Лекция 3. Функции активации. Инициализация весов

Денис Деркач, Дмитрий Тарасов

Слайды от А. Маевского, М. Гущина, А. Кленицкого, М Борисяка

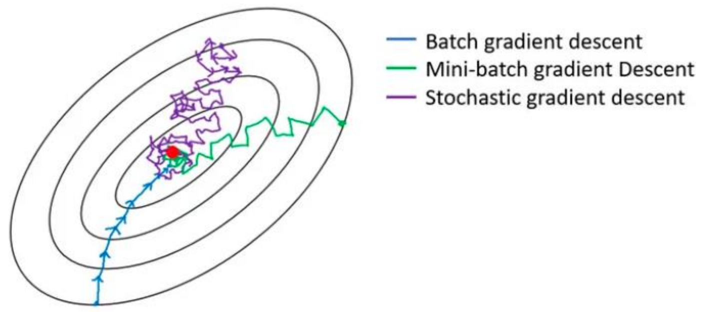
27 января 2025 года

Stochastic Gradient Descent
Backprop

Стохастический градиентный спуск

Stochastic gradient descent (SGD) – обновление весов вместе с каждым примером

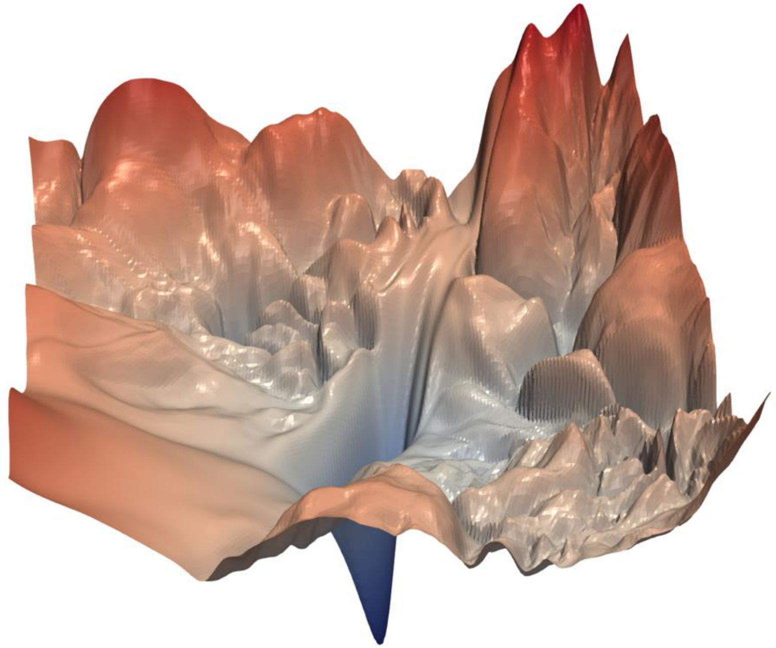
$$\theta_{t+1} = \theta_t - \alpha \nabla_\theta L(y_i, f(x_i, \theta))$$



- Несмещенная оценка полного градиента
- Если берем примеры из обучающей выборки в случайном порядке

Mini-batch stochastic gradient descent - обновляем веса после батча (пакета) из *B* примеров

$$\theta_{t+1} = \theta_t - \alpha \frac{1}{B} \sum_{i=1}^{B} \nabla_\theta L(y_i, f(x_i, \theta))$$



Можно эффективно использовать матричные вычисления

Обратное распространение ошибки

$$\hat{y} = w_3\sigma(w_2\sigma(w_1 x + b_1) + b_2) + b_3; \qquad L(y, \hat{y}) = (y - \hat{y})^2$$

$$x \to f_1 \to h_1 \to f_2 \to h_2 \to f_3 \to L$$

$$f_1 = w_1 x + b_1$$

$$h_1 = \sigma(f_1)$$

$$f_2 = w_2 h_1 + b_2$$

$$h_2 = \sigma(f_2)$$

$$f_3 = w_3 h_2 + b_3$$

$$L = (f_3 - y)^2$$

$$\frac{\partial L}{\partial f_3} = 2(f_3 - y)$$

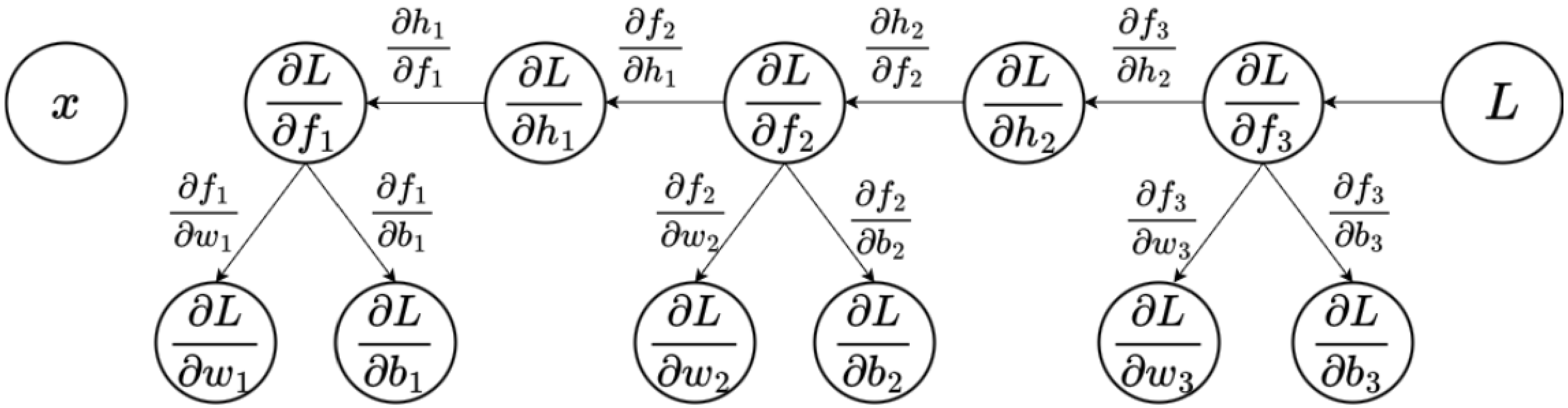
$$\frac{\partial L}{\partial h_2} = \left(\frac{\partial L}{\partial f_3}\right)\frac{\partial f_3}{\partial h_2}$$

$$\frac{\partial L}{\partial f_2} = \left(\frac{\partial L}{\partial f_3}\frac{\partial f_3}{\partial h_2}\right)\frac{\partial h_2}{\partial f_2}$$

$$\frac{\partial L}{\partial h_1} = \left(\frac{\partial L}{\partial f_3}\frac{\partial f_3}{\partial h_2}\frac{\partial h_2}{\partial f_2}\right)\frac{\partial f_2}{\partial h_1}$$

$$\frac{\partial L}{\partial f_1} = \left(\frac{\partial L}{\partial f_3}\frac{\partial f_3}{\partial h_2}\frac{\partial h_2}{\partial f_2}\frac{\partial f_2}{\partial h_1}\right)\frac{\partial h_1}{\partial f_1}$$

Добавление смешения



$$\frac{\partial L}{\partial f_3} = 2(f_3 - y)$$

$$\frac{\partial L}{\partial h_2} = \left(\frac{\partial L}{\partial f_3}\right)\frac{\partial f_3}{\partial h_2}$$

$$\frac{\partial L}{\partial f_2} = \left(\frac{\partial L}{\partial f_3}\frac{\partial f_3}{\partial h_2}\right)\frac{\partial h_2}{\partial f_2}$$

$$\frac{\partial L}{\partial h_1} = \left(\frac{\partial L}{\partial f_3}\frac{\partial f_3}{\partial h_2}\frac{\partial h_2}{\partial f_2}\right)\frac{\partial f_2}{\partial h_1}$$

$$\frac{\partial L}{\partial f_1} = \left(\frac{\partial L}{\partial f_3}\frac{\partial f_3}{\partial h_2}\frac{\partial h_2}{\partial f_2}\frac{\partial f_2}{\partial h_1}\right)\frac{\partial h_1}{\partial f_1}$$

$$\frac{\partial L}{\partial w_k} = \left(\frac{\partial L}{\partial f_k}\right)\frac{\partial f_k}{\partial w_k} = \frac{\partial L}{\partial f_k} \cdot h_{k-1}$$

$$\frac{\partial L}{\partial b_k} = \left(\frac{\partial L}{\partial f_k}\right)\frac{\partial f_k}{\partial b_k} = \frac{\partial L}{\partial f_k} \cdot 1$$

Немного интуиции

- Для простоты давайте пока опустим функции активации.
- Тогда выход нейронной сети, состоящей только из плотных слоев, буде :

$$\hat{y} = W_{out} \cdot \ldots \cdot W_{h2} \cdot W_{h1} x$$

- Обратите внимание, что градиент относительно любой из весовых матриц W_{hk} пропорционален **произведению** всех других матриц
- Например, для матриц 1×1, если все они имеют масштаб $S \in \mathbb{R}$, градиент g:

$$g \sim S^{m-1},$$

где m — глубина сети

- Если S слишком большое, градиенты **взорвутся**;
- Если S слишком маленькое, они **исчезнут**.

Больше интуиции

$$x \to \dots \xrightarrow{\mathbf{Output}_{i-1}} \boxed{\mathbf{Dense}_i \quad (W_i)} \xrightarrow{\mathbf{Output}_{i}} \boxed{\mathbf{Layer}_{i+1}} \xrightarrow{\mathbf{Output}_{i+1}} \dots \to \mathbf{Loss}$$

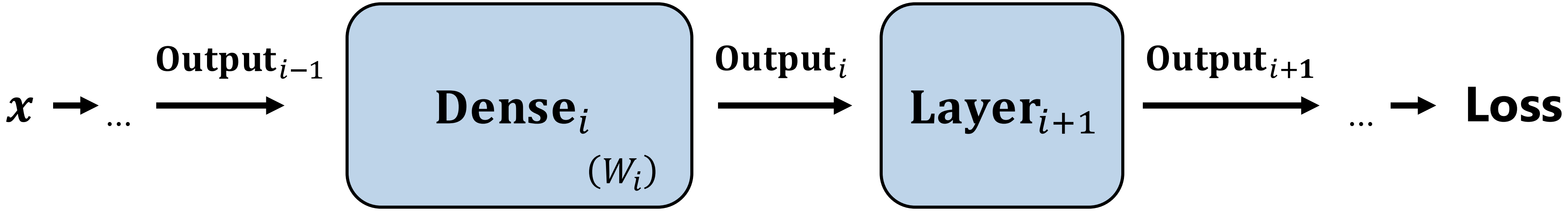
- Более подробно:

$$\frac{\partial \mathbf{Loss}}{\partial W_i} = \frac{\partial \mathbf{Loss}}{\partial \mathbf{Output}_i} \cdot \frac{\partial \mathbf{Dense}_i}{\partial W_i} = \underbrace{\frac{\partial \mathbf{Loss}}{\partial \mathbf{Output}_{i+1}} \cdot \frac{\partial \mathbf{Layer}_{i+1}}{\partial \mathbf{Output}_i}}_{\text{This will accumulate the product of the gradients for the subsequent layers}} \cdot \mathbf{Output}_{i-1}$$

- Идея: для стабильного обучения мы хотели бы «сохранить» масштаб градиентов на каждом шаге:

$$\mathrm{Var}\left(\frac{\partial \mathbf{Layer}_{i+1}}{\partial \mathbf{Output}_i} \cdot \frac{\partial \mathbf{Layer}_i}{\partial \mathbf{Output}_{i-1}}\right) \approx \mathrm{Var}\left(\frac{\partial \mathbf{Layer}_{i+1}}{\partial \mathbf{Output}_i}\right)$$

Ещё совсем много интуиции



- Аналогично, мы также хотели бы не масштабировать выходные данные на каждом этапе прямого прохода:

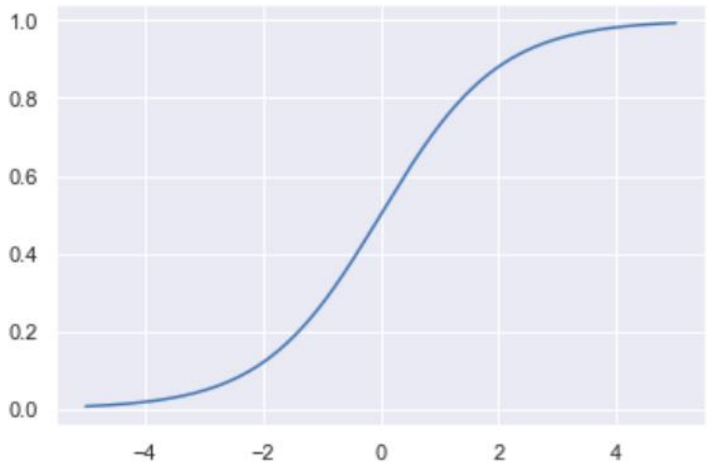
$$\mathrm{Var}\left(\mathbf{Layer}_{i+1}(\mathbf{Layer}_i(\mathbf{Output}_{i-1}))\right) \approx \mathrm{Var}\left(\mathbf{Layer}_i(\mathbf{Output}_{i-1})\right)$$

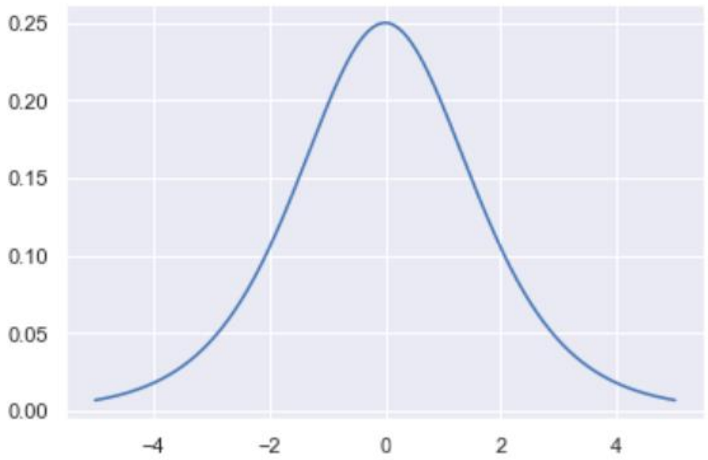
Функции активации (ещё про них)

Сигмоида

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

$$\sigma'(x) = \sigma(x)(1-\sigma(x))$$





- Насыщение на краях, градиент близок к нулю
- Максимальное значение производной 0,25
- Выход не центрирован вокруг нуля

Сигмоида - использование

По-прежнему используется в отдельных слоях, например:

- На выходе модели для бинарной классификации
- В гейтах (например, LSTM/GRU)

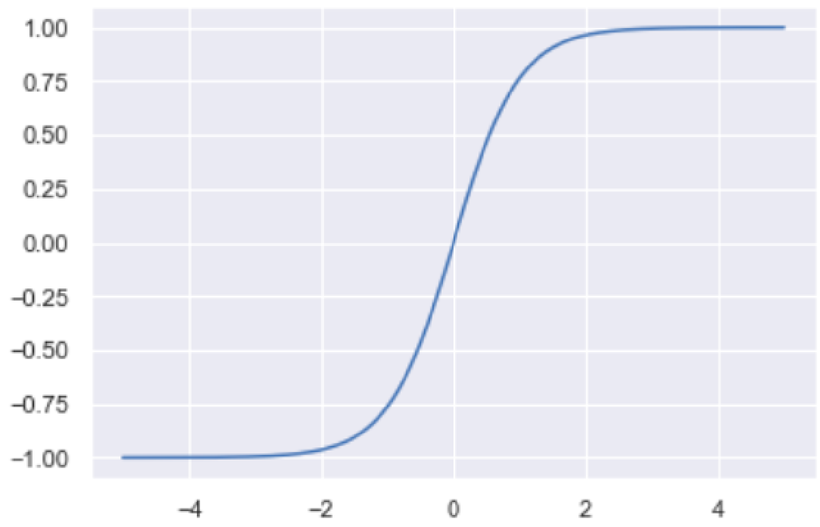
$$h_t = \sigma * h_{t-1} + (1 - \sigma) * h'_t$$

Гиперболический тангенс

Гиперболический тангенс

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

$$\tanh(x) = 2\sigma(2x) - 1$$

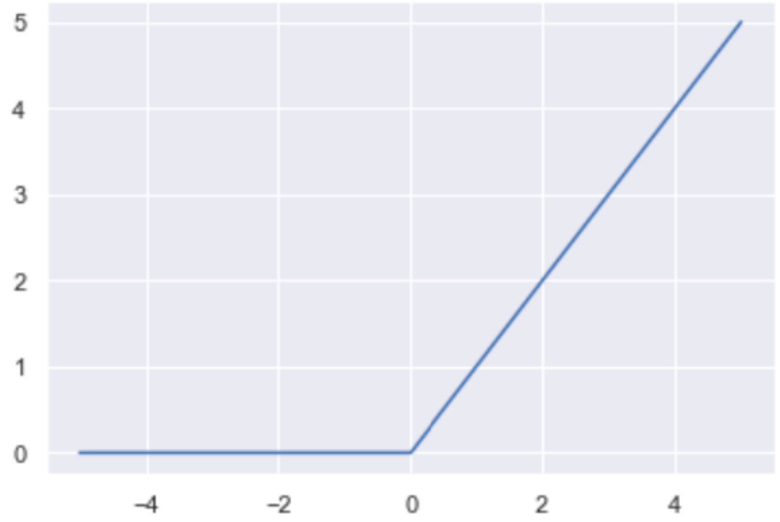


- Максимальное значение производной 1
- Выход центрирован, в нуле значение 0
- По-прежнему насыщение на краях

Rectified Linear Unit

$$f(x) = \max(0, x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

$$f'(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases}$$



Плюсы

- Нет насыщения, ненулевые градиенты
- Вычислительно очень дешево

Минус

- “Dead neurons” - нейроны, для которых всегда $x < 0$
- Проблема может усугубляться при большом learning rate

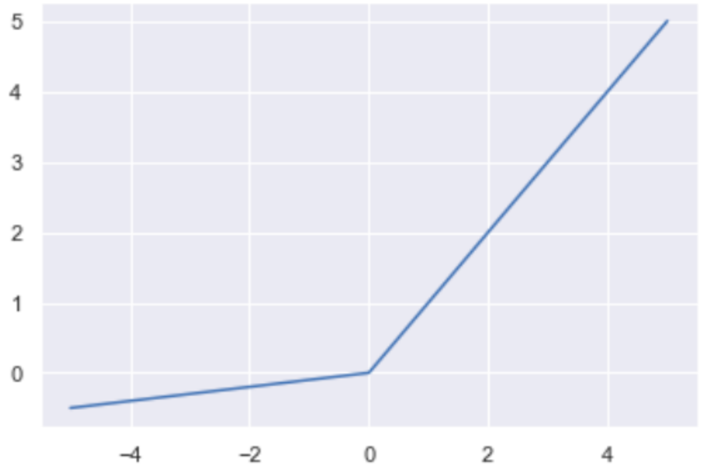
Модификации ReLU

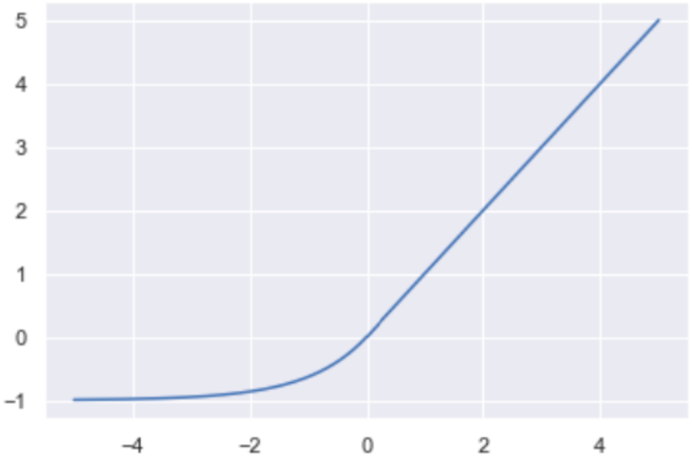
Leaky ReLU, PReLU

$$f(x) = \begin{cases} x, & x \geq 0 \\ \alpha x & x < 0 \end{cases}$$

ELU

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

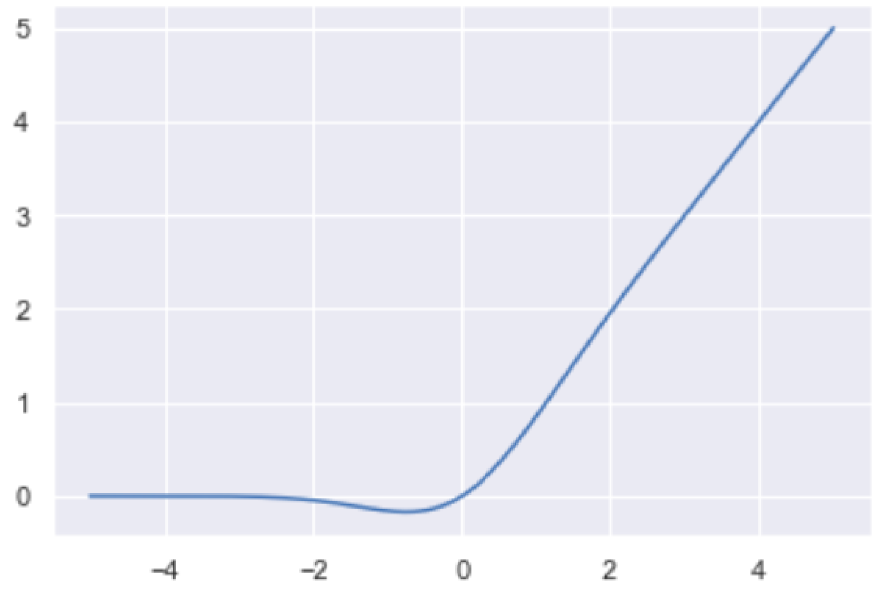




Решаем проблему “Dead neurons”

Gaussian Error Linear Unit

$$f(x) = xP(X \leq x),$$
$$X \sim N(0,1)$$



$$f(x) \approx \begin{cases} 0.5x + \frac{1}{2\pi}x^2, & |x| << 1 \\ ReLU(x), & |x| \to \infty \end{cases}$$

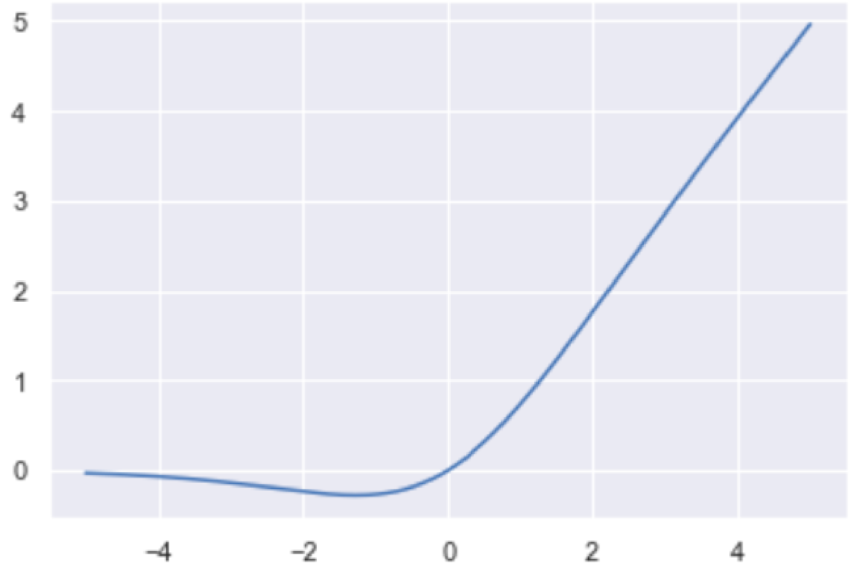
Обычно используется в трансформерах

Swish

Swish, он же SiLU (Sigmoid Linear Unit)

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

$$f'(x) = f(x) + \sigma(x)(1 - f(x))$$



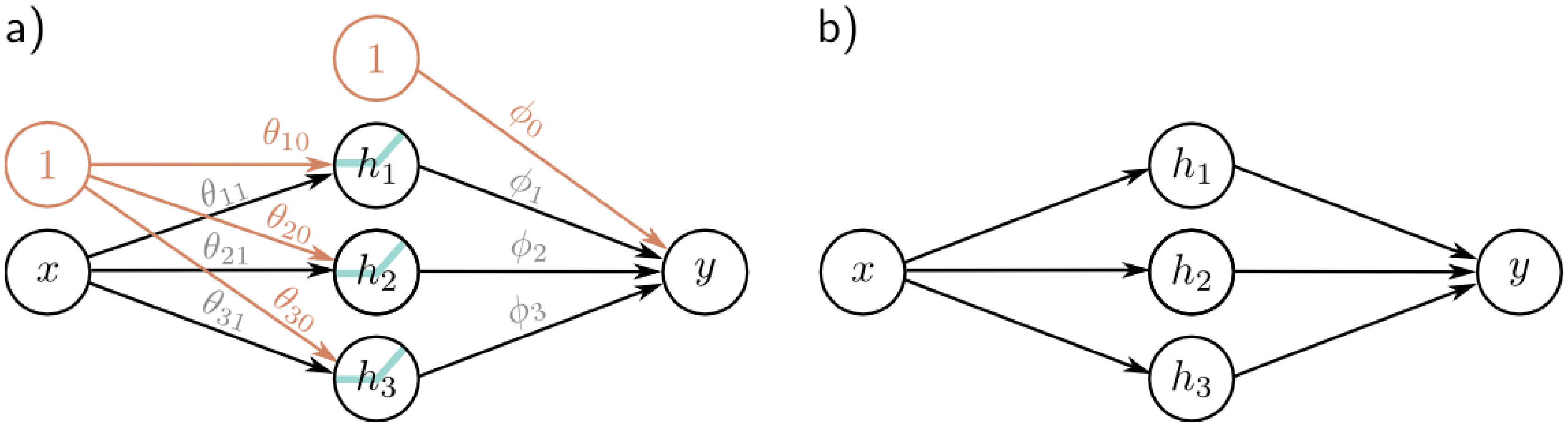
Влияние активации на нейросети

$$h_1 = a(\theta_{10} + \theta_{11} x)$$

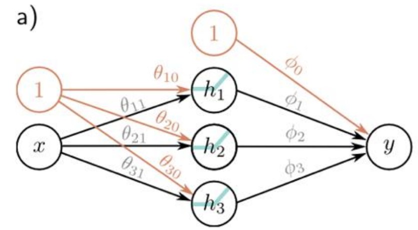
$$h_2 = a(\theta_{20} + \theta_{21} x)$$

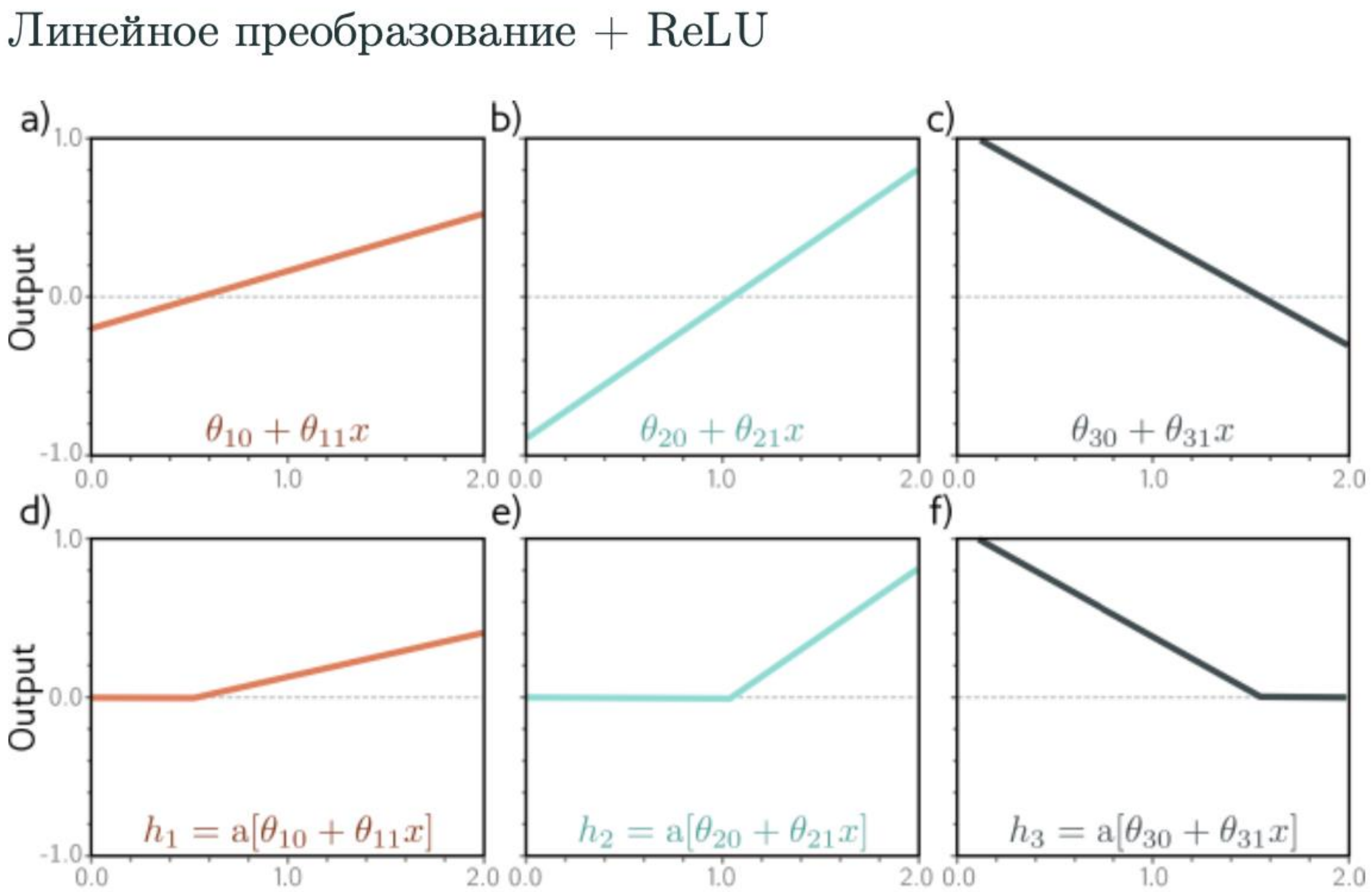
$$h_3 = a(\theta_{30} + \theta_{31} x)$$

$$y = \phi_0 + \phi_1 h_1 + \phi_2 h_2 + \phi_3 h_3$$

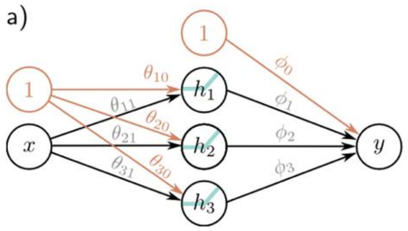


Нейросеть с ReLu

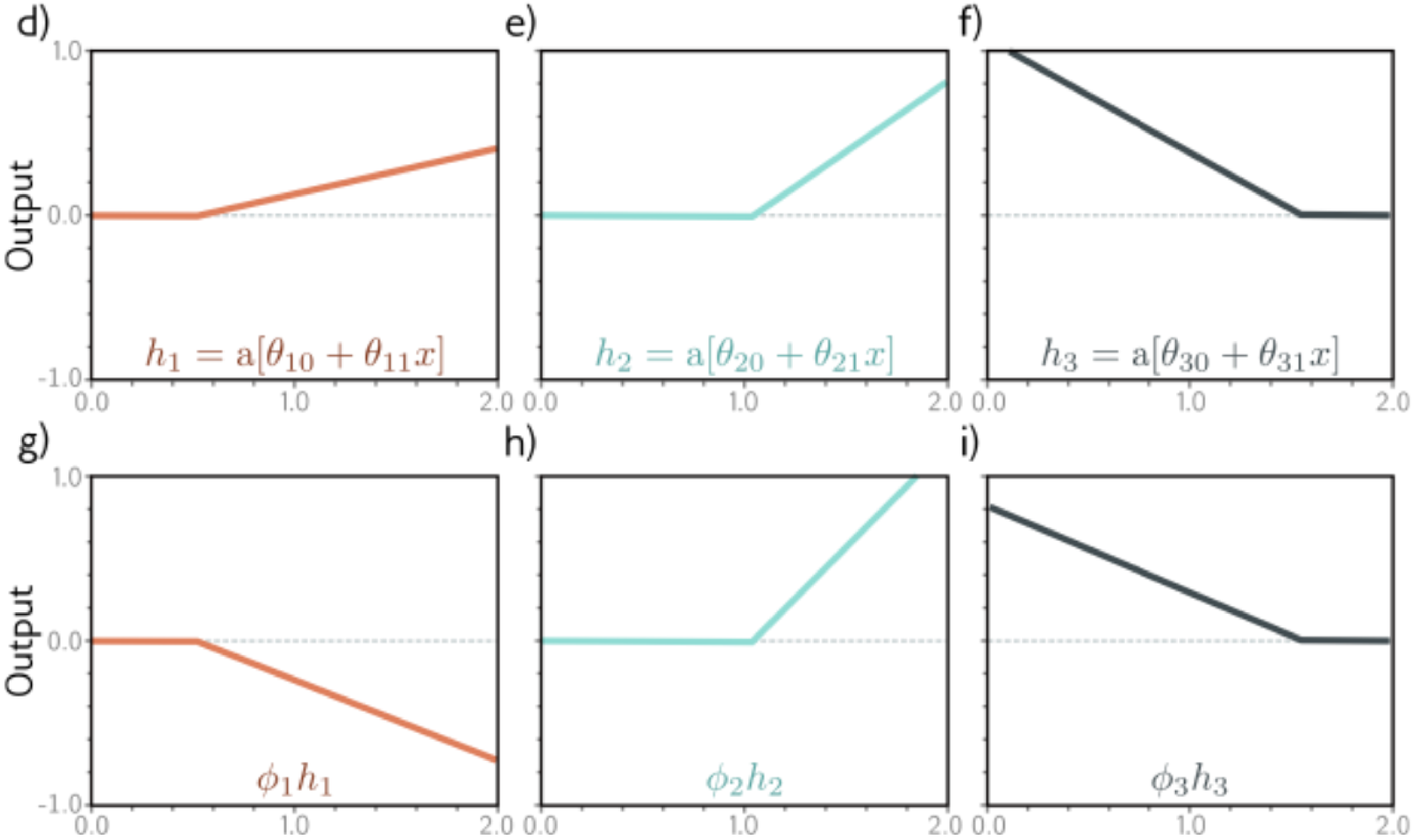




Нейросеть с ReLu

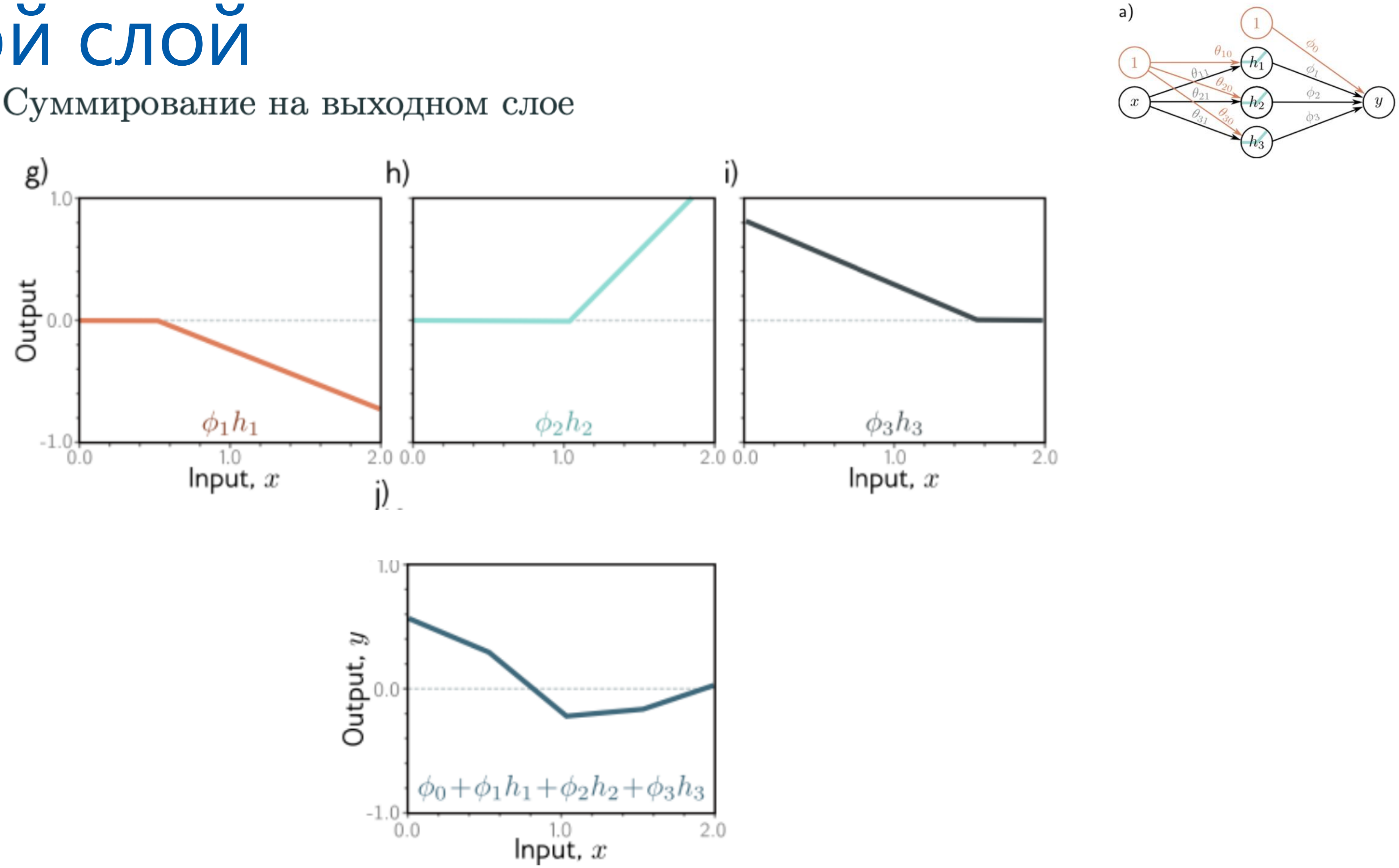


Умножение выходов скрытого слоя на веса на выходном слое

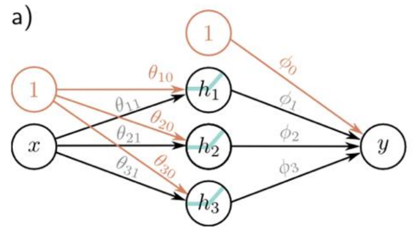


Выходной слой

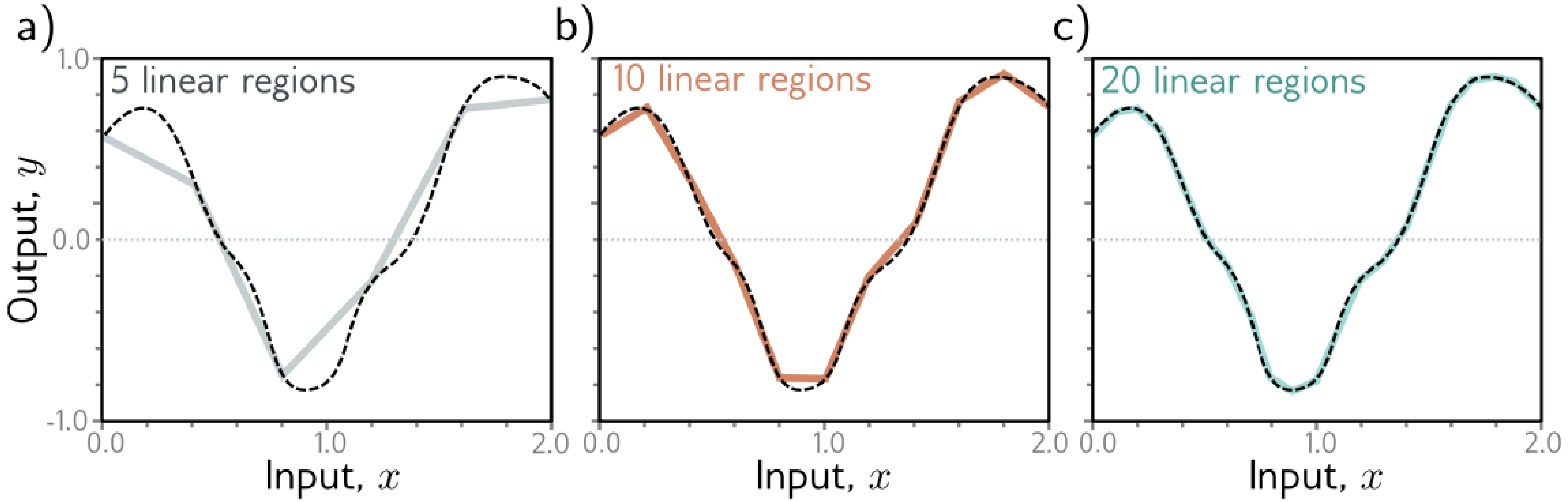
Суммирование на выходном слое



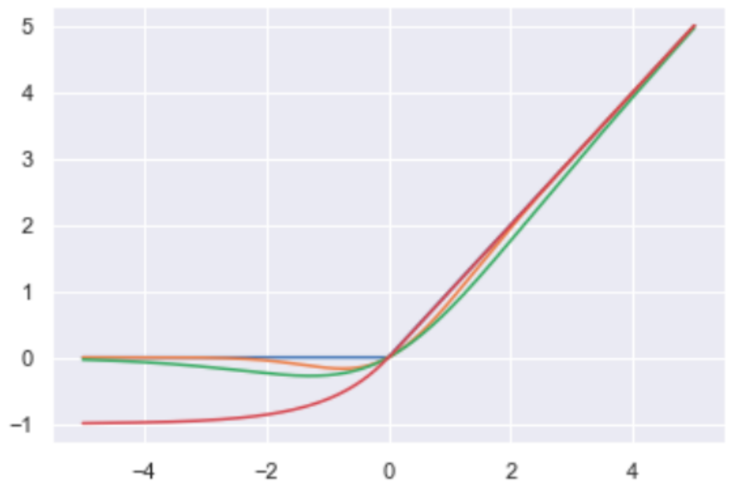
Универсальный аппроксиматор



Universal approximation theorem



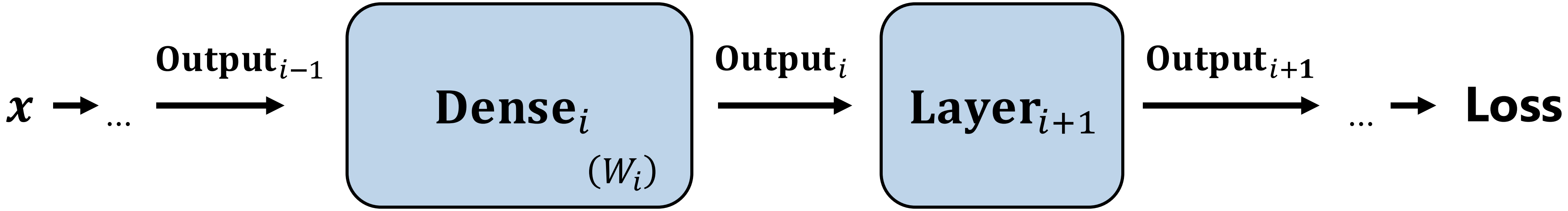
Тейк Хоум



- Избегать сигмоиды
- ReLU - хороший выбор в большинстве случаев
- Модификации ReLU могут чуть-чуть улучшить качество
- Подбор функции активации - не первый приоритет

Инициализация весов

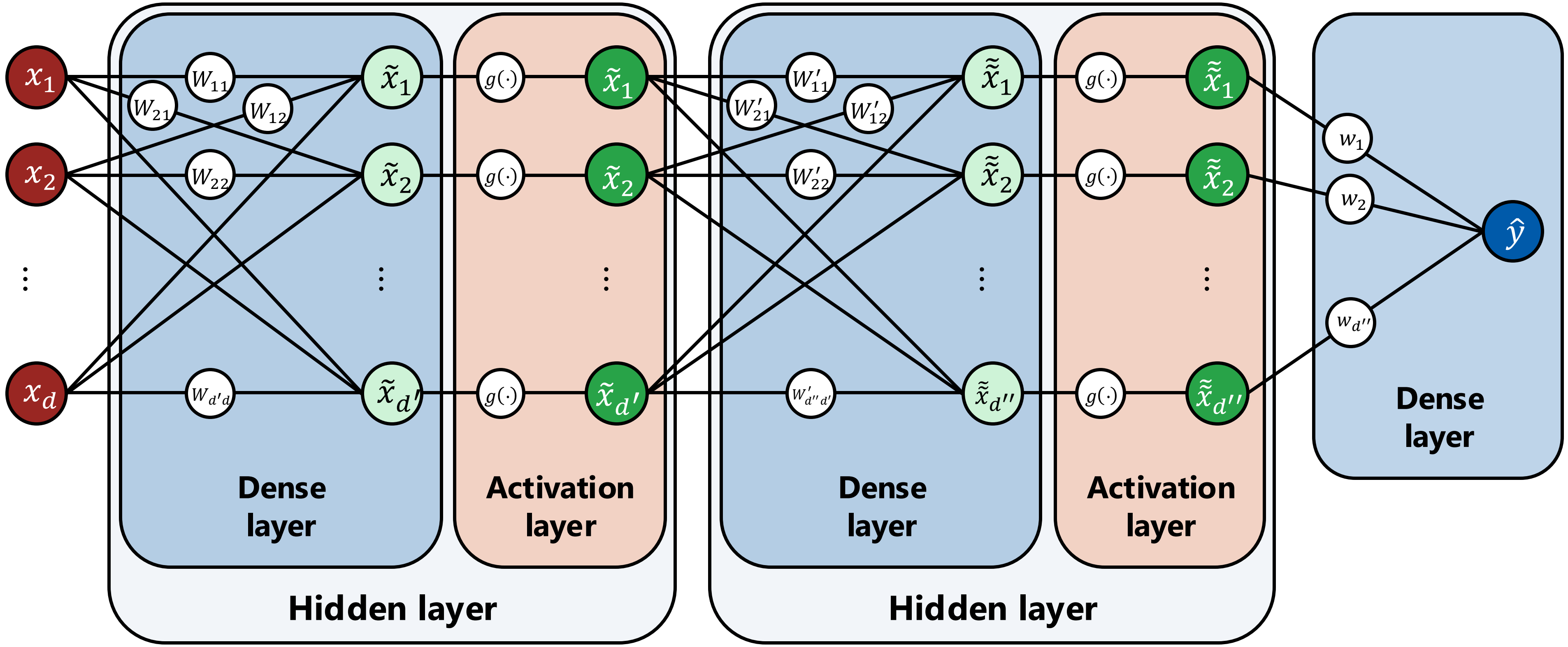
Reminder Ещё совсем много интуиции



- Аналогично, мы также хотели бы не масштабировать выходные данные на каждом этапе прямого прохода:

$$\mathrm{Var}\left(\mathbf{Layer}_{i+1}\left(\mathbf{Layer}_i\left(\mathbf{Output}_{i-1}\right)\right)\right) \approx \mathrm{Var}\left(\mathbf{Layer}_i\left(\mathbf{Output}_{i-1}\right)\right)$$

Initialization with a constant (?)



- What happens if we initialize all weights with the same value?
- Within each layer, the gradients for each of the weights will be the same as well ⇒ **updates will be the same** ⇒ network degrades!

Случайная инициализация

$$\mathrm{Var}\left(\frac{\partial \mathbf{Layer}_{i+1}}{\partial \mathbf{Output}_i} \cdot \frac{\partial \mathbf{Layer}_i}{\partial \mathbf{Output}_{i-1}}\right) \approx \mathrm{Var}\left(\frac{\partial \mathbf{Layer}_{i+1}}{\partial \mathbf{Output}_i}\right)$$

$$\mathrm{Var}\left(\mathbf{Layer}_{i+1}\left(\mathbf{Layer}_i(\mathbf{Output}_{i-1})\right)\right) \approx \mathrm{Var}\left(\mathbf{Layer}_i(\mathbf{Output}_{i-1})\right)$$

- Вообще, требования могут стать противоречивыми
- Например, для ReLU:

$$Var(W_{ij}) = \frac{2}{(\#\ \text{outgoing connections})}$$

$$Var(W_{ij}) = \frac{2}{(\#\ \text{incoming connections})}$$

- Часто можно использовать одно из них, или комбинацию:

$$Var(W_{ij}) = \frac{4}{(\#\ \text{outgoing connections}) + (\#\ \text{incoming connections})}$$

Xavier (Glorot) initialization

Understanding the difficulty of training deep feedforward neural networks - Glorot, X. & Bengio, Y. (2010)

- Для симметричных функций активации
- Идея - дисперсии выходов и градиентов на всех слоях должны быть одинаковыми

Рассмотрим один нейрон $y = w^T x = \sum_{i=1}^{n_{in}} w_i x_i$

В силу независимости и одинаковой распределенности w_i, x_i

$$Var[y] = Var\left[\sum_{i=1}^{n_{in}} w_i x_i\right] = \sum_{i=1}^{n_{in}} Var[w_i x_i] = n_{in} Var[w_i x_i]$$

$$Var[w_i x_i] = E[x_i]^2 Var[w_i] + E[w_i]^2 Var[x_i] + Var[w_i] Var[x_i] =$$
$$= Var[w_i] Var[x_i]$$

Xavier (Glorot) initialization

$$Var[y] = n_{in}Var[w_i x_i] = n_{in}Var[w_i]Var[x_i]$$

Получили условие для forward pass

$$n_{in}Var[w_i] = 1$$

Можно получить такое же условие для backward pass

$$n_{out}Var[w_i] = 1$$

Как объединить?

$$Var[w_i] = \frac{2}{n_{in} + n_{out}}$$

$$w_i \sim U\left[-\frac{\sqrt{6}}{\sqrt{n_{in} + n_{out}}}, \frac{\sqrt{6}}{\sqrt{n_{in} + n_{out}}}\right]$$

(Kaiming) He Intialization

Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification - He, K. et al. (2015)

- Для несимметричных функций активации (ReLU)
- Идея - дисперсии выходов или градиентов на всех слоях должны быть одинаковыми

$$\begin{aligned} Var[w_i x_i] &= E[x_i]^2 Var[w_i] + E[w_i]^2 Var[x_i] + Var[w_i] Var[x_i] = \\ &= E[x_i]^2 Var[w_i] + Var[w_i] Var[x_i] \\ &= Var[w_i] \left(E[x_i]^2 + Var[x_i] \right) = Var[w_i] E[x_i^2] \end{aligned}$$

Для ReLU

$$E[x_i^2] = \frac{1}{2} Var[y_{prev}]$$

$$Var[y] = n_{in} Var[w_i x_i] = \frac{1}{2} n_{in} Var[w_i] Var[y_{prev}]$$

(Kaiming) He Intialization

Либо используем условие для forward pass

$$Var[w_i] = \frac{2}{n_{in}}$$

$$w_i \sim N(0, \sqrt{2/n_{in}})$$

или

$$w_i \sim U\left[-\frac{\sqrt{6}}{\sqrt{n_{in}}}, \frac{\sqrt{6}}{\sqrt{n_{in}}}\right]$$

Либо условие для backward pass

$$Var[w_i] = \frac{2}{n_{out}}$$

Ортогональная инициализация

Ортогональные матрицы - повороты и отражения

$$A^T A = I$$

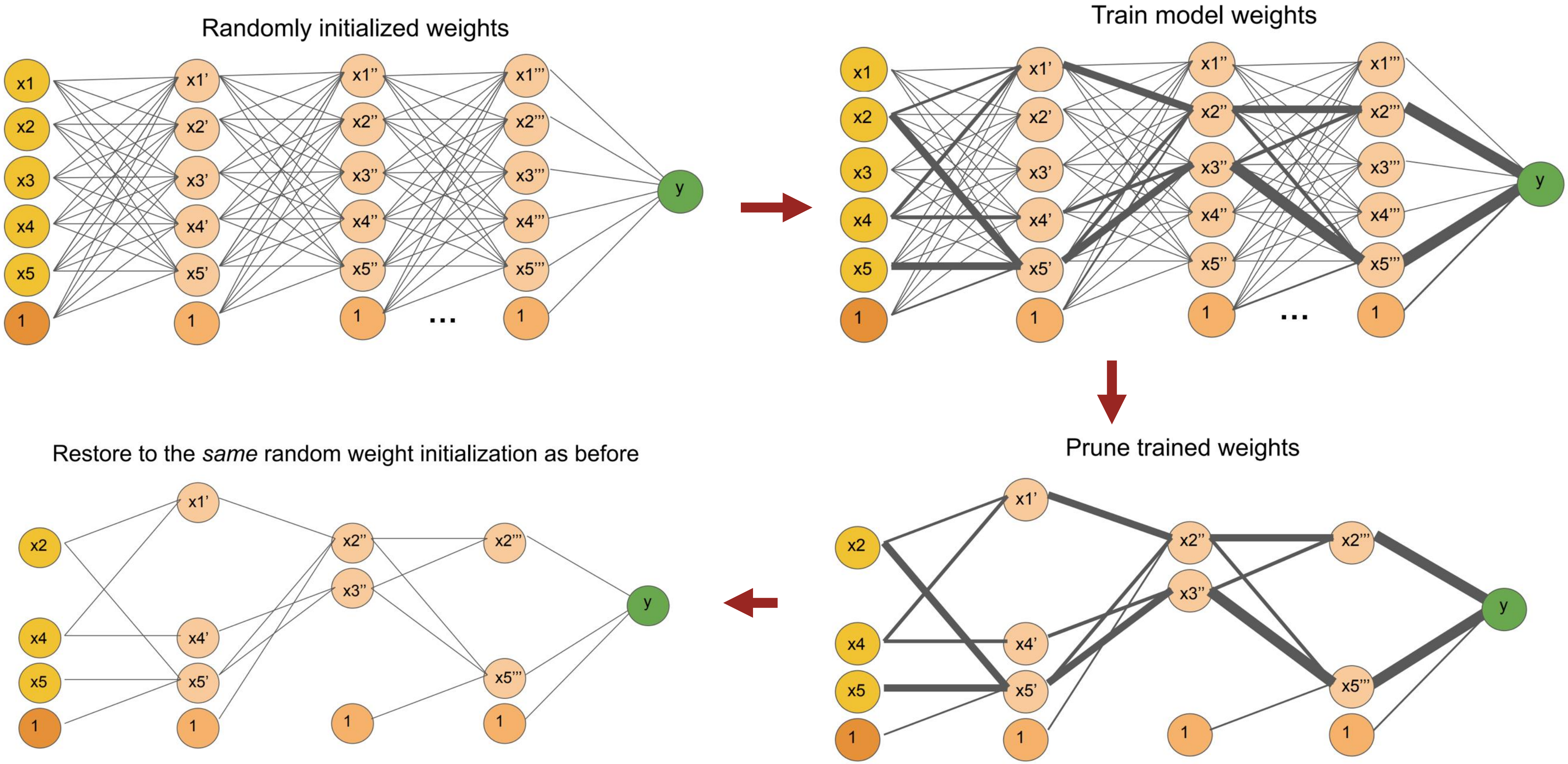
Столбцы (и строки) ортнормированны: $\sum_i A_{ij} A_{ik} = \delta_{ik}$

Не изменяют длины векторов:

$$||Ax|| = ||x||$$

Поэтому градиенты не будут взрываться и затухать

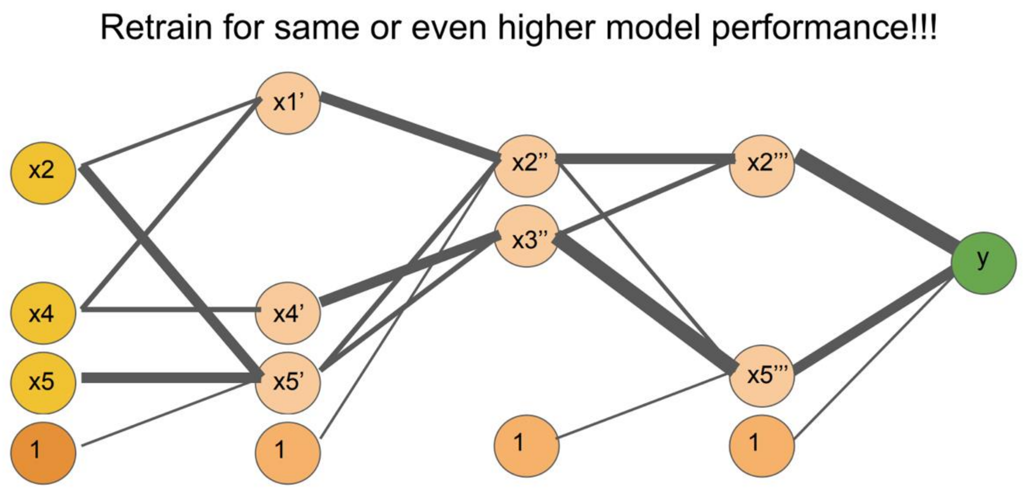
Лотерейные билеты



https://indico.fnal.gov/event/46852/contributions/204048/attachments/139265/174707/The-Lottery-Ticket-Hypothesis.pdf

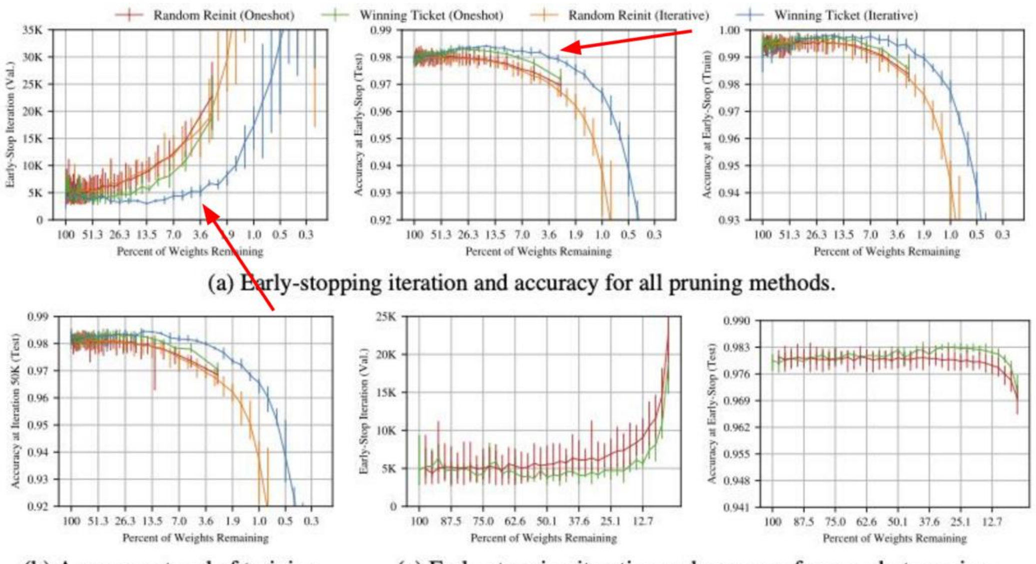
Лотерейные билеты

Случайно инициализированная, плотная нейронная сеть содержит подсеть, которая инициализируется таким образом, что при обучении в изоляции она может соответствовать тестовой точности исходной сети после обучения максимум на том же количестве итераций.



https://arxiv.org/abs/1803.03635

- Глубокие нейронные сети очень перепараметризованы
- Эта перепараметризация покупает много билетов
- С таким количеством билетов один часто будет победителем!



(a) Early-stopping iteration and accuracy for all pruning methods.

(b) Accuracy at end of training.

(c) Early-stopping iteration and accuracy for one-shot pruning.

Финальные мысли

- Правильная инициализация имеет гораздо большее значение, чем функция активации.
- Полностью случайная инициализация может также нарушить сходимость, потому надо использовать одну из предложенных.

Функции потерь

Оценка максимального правдоподобия

Модель $f(x,\theta)$ определяет параметры распределения вероятности $P(y|x)$:

$$P(y|x) = P(y|f(x,\theta))$$

Функция правдоподобия:

$$L(\theta) = P(D|\theta) = \prod_{i=1}^{N} P(y_i|f(x_i,\theta))$$

Метод максимального правдоподобия (maximum likelihood estimation):

$$\theta = \underset{\theta}{\operatorname{argmax}}\, L(\theta) = \underset{\theta}{\operatorname{argmax}} \left[\prod_{i=1}^{N} P(y_i|f(x_i,\theta))\right]$$

Оценка максимального правдоподобия

$$\theta = \underset{\theta}{\operatorname{argmax}} \left[\log \left(\prod_{i=1}^{N} P(y_i | f(x_i, \theta)) \right) \right]$$

$$= \underset{\theta}{\operatorname{argmax}} \left[\sum_{i=1}^{N} \log \left(P(y_i | f(x_i, \theta)) \right) \right]$$

$$= \underset{\theta}{\operatorname{argmin}} \left[-\sum_{i=1}^{N} \log \left(P(y_i | f(x_i, \theta)) \right) \right]$$

Negative log-likelihood loss:

$$\mathcal{L} = -\sum_{i=1}^{N} \log \left(P(y_i | f(x_i, \theta)) \right)$$

Ординарная регрессия

Предположим, что шум распределен нормально с центром в нуле:

$$y = f(x,\theta) + \epsilon, \qquad \epsilon \sim \mathcal{N}(0,\sigma^2)$$

то есть

$$P(y|x,\theta,\sigma) = \mathcal{N}(y|f(x,\theta),\sigma^2)$$
$$= \frac{1}{\sqrt{2\pi\sigma^2}} exp\left[-\frac{(y-f(x,\theta))^2}{2\sigma^2}\right]$$

$$\mathcal{L} = -\sum_{i=1}^{N} \log\left(\frac{1}{\sqrt{2\pi\sigma^2}} exp\left[-\frac{(y-f(x,\theta))^2}{2\sigma^2}\right]\right)$$
$$= \sum_{i=1}^{N}\left[-\log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) + \frac{(y-f(x,\theta))^2}{2\sigma^2}\right] \to (y-f(x,\theta))^2$$

Получили обычный метод наименьших квадратов

Обобщённые линейные модели и другие

- Можем предсказывать и среднее, и дисперсию:

$$\theta = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{N} \left[-\log\left(\frac{1}{\sqrt{2\pi f_2(x,\theta)^2}} \right) + \frac{(y - f_1(x,\theta))^2}{2 f_2(x,\theta)^2} \right]$$

- Если использовать распределение Лапласа, то получим MAE
- Если использовать распределение Пуассона, то можем предсказывать счетчики событий
- Если использовать бета-распределение, то можем предсказывать пропорции

Бинарная классификация

Распределение Бернулли

$$P(y|\lambda) = \lambda^y(1-\lambda)^{1-y} = \begin{cases} \lambda, & y = 1 \\ 1-\lambda, & y = 0 \end{cases}$$

Чтобы оценивать λ - вероятность от 0 до 1, используем сигмоиду:

$$f(x, \theta) = \sigma(x, \theta)$$

$$P(y|f(x,\theta)) = f(x,\theta)^y(1-f(x,\theta))^{1-y}$$

Функция потерь - binary cross-entropy

$$\mathcal{L}(\theta) = \sum_{i=1}^{N} [-y_i \log f(x_i, \theta) - (1-y_i)\log(1-f(x_i,\theta))]$$

Многоклассовая классификация

Softmax:

$$softmax_k(z) = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}}$$

$$P(y = k | f(x, \theta)) = softmax_k \left[f(x, \theta) \right]$$

Функция потерь - cross-entropy (aka softmax loss)

$$\mathcal{L}(\theta) = -\sum_{i=1}^{N} \log \left(softmax_{y_i} \left[f(x, \theta) \right] \right)$$

Softmax поведение на границах

$$softmax_k(z) = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}}$$

Из-за численного переполнения

- При больших отрицательных значениях z можем получить 0 в знаменателе
- При больших положительных значениях z можем получить бесконечность

Трюк:

$$\frac{e^{z_k+c}}{\sum_{j=1}^{K} e^{z_j+c}} = \frac{e^{z_k} e^{c}}{\sum_{j=1}^{K} e^{z_j} e^{c}} = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}}$$

$$z_k \to z_k - \max_j z_j$$

Другие функции потерь

- Hinge loss
- Focal loss
- Dice loss
- Triplet loss
- Contrastive loss

И многие другие..

Заключение

- Стартовые идеи, включая архитектурные особенности и инициализацию могут сильно повлиять на процесс сходимости сети.
- Подходы по выбору параметров основываются на знании задачи и способах оптимизации сети.