Глубинное обучение
Лекция 2: Механика нейросетей и backprop

Лектор: Антон Осокин

ФКН ВШЭ, 2018

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

Основной шаг обучения нейросетей

- Даны объект x, ответ y, значения параметров θ
- Итерация обучения:
  - Выходы нейросети f(x, θ)
  - Функция потерь $\ell(f(x,\theta),y)$

проход вперёд

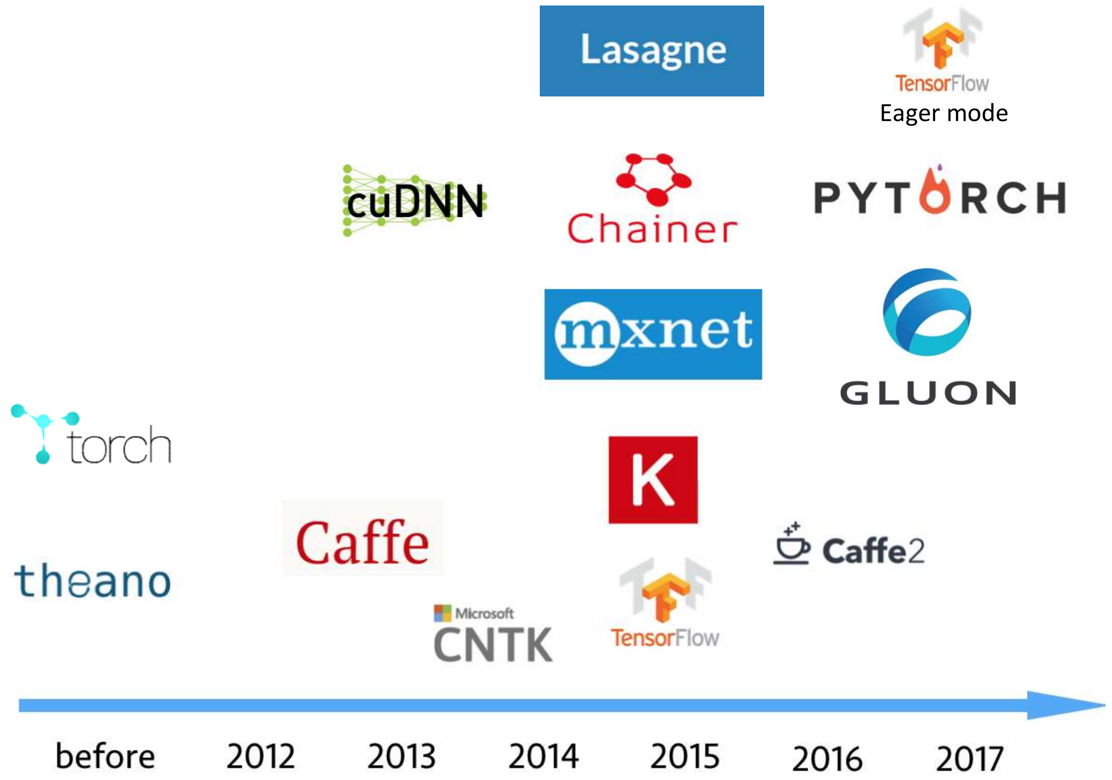
  - Градиент потерь по выходам нейросети
  - Градиент потерь по параметрам

проход назад

  - Шаг стохастической оптимизации

Deep learning frameworks

1. Есть все необходимые функции (CPU, GPU)
2. Организация единой системы
   – Собирает полные производные из производных слоев
   – Автоматическое дифференцирование



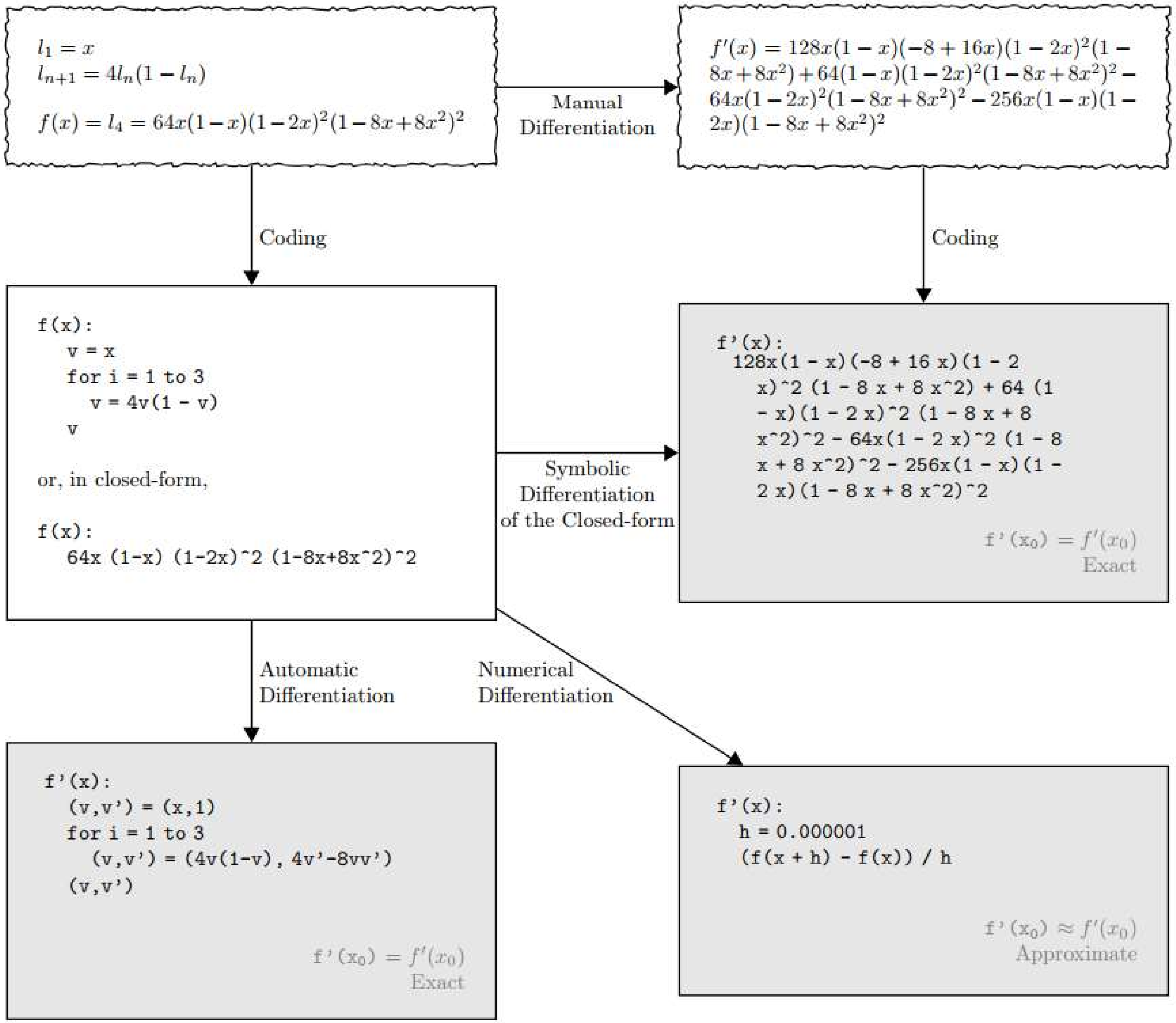
Производные на компьютере

[Baydin et al., 2017]

- Аналитическая формула
- Численный градиент
  - Конечные разности
- Символьное дифференцирование
  - Производная вычисляется алгебраически
  - Сначала формула, затем вычисления
- Алгоритмическое дифференцирование
  - Производная вычисляется как композиция элементарных производных во время выполнения программы

Производные на компьютере

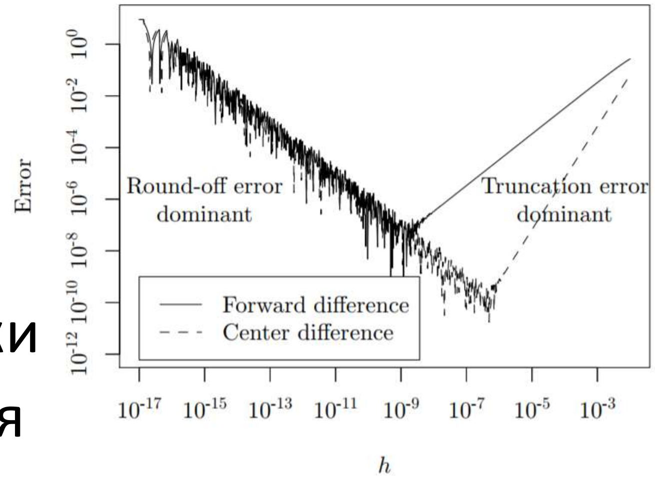
[Baydin et al., 2017]



Производные на компьютере

[Baydin et al., 2017]

- Аналитическая формула
  - Возможны очень эффективные реализации
  - Требуется выводить формулы, не всегда есть удобная запись
- Численный градиент (конечные разности)
  - Универсальный метод
  - Численные нестабильности (A + ε, A – A)
  - Медленно
- Символьное дифференцирование
  - Производная вычисляется алгебраически
  - Сначала вся формула, потом вычисления
- Алгоритмическое дифференцирование
  - Выполнение начинается сразу
  - Сохраняются промежуточные результаты



Алгоритмическое дифференцирование: прямой метод (forward mode)

- Производные вместе с выполнением
  - Функция $y = f(x), x \in \mathbb{R}^n, y \in \mathbb{R}^m$
  - Производные $v'_i = \frac{dv_i}{dx_1}$

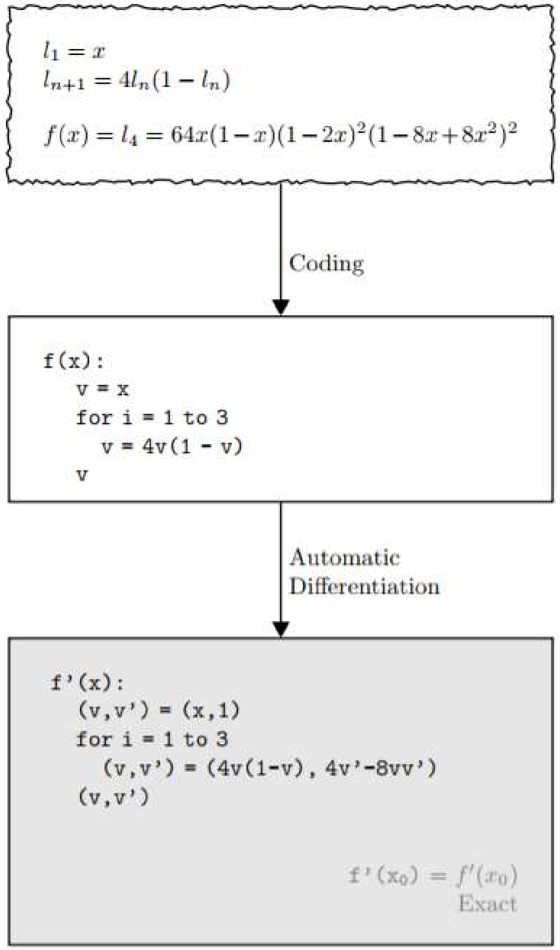
$$v_3 := v_1 v_2 \quad \Rightarrow \quad v'_3 := v'_1 v_2 + v_1 v'_2$$

$$v_3 := \ln(v_2) \quad \Rightarrow \quad v'_3 := \frac{1}{v_2} v'_2$$

- Якобиан вычисляется по столбцам

$$J = \begin{pmatrix} \frac{dy_1}{dx_1} & \cdots & \frac{dy_1}{dx_n} \\ \vdots & \ddots & \vdots \\ \frac{dy_m}{dx_1} & \cdots & \frac{dy_m}{dx_n} \end{pmatrix}$$

- Легко вычислять произведения Jr



Алгоритмическое дифференцирование: обратный метод (reverse mode, backprop)

- Сначала выполнение, потом производные
  - Функция $y = f(x), x \in \mathbb{R}^n, y \in \mathbb{R}^m$
  - Производные $\bar{v}_i = \frac{dy_1}{dv_i}$

$$v_3 := v_1 v_2 \quad \Rightarrow \quad \bar{v}_1 := \bar{v}_3 v_2,\ \bar{v}_2 := v_1 \bar{v}_3$$

$$v_3 := \ln(v_2) \quad \Rightarrow \quad \bar{v}_2 := \bar{v}_3 \frac{1}{v_2}$$

- Якобиан вычисляется по строкам

$$J = \begin{pmatrix} \frac{dy_1}{dx_1} & \cdots & \frac{dy_1}{dx_n} \\ \vdots & \ddots & \vdots \\ \frac{dy_m}{dx_1} & \cdots & \frac{dy_m}{dx_n} \end{pmatrix}$$

Если m = 1, то Якобиан = градиент^T

- Легко вычислять произведения rJ

Прямой или обратный метод?

- Производная из элементарных производных
- Функция $y = f_4(f_3(f_2(f_1(x)))), x \in \mathbb{R}^n, y \in \mathbb{R}^m$
- Полная производная

$$\frac{dy}{dx} = \frac{df_1(x)}{dx}\frac{\partial f_2(f_1)}{\partial f_1}\frac{\partial f_3(f_2)}{\partial f_2}\frac{\partial f_4(f_3)}{\partial f_3}$$

- Прямой метод

$$\frac{dy}{dx} = \frac{df_1(x)}{dx}\frac{\partial f_2(f_1)}{\partial f_1}\frac{\partial f_3(f_2)}{\partial f_2}\frac{\partial f_4(f_3)}{\partial f_3}$$

- Обратный метод

$$\frac{dy}{dx} = \frac{df_1(x)}{dx}\frac{\partial f_2(f_1)}{\partial f_1}\frac{\partial f_3(f_2)}{\partial f_2}\frac{\partial f_4(f_3)}{\partial f_3}$$

Прямой или обратный метод?

- Производная из элементарных производных
- Функция $y = f_4(f_3(f_2(f_1(x)))), x \in \mathbb{R}^n, y \in \mathbb{R}^m$

- Прямой метод эффективен, когда $n \ll m$

- Обратный метод эффективен когда $n \gg m$

- В МО, x – признаки и параметры, y – функция потерь
  - Обычно используется обратный метод
  - В сложных случаях можно использовать комбинации

- Оптимальный порядок умножения матриц можно найти с помощью динамического программирования

Как вычислить Гессиан на вектор?

- Функция $y = f(x_1, x_2, \ldots, x_n), x \in \mathbb{R}^n, y \in \mathbb{R}$
- Гессиан – матрица вторых производных

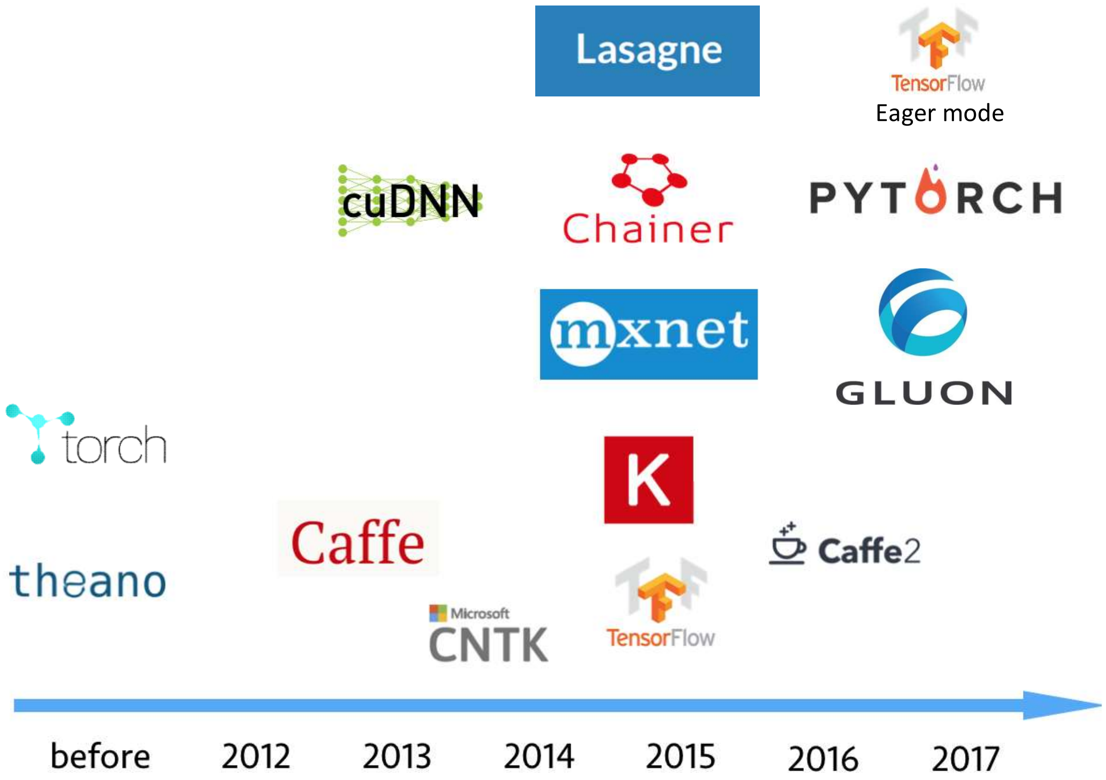
$$H = \begin{pmatrix} \frac{df}{dx_1 dx_1} & \cdots & \frac{df}{dx_1 dx_n} \\ \vdots & \ddots & \vdots \\ \frac{df}{dx_n dx_1} & \cdots & \frac{df}{dx_n dx_n} \end{pmatrix}$$

- Гессиан – очень большая матрица
- Стохастические методы второго порядке требуют вычисления Hv (пример – KFAC [Martens&Grosse, 2015])

Метод 1:	Метод 2:
1. $g(x) = v^T \nabla f(x)$	1. $g(x) = \nabla f(x)$
2. $Hv = \nabla[g(x)]$	2. $Hv = J_g v$

Deep learning frameworks

1. Есть все необходимые функции (CPU, GPU)
2. Организация единой системы
   – Собирает полные производные из производных слоев
   – Автоматическое дифференцирование



Библиотеки для глубинного обучения

1. Низкоуровневые операции: BLAS, LAPACK,
2. Линейная алгебра с дифференцированием
   – Отдельное построение вычислительного графа
   – Построение графа совместно с выполнением

   Eager mode
3. Высокоуровневые библиотеки

Рекомендации (январь 2018)

- Production

TensorFlow™ mxnet Caffe2

- Data science

Keras

- ML research

PYTORCH

TensorFlow Eager mode GLUON ?

Зачем знать про backprop?

Andrej Karpathy Follow

Director of AI at Tesla. Previously Research Scientist at OpenAI and PhD student at Stanford. I like to train deep neural nets on large datasets.

Dec 19, 2016 · 7 min read

Yes you should understand backprop

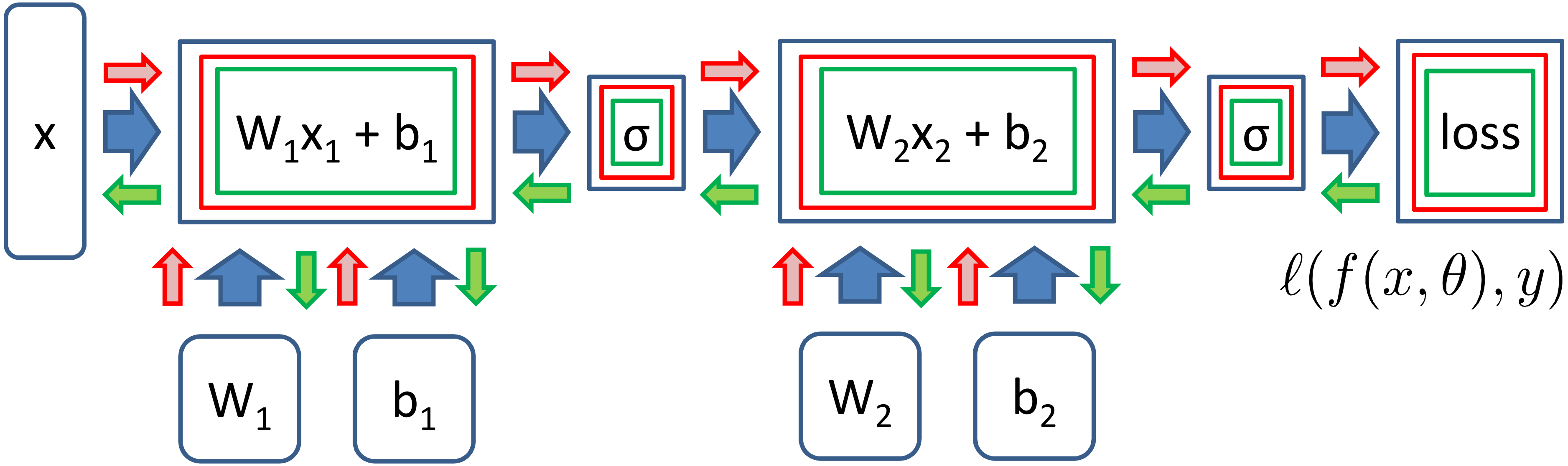
“Backprop – leaky abstraction!”

- Почему сеть не обучается?
- Почему сеть обучается медленно?

Back-propagation

[Rumelhart&McClelland, 1986]

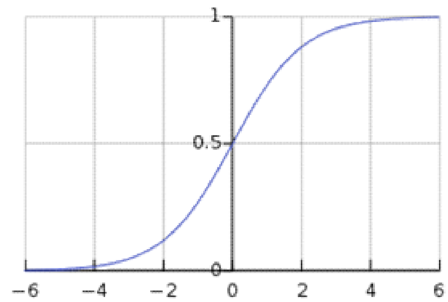
- Вход: x_i, y_i, параметры W_1, b_1, W_2, b_2
- Найти градиент по параметрам нейросети



1. Проход вперёд (вычисление слоёв и функции потерь)
2. Проход назад (вычисление градиентов)

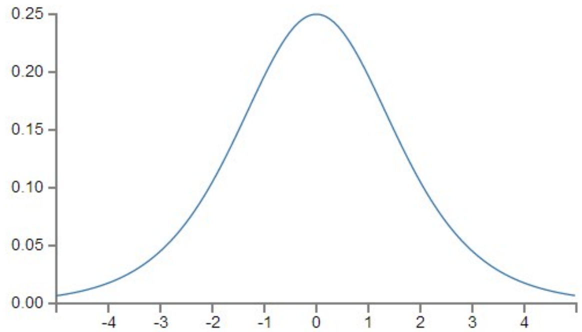
Проблемы в функциях активаций

[Karpathy, 2016]



- Сигмоида (или tanh)

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

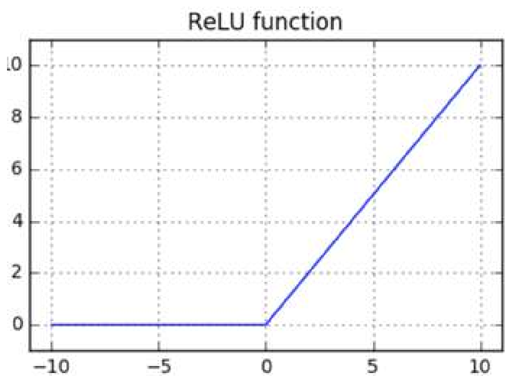


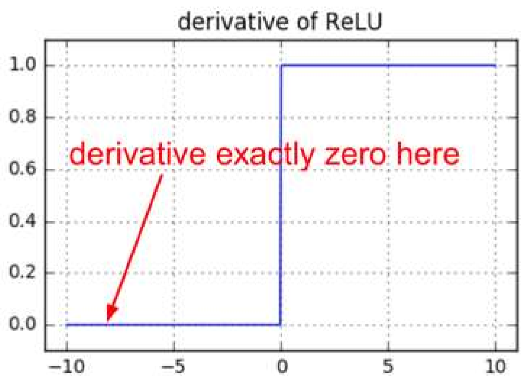
- Градиент $\frac{\partial \sigma(x)}{\partial x} = \sigma(x)(1 - \sigma(x))$
  - Градиент ≈ 0 при |x| ≥ 6
  - Максимум градиента = 0.25
    - При каждом умножении макс. градиента уменьшается

- $\mathrm{ReLu}(x) = \begin{cases} 0, x \leq 0, \\ x, x > 0 \end{cases}$

  $$\frac{\partial \mathrm{ReLu}(x)}{\partial x} = \begin{cases} 0, x \leq 0, \\ 1, x > 0 \end{cases}$$

  - Мёртвые узлы!





Инициализация сетей

- 'glorot' ('xavier') и 'kaiming'
- Инициализация линейного слоя:

[Glorot&Bengio, 2010]
[He et al., 2015]

$$y = w^T h(x)$$

- Идея: выбрать дисперсию весов так, чтобы активации были распределены стандартно-нормально
- Пусть w – i.i.d., x – i.i.d., x и w – независимы, $\mathbb{E}w_i = 0$

$$\mathrm{Var}[y] = d\mathrm{Var}[w_i h(x_i)] = d\mathrm{Var}[w_i]\mathbb{E}[h(x_i)^2]$$

- Выберем $\mathrm{Var}[w_i] := \frac{1}{d}\frac{\mathrm{Var}[x_i]}{\mathbb{E}[h(x_i)^2]}$
- Для ReLu $\mathrm{Var}[w_i] := \frac{2}{d}$, или $w_i \sim \mathcal{N}(0, \sigma^2 := \frac{2}{d})$

Заключение

- Backprop –автоматическое дифференцирование
  – По умолчанию используется обратный метод
  – В ряде случае нужны и другие методы

- При обучении моделей надо думать о градиенте
  – Затухание и взрыв градиентов
  – Инициализация
  – Специальные архитектуры