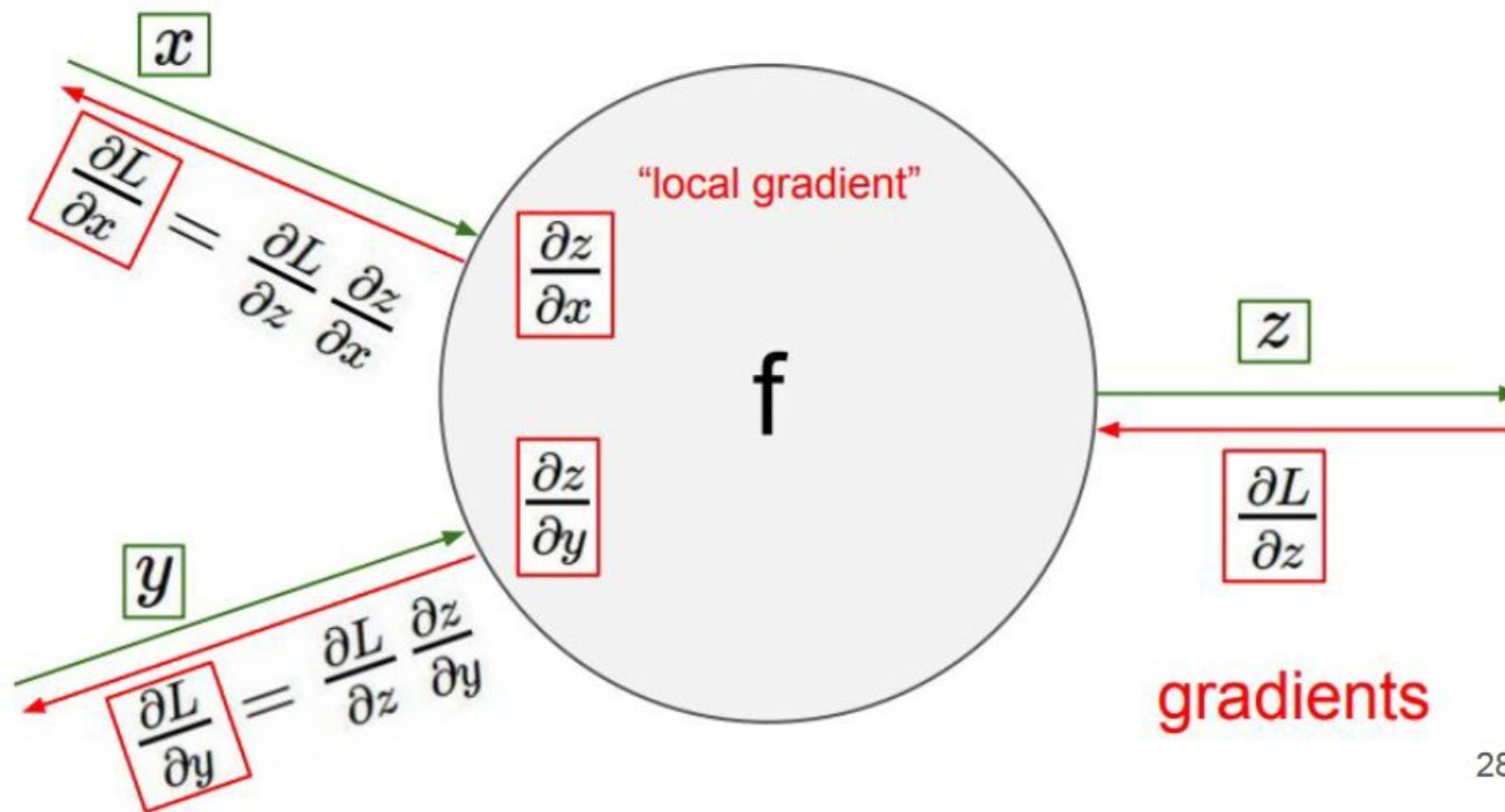
Стохастическим градиентным спуском

Обратное распространение ошибки

Он же:

- backpropagation
- цепное правило
- производная сложной функции

Обратное распространение ошибки



Обратное распространение ошибки

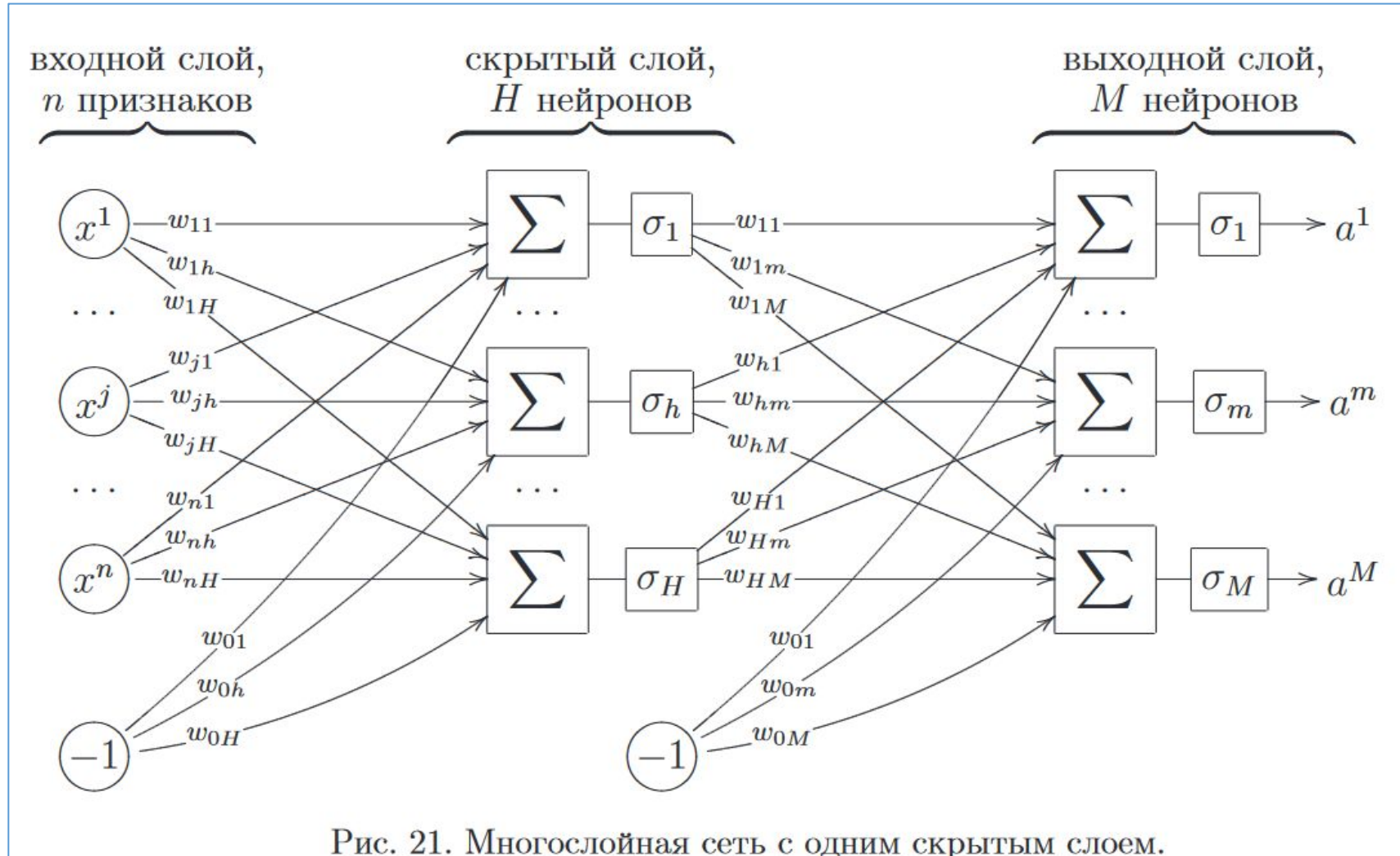


Рис. 21. Многослойная сеть с одним скрытым слоем.

Алгоритм обратного распространения ошибки

Вход: выборка $(x_i, y_i)_{i=1}^{\ell}$, архитектура $(H_l)_{l=1}^L$, параметры η, λ ;

Выход: вектор весов всех слоёв $w = (W^1, \dots, W^L)$;

инициализировать веса w ;

повторять

выбрать объект x_i из X^{ℓ} (например, случайно);

прямой ход: для всех $l = 1..L, h = 1..H_l$

$$\Sigma_{ih}^l := \sum_{k=0}^{H_{l-1}} w_{kh}^l x_{ik}^{l-1}; \quad x_{ih}^l := \sigma_h^l(\Sigma_{ih}^l); \quad z_{ih}^l := (\sigma_h^l)'(\Sigma_{ih}^l);$$

$$\varepsilon_{hi}^L := \frac{\partial \mathcal{L}_i(w)}{\partial x_h^L}, \quad h = 1..H_L;$$

обратный ход: для всех $l = L..2, k = 0..H_{l-1}$

$$\varepsilon_{ik}^{l-1} = \sum_{h=0}^{H_l} \varepsilon_{ih}^l z_{ih}^l w_{kh}^l;$$

градиентный шаг: для всех $l = 1..L, k = 0..H_{l-1}, h = 1..H_l$

$$w_{kh} := w_{kh} - \eta \varepsilon_{ih}^l z_{ih}^l x_{ik}^{l-1};$$

пока значения Q и/или веса w не стабилизируются;

Обратное распространение ошибки

Плюсы метода:

- + вычисляется, практически рекурсивно, что даёт скорость
- + работает с любой шириной, глубиной сети и функциями активации
- + возможность распараллелить

Минусы:

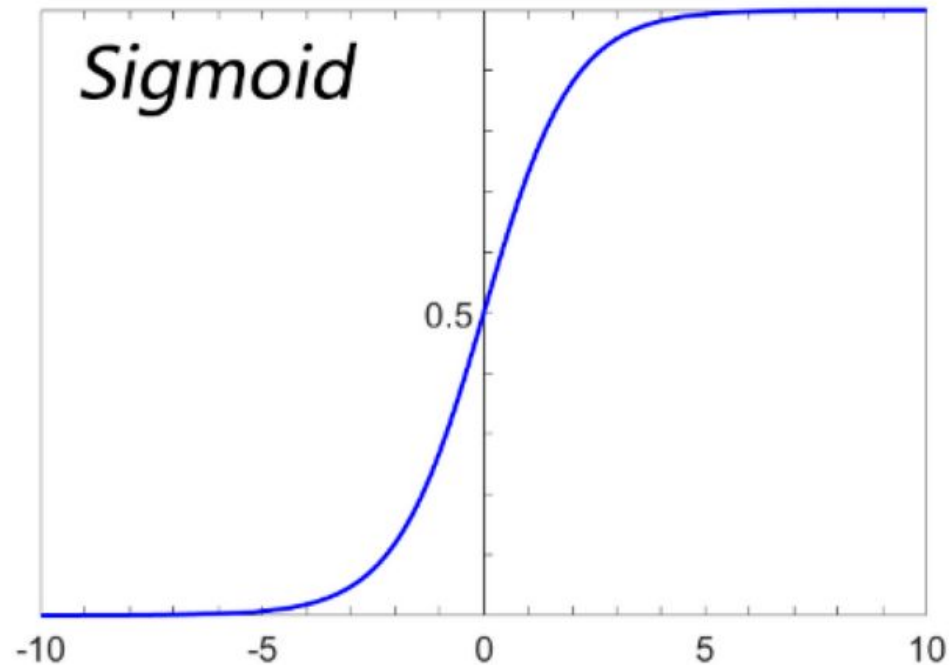
- медленная сходимость
- застревание в локальных экстремумах
- «паралич» сети из-за горизонтальных асимптот сигмоиды
- проблема переобучения
- подбор – искусство

Модель глубокого обучения

Предварительные выводы:

- модели глубокого обучения состоят из взаимосвязанных слоёв
- слои хранят параметры модели
- слои состоят из нейронов (иногда называют *ядра*)
- выходное значение нейрона подаётся на функцию активации
- обучаются модели с помощью метода обратного распространения ошибки, который использует две идеи:
 - метод случайного градиентного спуска
 - производная сложной функции
- Сайт для представления, как нейросети обучаются
 - <http://playground.tensorflow.org>

Функция активации: СИГМОИДА

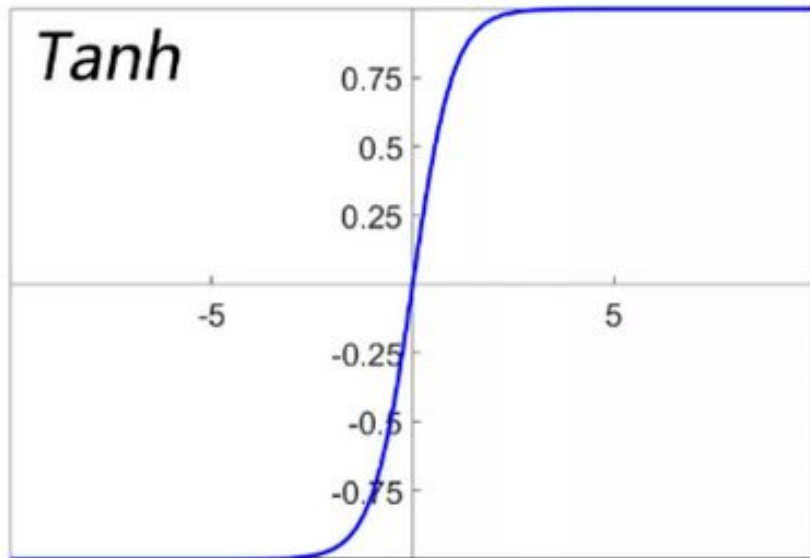


$$f(x) = \frac{1}{1 + e^{-x}}$$

Минусы:

- на плечах производная ноль
- ОДЗ не центрирована
- вычисление экспоненты

Функция активации: гиперболический тангенс



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

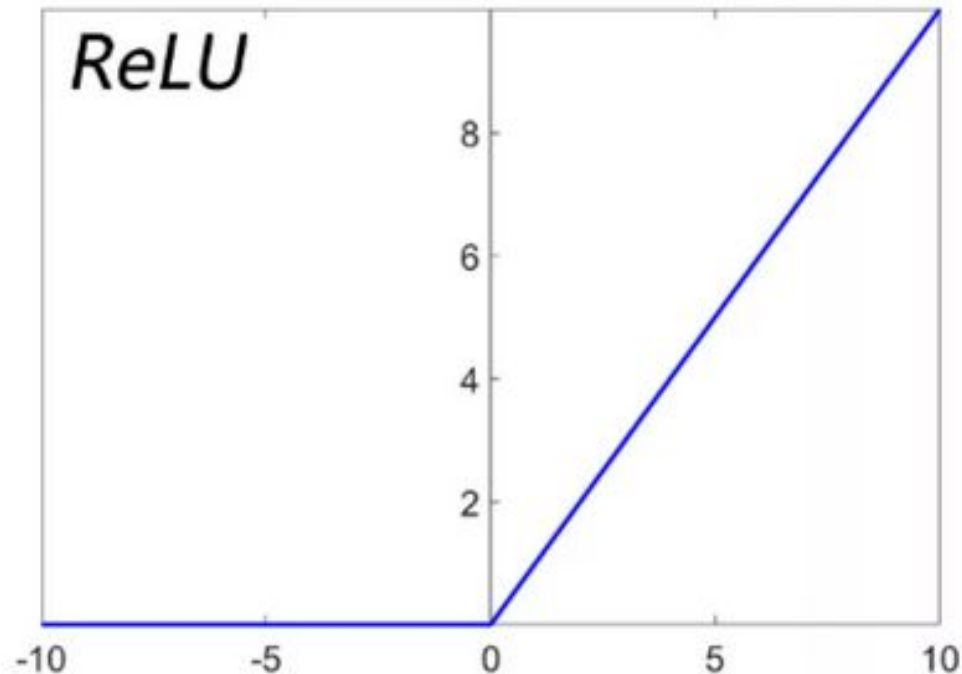
Плюсы:

+ центрирован

Минусы:

- на плечах производная ноль
- ОДЗ не центрирована
- вычисление экспоненты

Функция активации: ReLU



$$f(x) = \max(0, x)$$

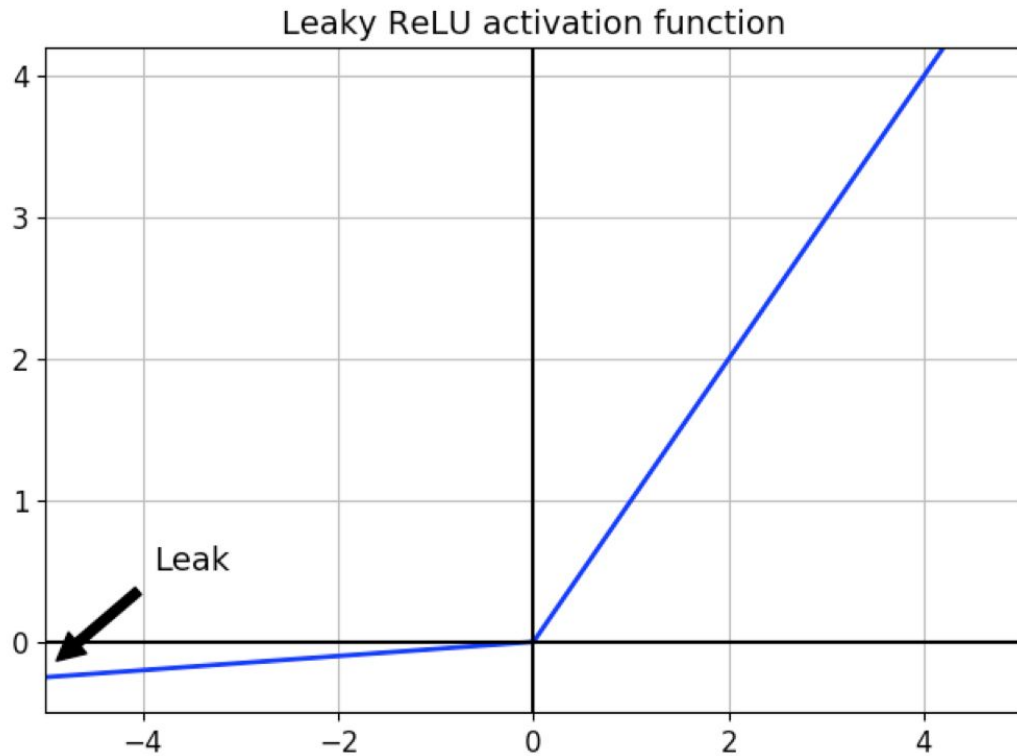
Плюсы:

- + центрирована
- + нелинейная
- + быстрая
- + дифференцируемая

Минусы:

- не центрирована

Функция активации: Leaky ReLU



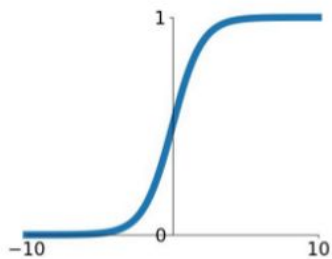
Плюсы:

- + центрирована
- + нелинейная
- + быстрая
- + дифференцируемая
- + «центрирована»

Различные функции активации

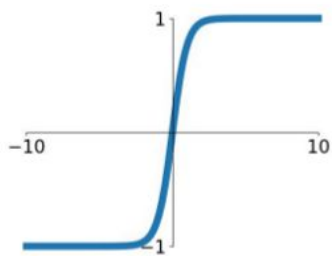
Sigmoid

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



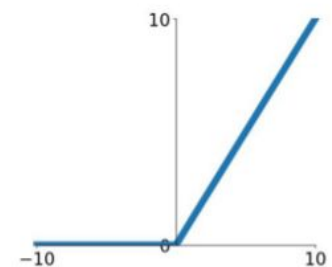
tanh

$$\tanh(x)$$



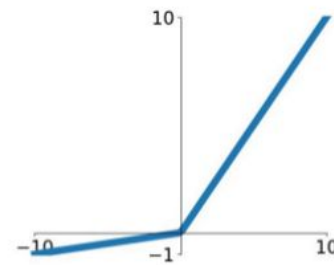
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

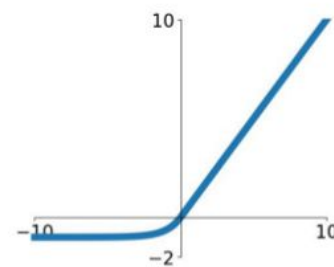


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



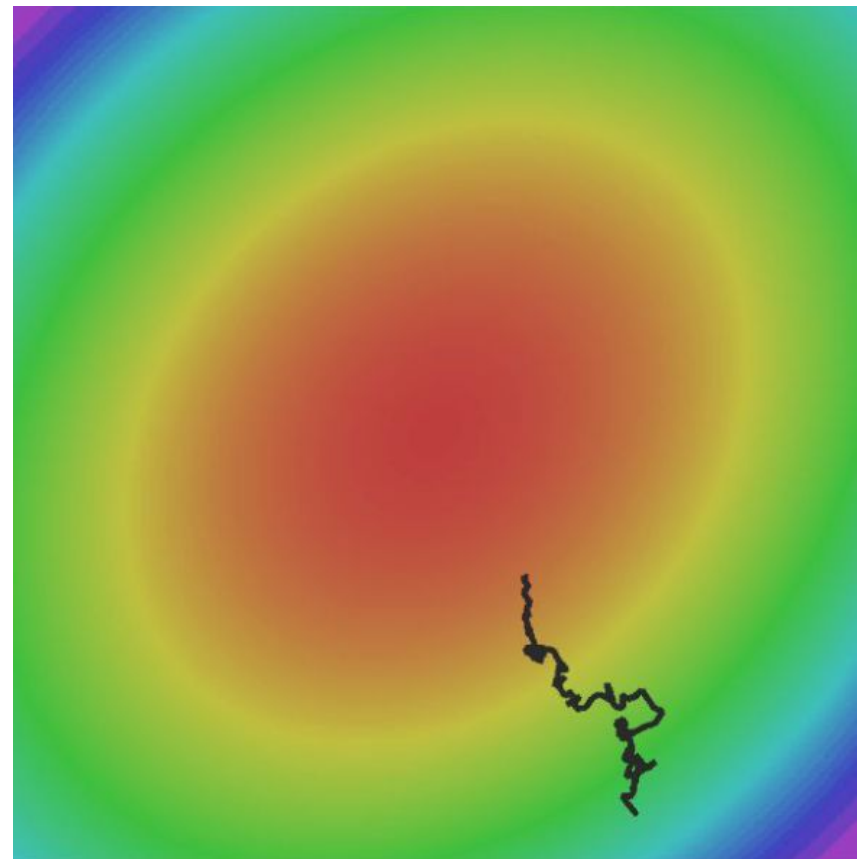
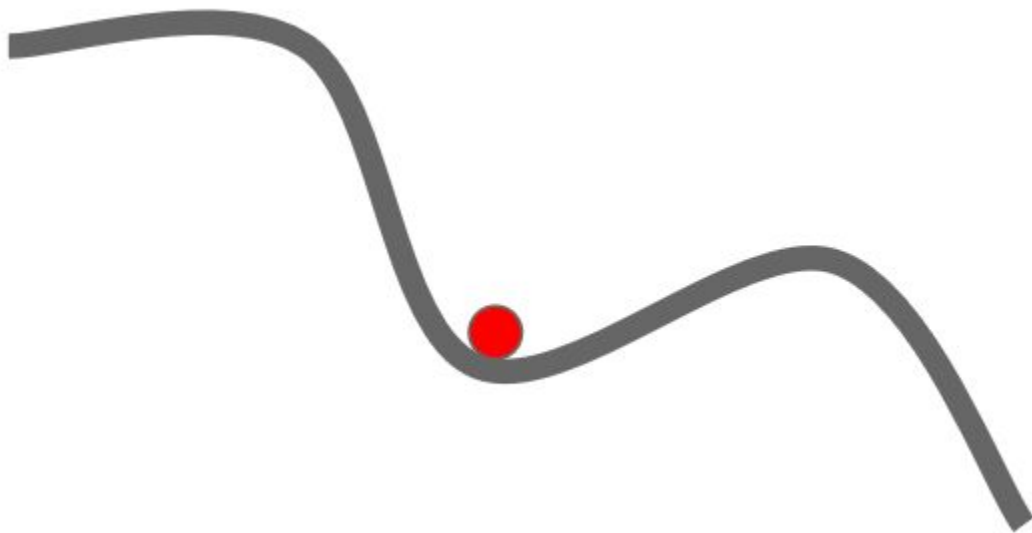
Различные функции активации

Замечание:

- ReLU – отправная точка
- изменяйте аккуратно скорость обучения
- попробуйте Leaky ReLU или ELU
- вряд ли гиперболический тангенс взлетит
- не используйте сигмоиду

Недостатки SGD

- застревание в локальных экстремумах
- «медленная» сходимость



Эвристики SGD: Momentum

Simple SGD

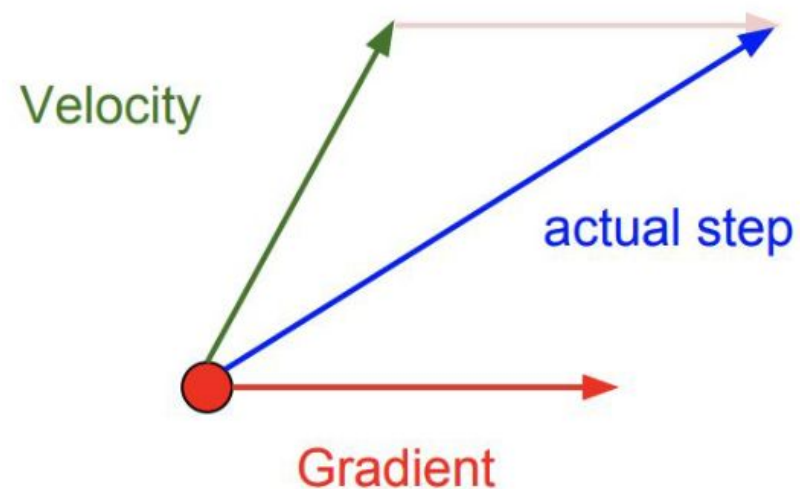
$$x_{t+1} = x_t - \alpha \nabla f(x_t)$$

SGD with momentum

$$v_{t+1} = \rho v_t + \nabla f(x_t)$$

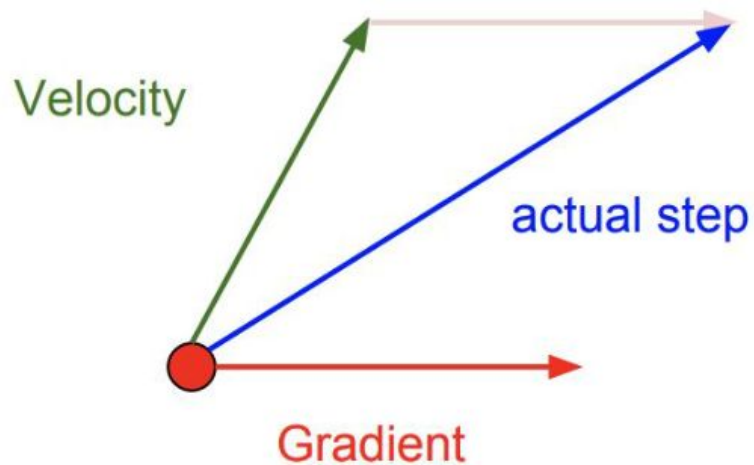
$$x_{t+1} = x_t - \alpha v_{t+1}$$

Momentum update:



Эвристики SGD: Momentum

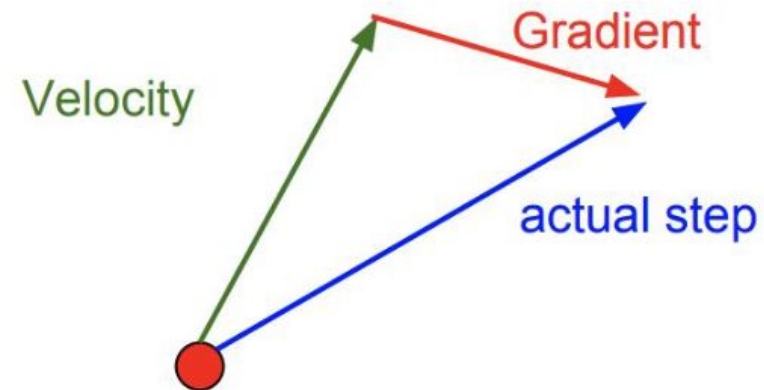
Momentum update:



$$v_{t+1} = \rho v_t + \nabla f(x_t)$$

$$x_{t+1} = x_t - \alpha v_{t+1}$$

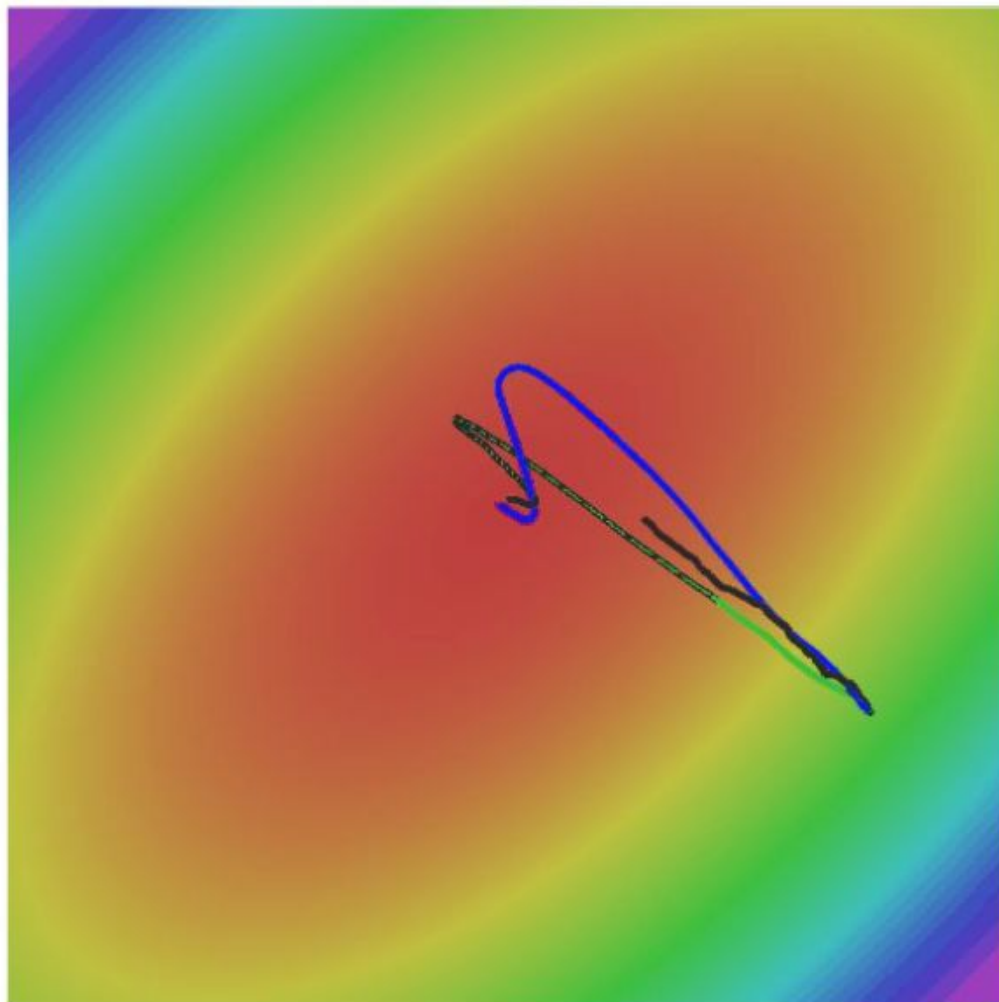
Nesterov Momentum



$$v_{t+1} = \rho v_t - \alpha \nabla f(x_t + \rho v_t)$$

$$x_{t+1} = x_t + v_{t+1}$$

Эвристики SGD: Momentum



- SGD
- SGD+Momentum
- Nesterov

Эвристики SGD: AdaGrad и RMSProp

Adagrad: SGD with cache

$$\text{cache}_{t+1} = \text{cache}_t + (\nabla f(x_t))^2$$

$$x_{t+1} = x_t - \alpha \frac{\nabla f(x_t)}{\text{cache}_{t+1} + \varepsilon}$$



RMSProp: SGD with cache with exp. Smoothing

$$\text{cache}_{t+1} = \beta \text{cache}_t + (1 - \beta)(\nabla f(x_t))^2$$

$$x_{t+1} = x_t - \alpha \frac{\nabla f(x_t)}{\text{cache}_{t+1} + \varepsilon}$$



Slide 29 Lecture 6 of Geoff Hinton's Coursera class

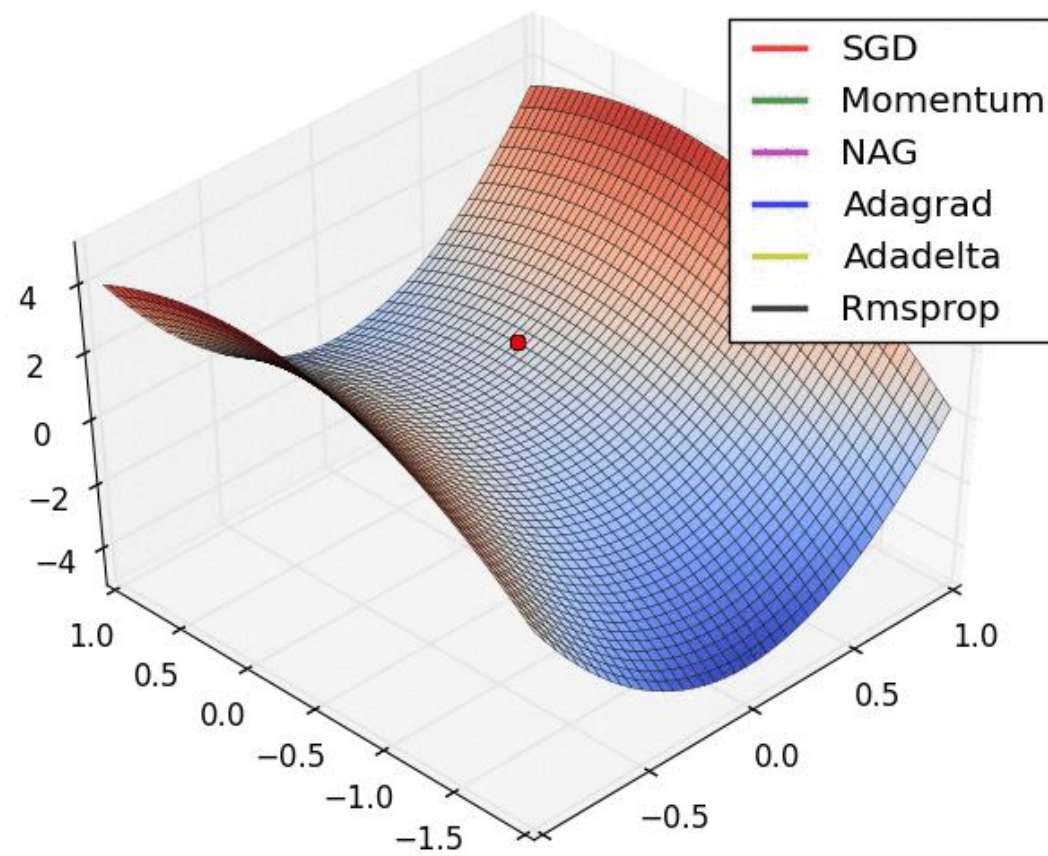
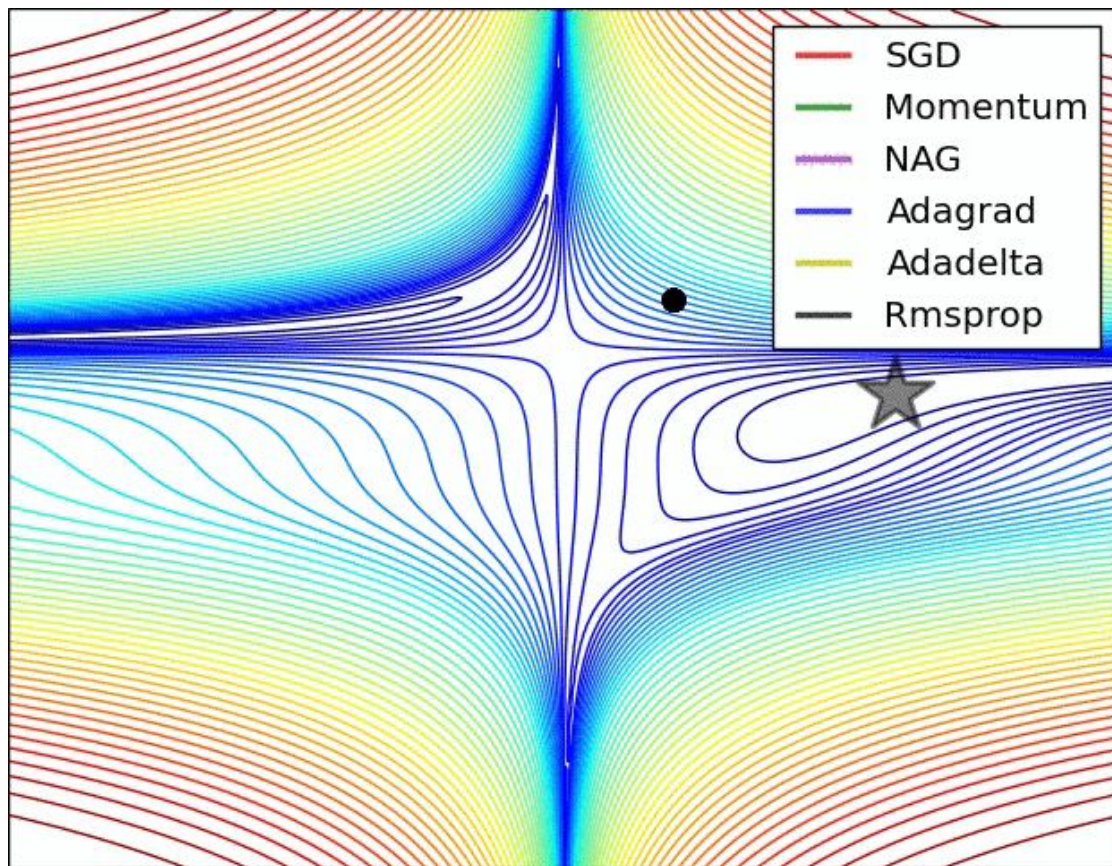
http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf

Эвристики SGD: Adam

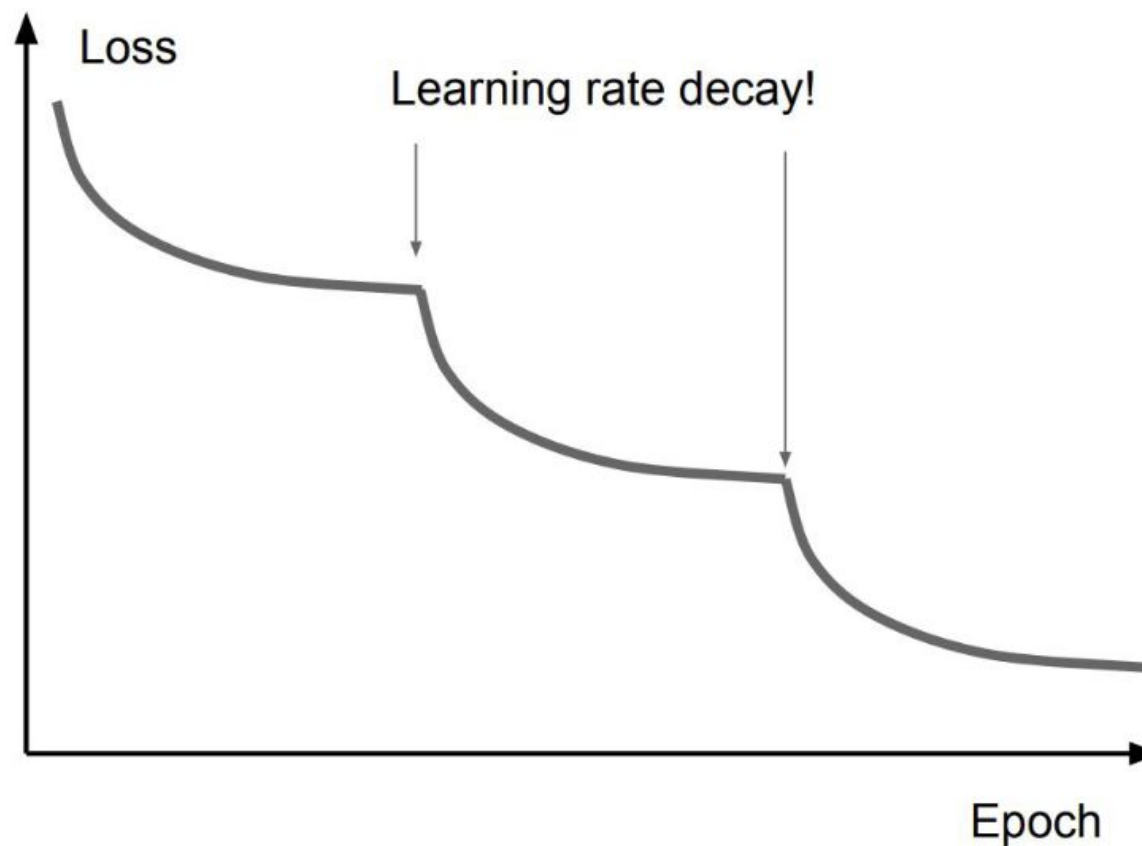
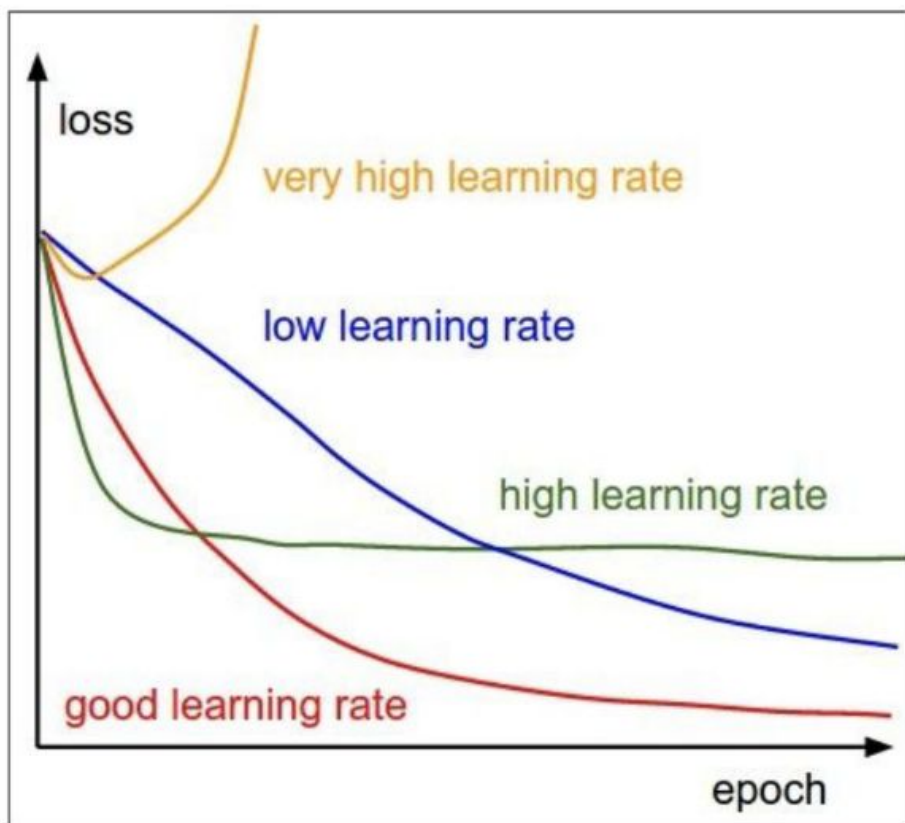
Включает в себя все перечисленные подходы

$$\begin{aligned}v_{t+1} &= \gamma v_t + (1 - \gamma) \nabla f(x_t) \\ \text{cache}_{t+1} &= \beta \text{cache}_t + (1 - \beta) (\nabla f(x_t))^2 \\ x_{t+1} &= x_t - \alpha \frac{v_{t+1}}{\text{cache}_{t+1} + \varepsilon}\end{aligned}$$

Эвристики SGD



Эвристики SGD: подбор скорости обучения



Эвристики SGD

Замечание:

- чем навороченней эвристика, тем больше требуется памяти для хранения кэшей, моментов и т.д.
- тем не менее Adam – хороший выбор для начала
 - байка про Карпатого
- уменьшайте скорость обучения по мере сходимости
- динамически проверяйте качество

Эвристики SGD



Andrej Karpathy ✓
@karpathy



3e-4 is the best learning rate for Adam, hands down.

6:01 AM · Nov 24, 2016 · [Twitter Web Client](#)

108 Retweets **461** Likes



Andrej Karpathy ✓ @karpathy · Nov 24, 2016



Replying to @karpathy

(i just wanted to make sure that people understand that this is a joke...)



9



3

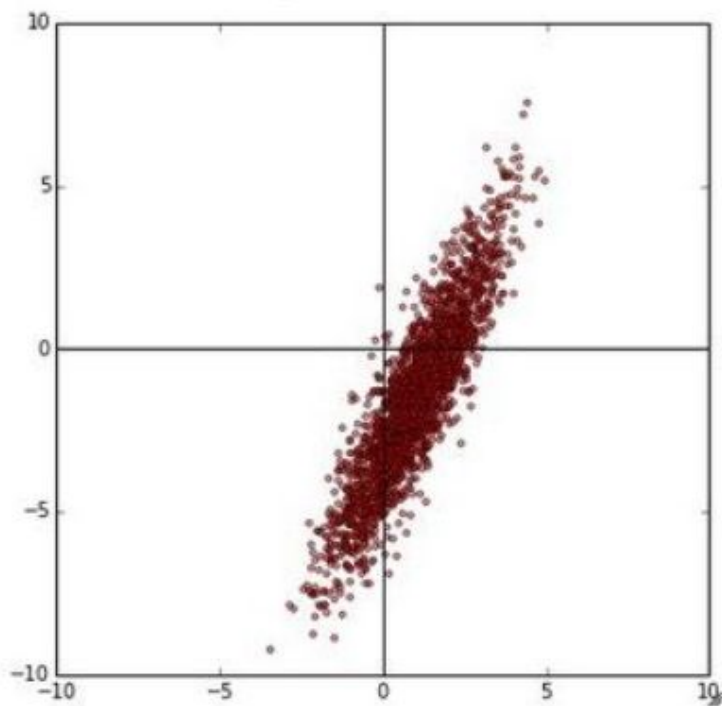


119

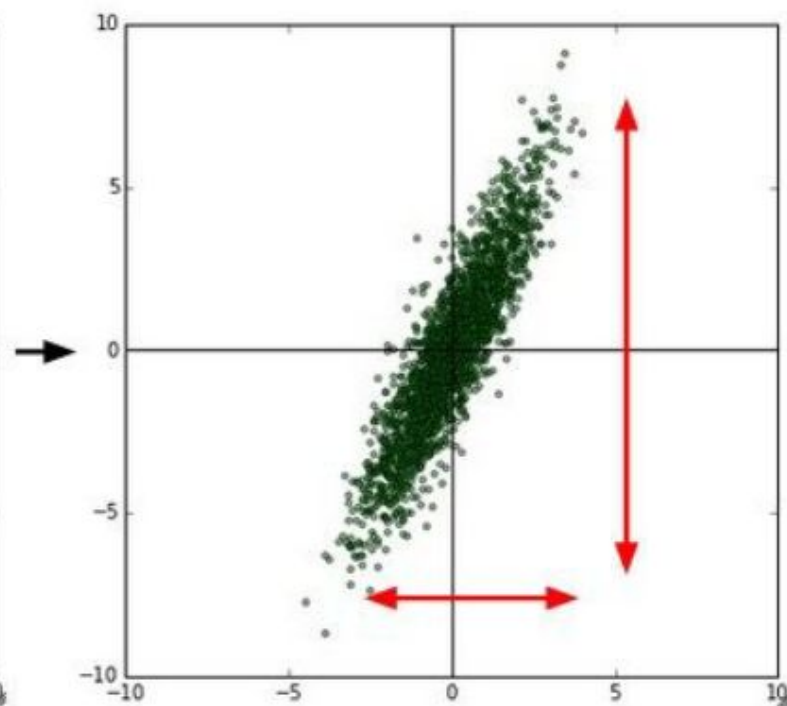


Нормировка признаков

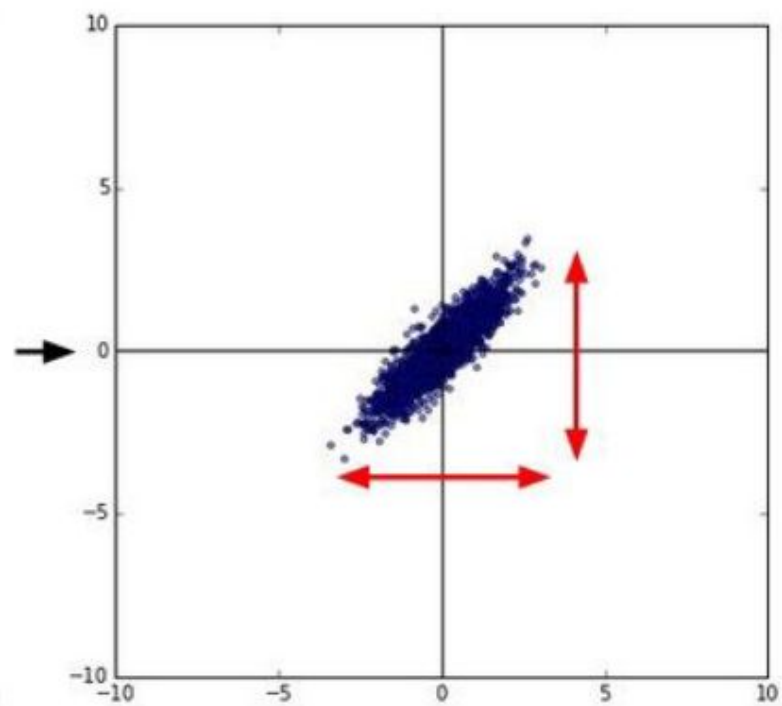
original data



zero-centered data



normalized data



Нормировка признаков

Замечание:

- модель, обученная на нормированных признаках, менее чувствительна к изменениям в данных
- градиентный спуск лучше сойдётся

Нормализация батча

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\dots m}\}$;

Parameters to be learned: γ, β

Output: $\{y_i = \text{BN}_{\gamma, \beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^m x_i \quad // \text{ mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2 \quad // \text{ mini-batch variance}$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \quad // \text{ normalize}$$

$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i) \quad // \text{ scale and shift}$$

Нормализация батча

Замечание:

- позволяет решить проблему метода обратного распространения ошибки: параметры модели оптимизируются «несогласовано»
- намного ускоряет сходимость
- позволяет увеличить скорость обучения

Регуляризация

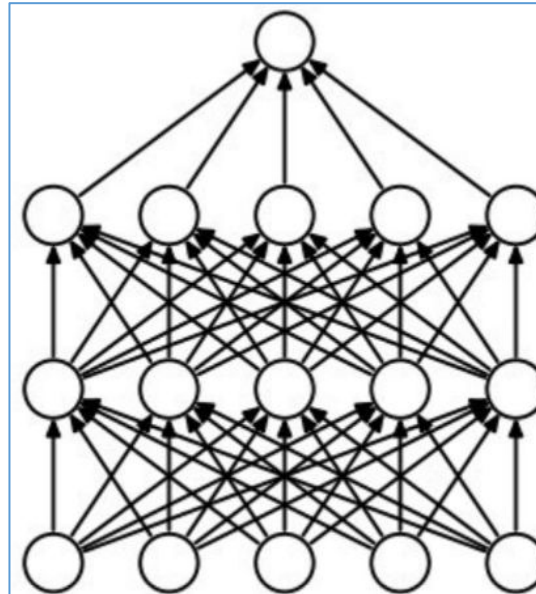
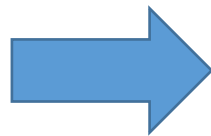
- L2 regularization:
- L1 regularization:
- Elastic Net (L1 + L2):

$$R(W) = \|W\|_2^2$$

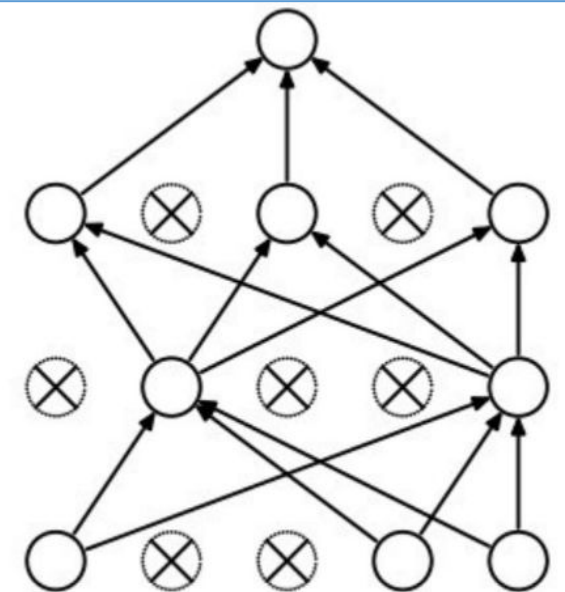
$$R(W) = \|W\|_1$$

$$R(W) = \beta \|W\|_2^2 + \|W\|_1$$

Прореживание
(Dropout)



(a) Standard Neural Net



(b) After applying dropout.

Регуляризация: прореживание

- прореживание (dropout) – приравнивание к нулю случайно выбираемых признаков на этапе обучения
- на этапе валидации используются все признаки, но выход слоя умножается на понижающий коэффициент
- позволяет предотвратить переобучение

Вывод

Улучшить сходимость позволяют:

- адаптивный градиентный шаг (Adam)
- специальная функция активации (ReLU)
- регуляризация и DropOut
- пакетная нормализация (batch normalization)
- инициализация параметров модели