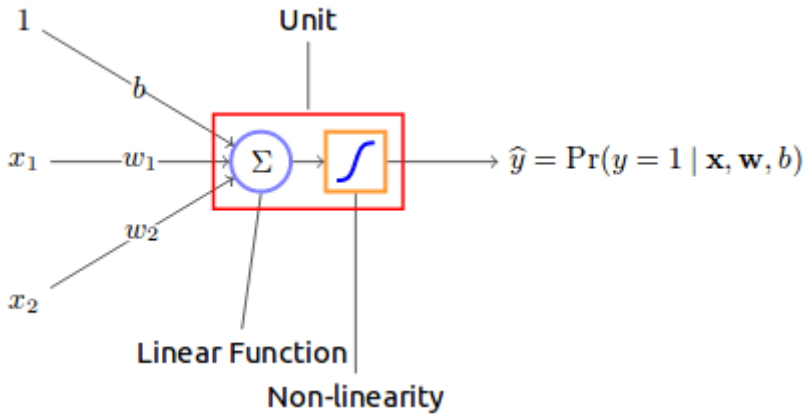
Глубокое обучение
Нейронные сети

Дьяконов А.Г.

Московский государственный университет
имени М.В. Ломоносова (Москва, Россия)

Простейшая нейросеть – 1 нейрон



Разделяющая поверхность – линейная

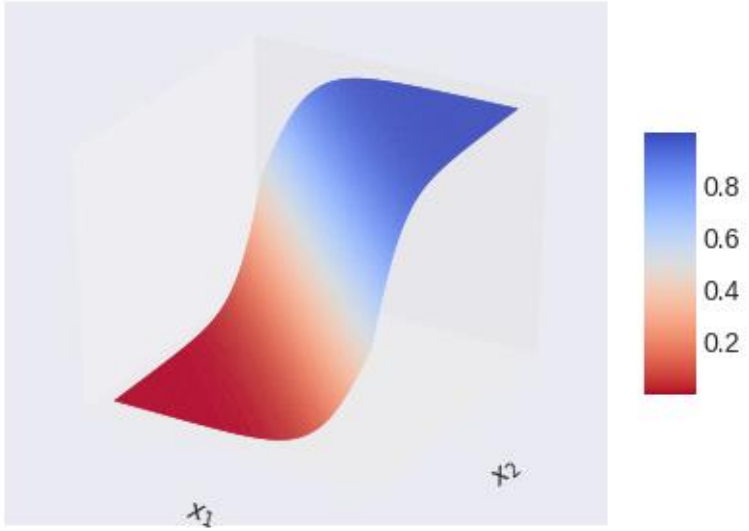
$$a(x) = b + w_1 x_1 + \ldots + w_n x_n = \sum_{t=0}^{n} w_t x_t$$

$$h(x) = \sigma(a(x))$$

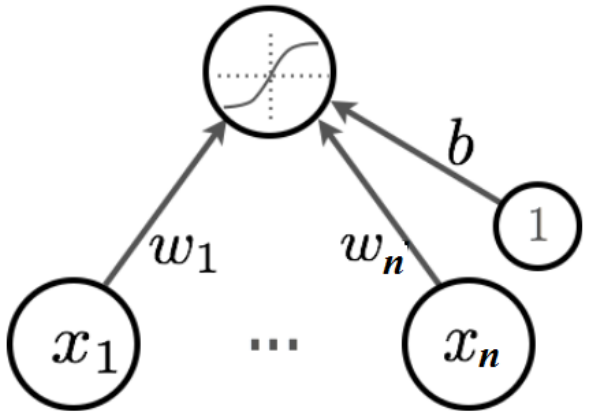
b – **смещение**

σ – **функция активации**

w_t – **веса связей**



Линейные модели – нейросети!



Линейная регрессия

$$a(x) = b + w_1 x_1 + \ldots + w_n x_n$$

Логистическая регрессия

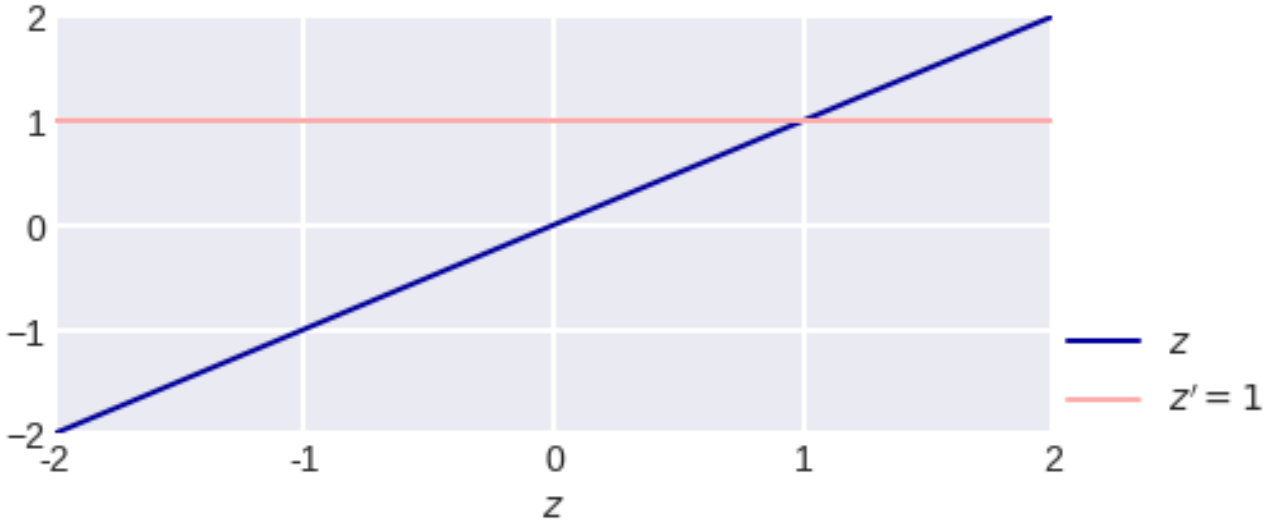
$$a(x) = \sigma(b + w_1 x_1 + \ldots + w_n x_n)$$

Линейный классификатор

$$a(x) = \text{th}(b + w_1 x_1 + \ldots + w_n x_n)$$

Функции активации

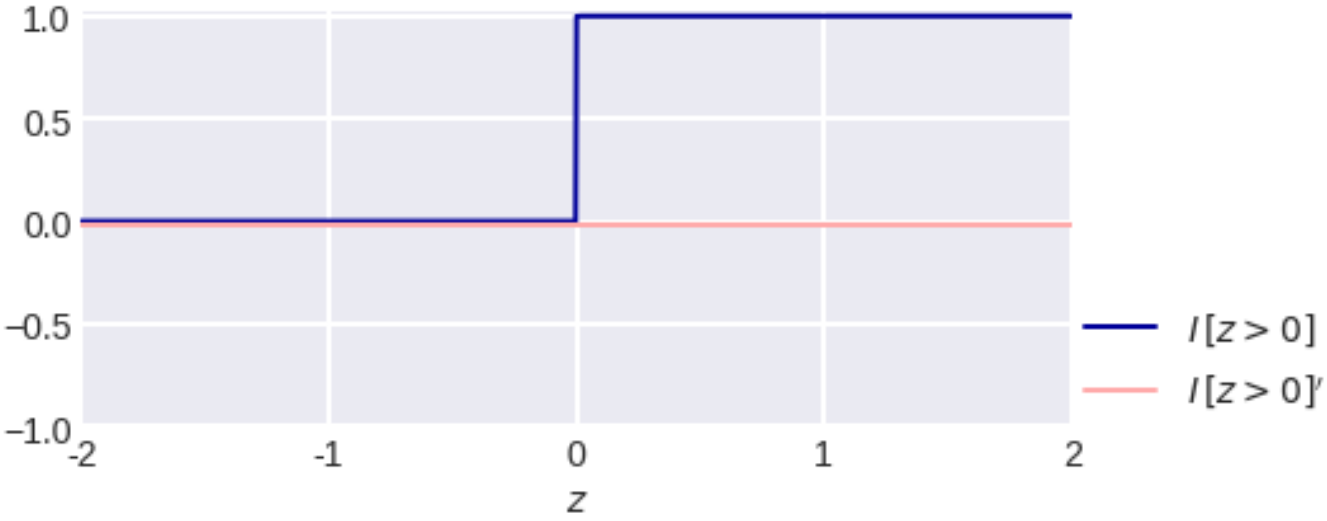
Тождественная функция (линейная / linear activation function)



$$f(z) = z$$

$$\frac{\partial f(z)}{\partial z} = 1$$

Пороговая функция (threshold function)

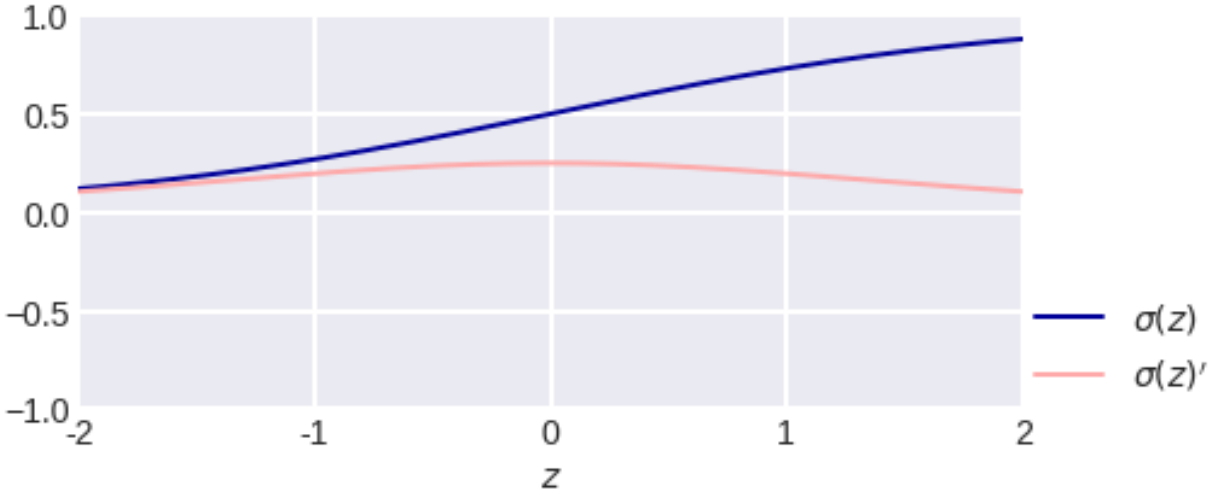


$$\text{th}(z) = I[z > 0]$$

$$\frac{\partial \text{th}(z)}{\partial z} = 0$$

Функции активации

Сигмоида (sigmoid activation function)

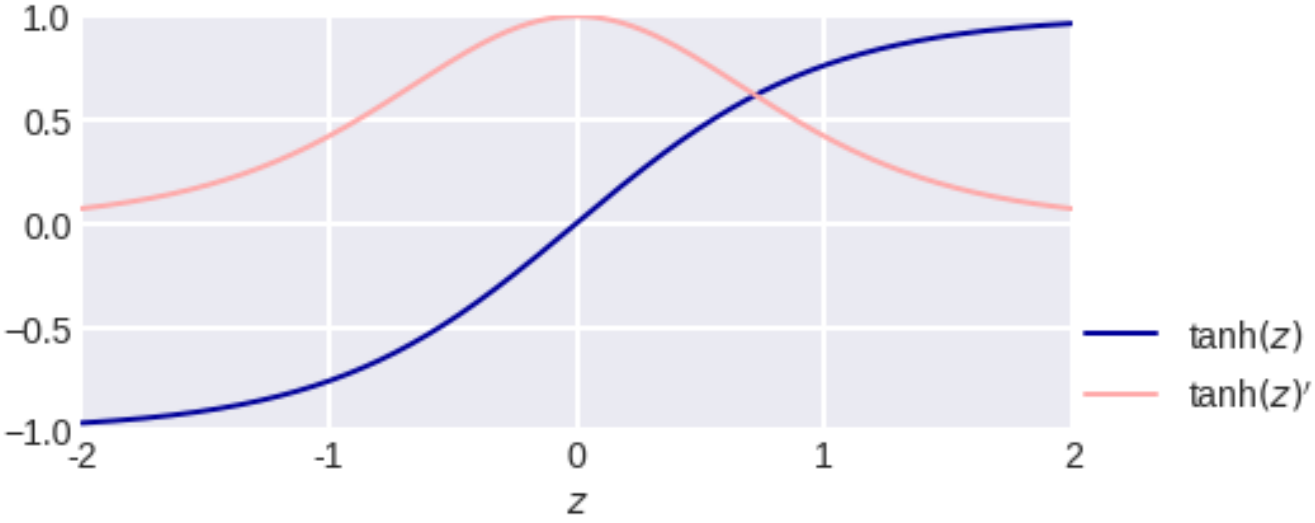


$$\sigma(z) = \frac{1}{1+e^{-z}} \in (0,1)$$

$$\frac{\partial \sigma(z)}{\partial z} =$$

$$= \sigma(z)(1-\sigma(z)) > 0$$

Гиперболический тангенс (hyperbolic tangent)



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

$$= \frac{e^{+z} - e^{-z}}{e^{+z} + e^{-z}} = \frac{e^{+2z} - 1}{e^{+2z} + 1}$$

$$\frac{\partial \tanh(z)}{\partial z} = 1 - \tanh^2(z)$$

Функции активации
в задачах классификации

$$\text{softmax}(z_1,\ldots,z_k)=\frac{1}{\sum_{t=1}^{k}\exp(z_t)}(\exp(z_1),\ldots,\exp(z_k))^{\text{T}}$$

сумма выходов = 1

выходы интерпретируются как вероятности

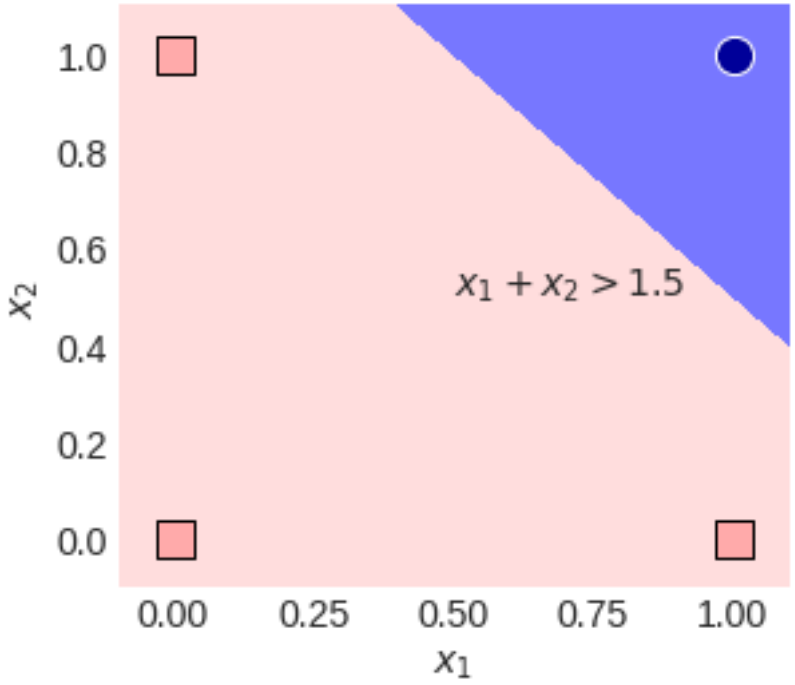
$$[0.5,\ 0.5,\ 0.1,\ 0.7]\rightarrow[0.257,\ 0.257,\ 0.172,\ 0.314]$$

$$[-1.0,\ 0,\ 1.0,\ 0,\ -1.0]\rightarrow[0.07,\ 0.18,\ 0.5,\ 0.18,\ 0.07]$$

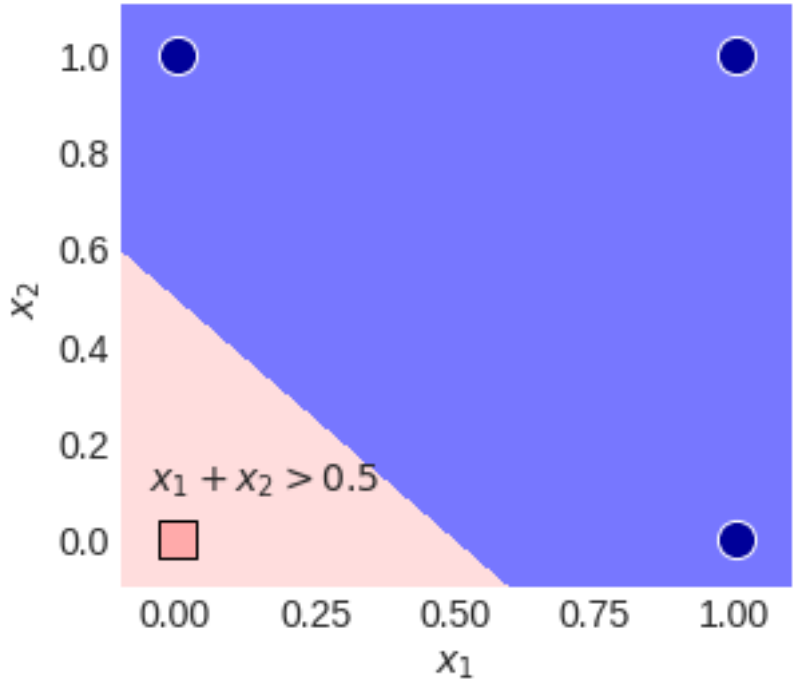
$$[1.0,\ 1.0,\ 1.0,\ 2.0,\ 1.0]\rightarrow[0.15,\ 0.15,\ 0.15,\ 0.4,\ 0.15]$$

Что может один нейрон

Логическое И



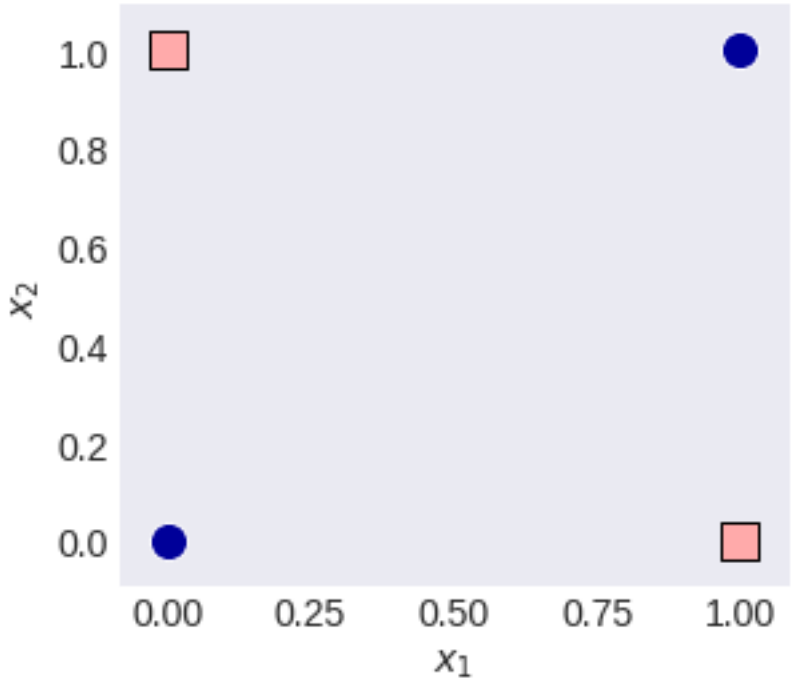
Логическое ИЛИ



Для простоты – пороговая функция активации

Что НЕ может один нейрон

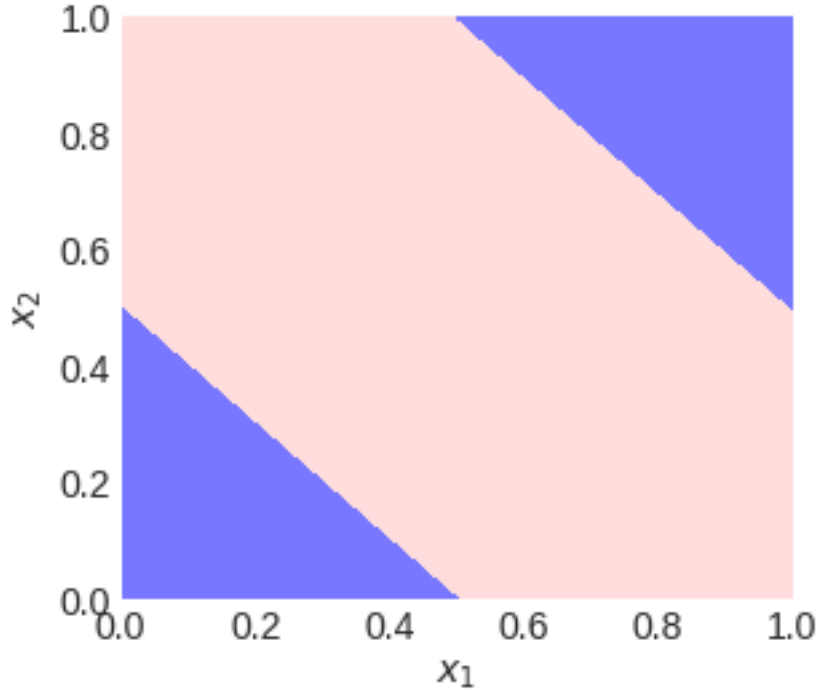
Исключающее ИЛИ

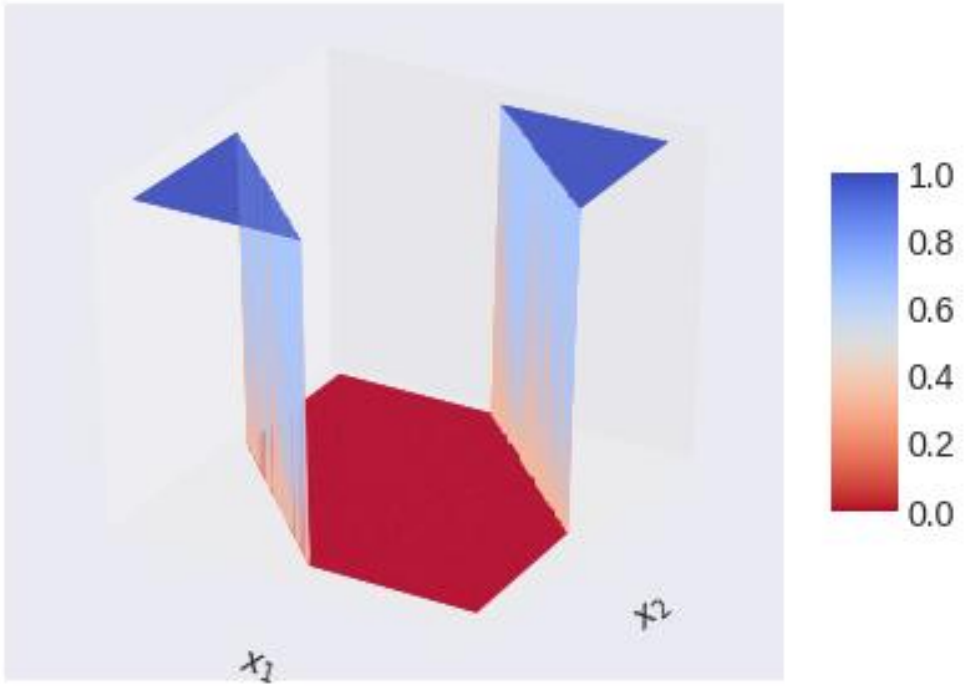


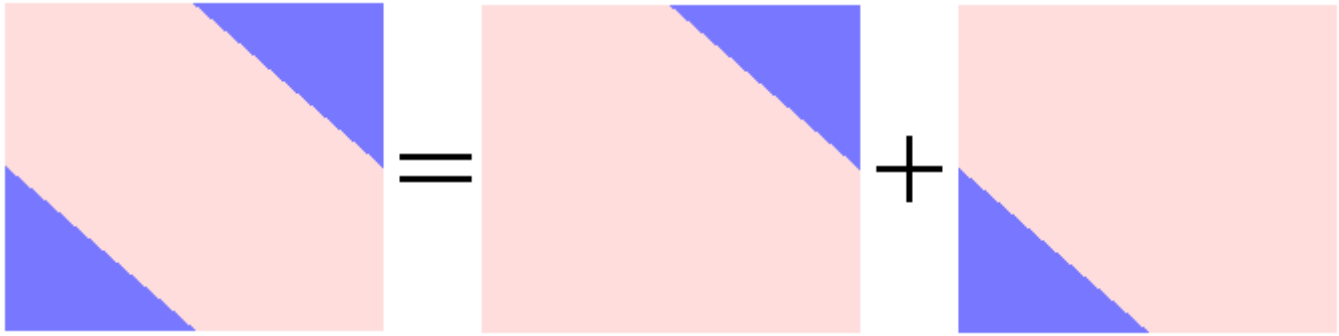
$$\text{th}(\text{th}(x_1 + x_2 - 1.5) + \text{th}(-x_1 - x_2 + 0.5) - 0.5)$$

$$\text{th}(\text{th}(0+0-1.5)+\text{th}(-0-0+0.5)-0.5) = \text{th}(0+1-0.5) = 1$$
$$\text{th}(\text{th}(0+0-1.5)+\text{th}(-0-1+0.5)-0.5) = \text{th}(0+0-0.5) = 0$$
$$\text{th}(\text{th}(0+0-1.5)+\text{th}(-1-0+0.5)-0.5) = \text{th}(0-0-0.5) = 0$$
$$\text{th}(\text{th}(1+1-1.5)+\text{th}(-1-1+0.5)-0.5) = \text{th}(1+0-0.5) = 1$$

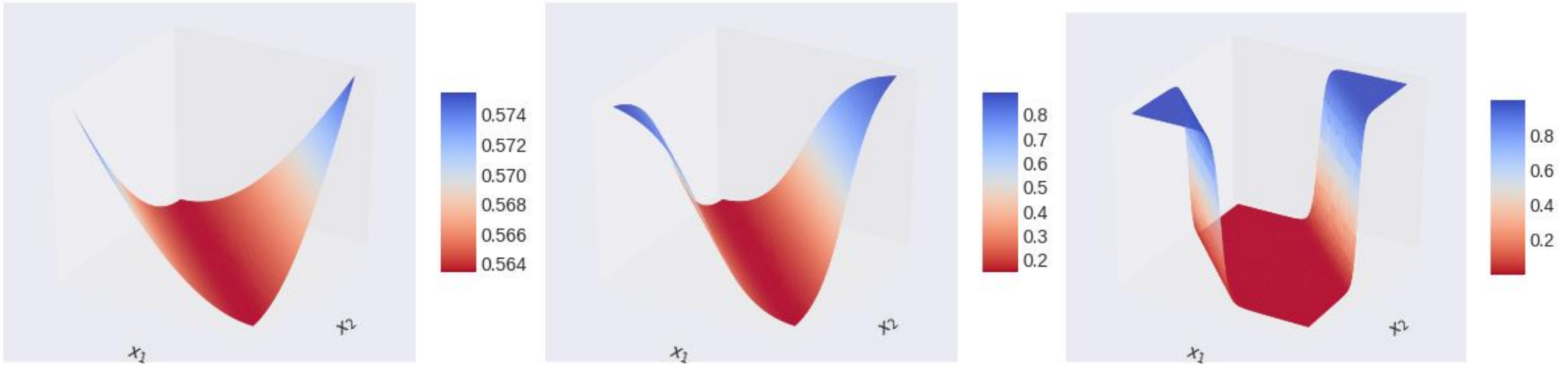
Что НЕ может один нейрон







Сигмоида стремится к пороговой функции



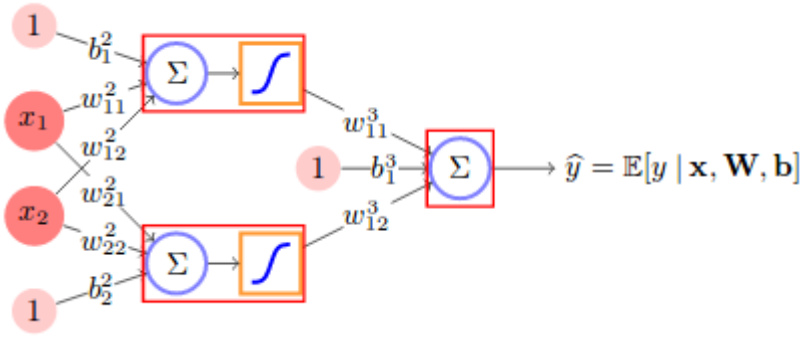
$$\sigma_c(\sigma_c(x_1 + x_2 - 1.5) + \sigma_c(-x_1 - x_2 + 0.5) - 0.5)$$

$$\sigma_c(z) = \frac{1}{1 + e^{-cz}}$$

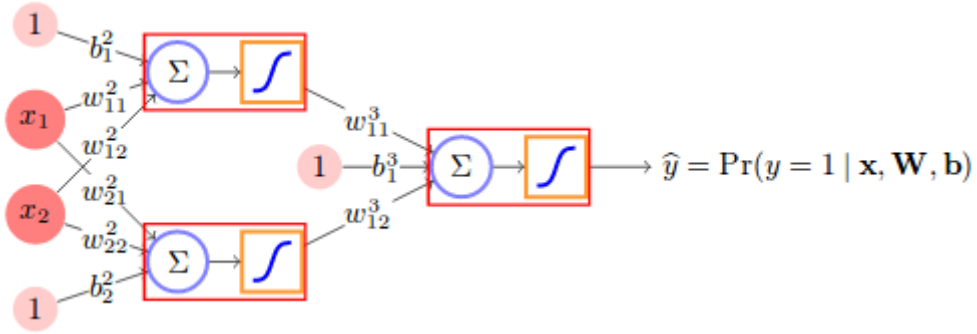
- **Сигмоиду проще обучать – дифференцируемая**
- **Есть возможность получать «вероятности»**

Двуслойная нейронная сеть

Регрессия



Классификация



Такой нейронной сети хватит…

Теорема об универсальной аппроксимации [Hornik, 1991]

Любую непрерывную функцию можно с любой точностью приблизить нейросетью глубины 2 с сигмоидной функцией активации на скрытом слое и линейной функции на выходном слое

Нейросеть глубины два с фиксированной функцией активации в первом слое и линейной функцией активации во втором может равномерно аппроксимировать (м.б. при увеличении числа нейронов в первом слое) любую непрерывную функцию на компактном множестве тогда и только тогда, когда функция активации неполиномиальная.

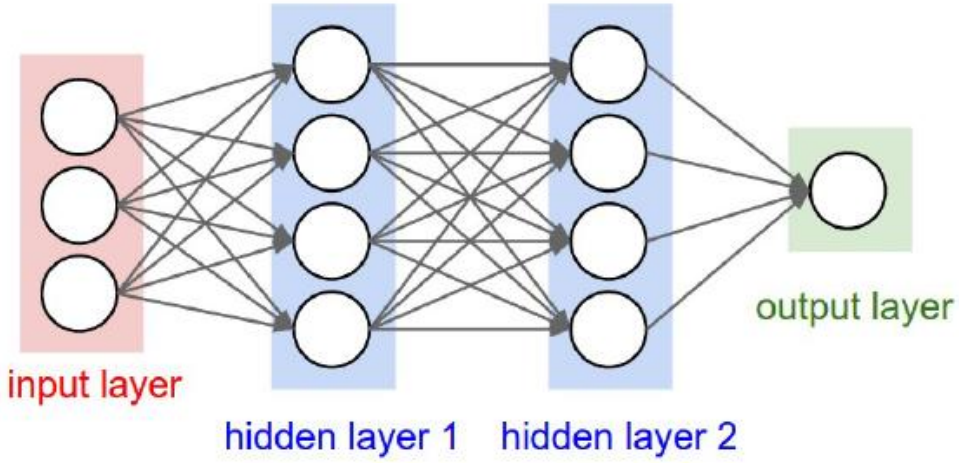
http://www2.math.technion.ac.il/~pinkus/papers/neural.pdf

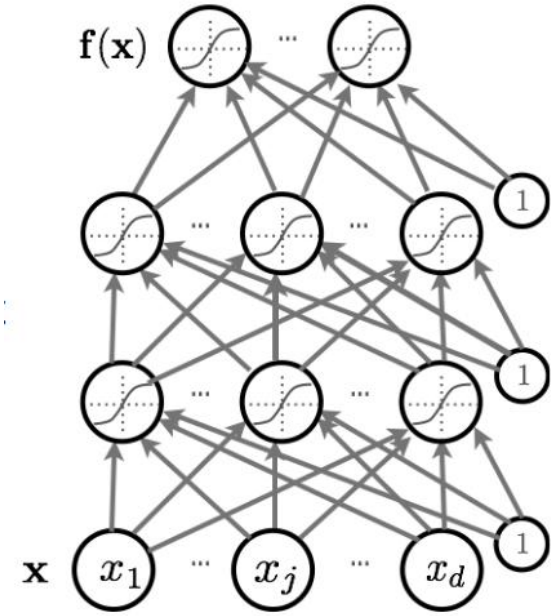
Более того, функция активации м.б. любая (неполиномиальная)!

Но…

- **много нейронов (неизвестно сколько)**
- **экспоненциальные веса**
- **сложность обучения**

Многослойная нейронная сеть – пример нелинейной модели





Ориентированный граф вычислений

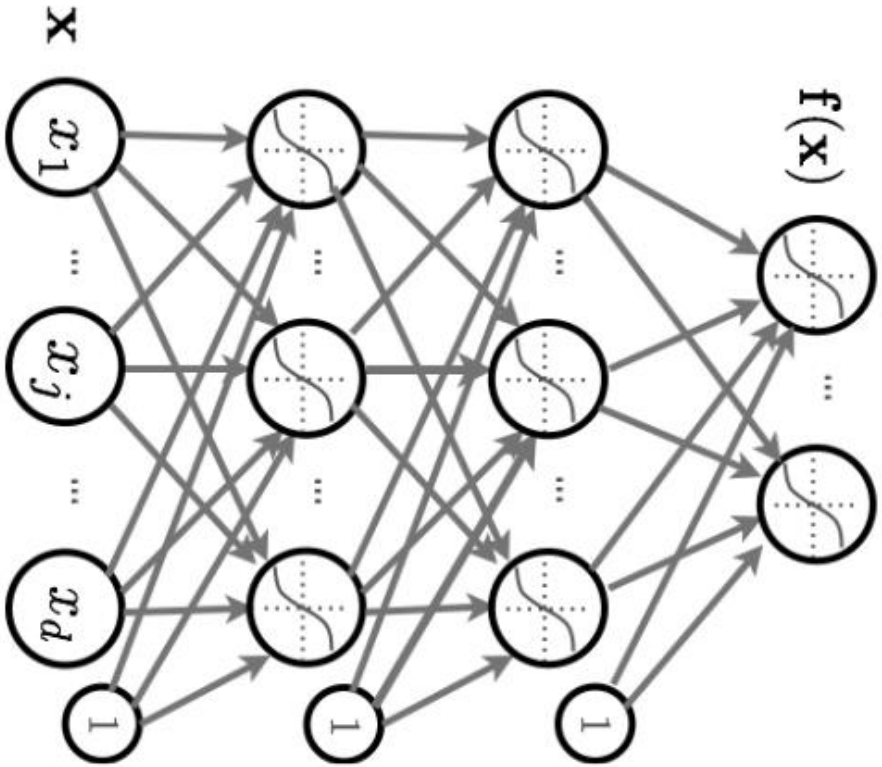
Вершины – переменные или нейроны

Рёбра – зависимости

Сеть прямого распространения – Feedforward Neural Network

(т.е. нет циклов)

все нейроны предыдущего слоя связаны с нейронами следующего



входной слой

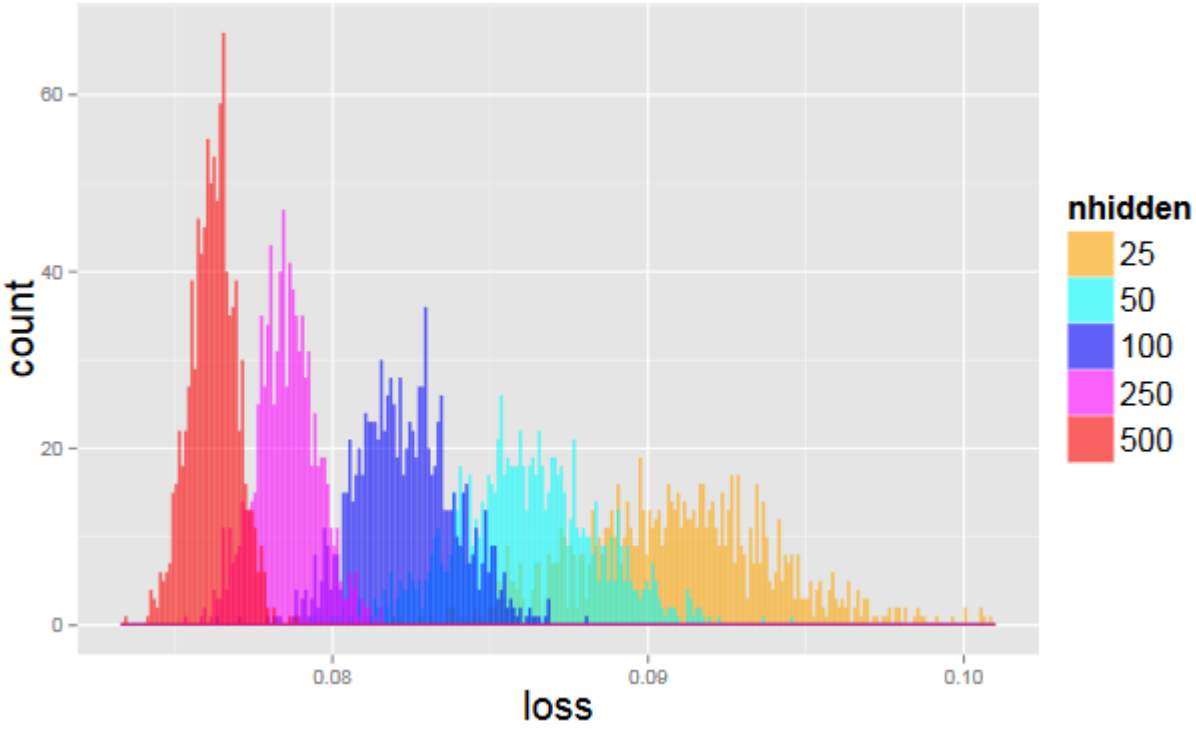
один или несколько скрытых слоёв

выходной слой

Важная аналогия

Глубокая НС – последовательное преобразование признакового пространства

Зачем нужны глубокие нейронные сети:



Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, Yann LeCun «The Loss Surfaces of Multilayer Networks» 2015, https://arxiv.org/abs/1412.0233

Обучение

Как принято…

минимизация регуляризованного эмпирического риска

$$\frac{1}{m}\sum_{t=1}^{m} L(a(x_i \mid w), y_i) + \lambda R(w) \to \min_w$$

Задача оптимизации невыпуклая!

«Настройка» нейронной сети – получение весов w

Метод стохастического градиента

$$w^{(t+1)} = w^{(t)} - \eta \nabla[L(a(x_i \mid w^{(t)}), y_i) + \lambda R(w^{(t)})]$$

т.к. очень много слагаемых… и так быстрее;)

Функции ошибки

Классификация – logloss (CrossEntropyLoss)

$$L((a_1,\ldots,a_l),y) = -\log\frac{\exp(a_y)}{\sum_{j=1}^{l}\exp(a_j)} =$$

$$= -a_y + \log\sum_{j=1}^{l}\exp(a_j)$$

– Часто при реализации делают так:

$$-a_y + \max\{a_j\} + \log\left(\sum_{j=1}^{l}\exp(a_j - \max\{a_j\})\right)$$

Обратное распространение (Backpropagation)

Идея: вычисление производной сложной функции

$$\nabla f(w, g(w), h(w)) = \frac{\partial f}{\partial w} + \frac{\partial f}{\partial g}\nabla g(w) + \frac{\partial f}{\partial h}\nabla h(w)$$

Автоматическое дифференцирование

Прямое распространение

$$x, w \to f(x, w, g(x, w), h(x, w))$$

вычисление ответов, функции ошибки

Обратное распространение

$$x, w, \nabla g, \nabla h \to \nabla f$$

вычисление градиентов

Производные на компьютере

Atilim Gunes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, Jeffrey Mark Siskind «Automatic differentiation in machine learning: a survey» 2015-2018 https://arxiv.org/abs/1502.05767

Функции активации

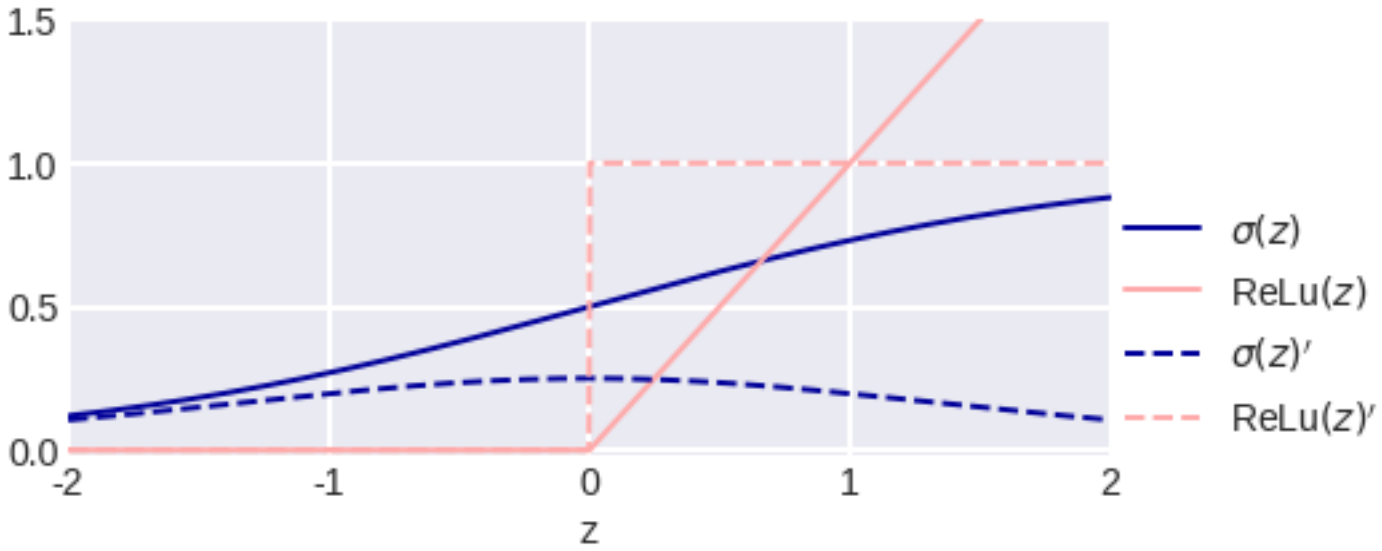
Проблема – затухание градиента

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

$$\frac{\partial \sigma(z)}{\partial z} = \sigma(z)(1-\sigma(z))$$

$$\mathrm{ReLU}(z) = \max(0, z)$$

$$\frac{\partial \mathrm{ReLU}(z)}{\partial z} = I[z > 0]$$



Rectified Linear Unit

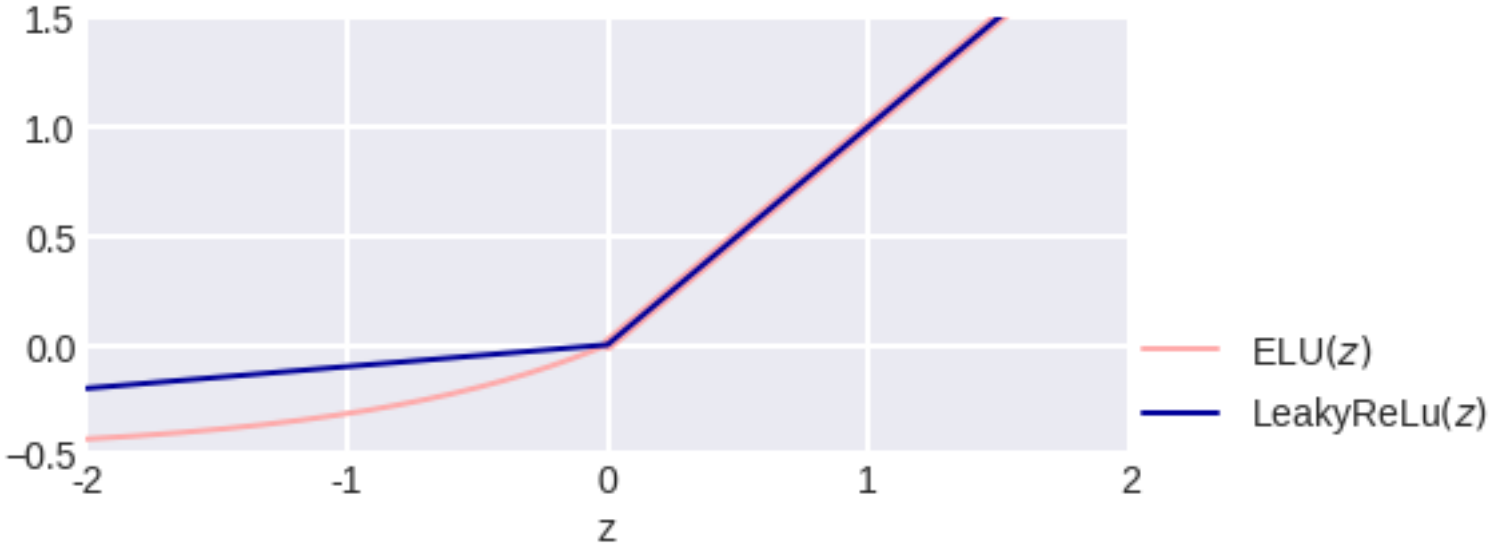
Что плохого в сигмоиде

- «убивает» градиенты
- выходы не отцентрированы (легко устранить → tanh)
- вычисление экспоненты всё-таки дорого…

Ещё функции активации

$$\text{LeakyReLU}(z) = \max(0.1z, z)$$

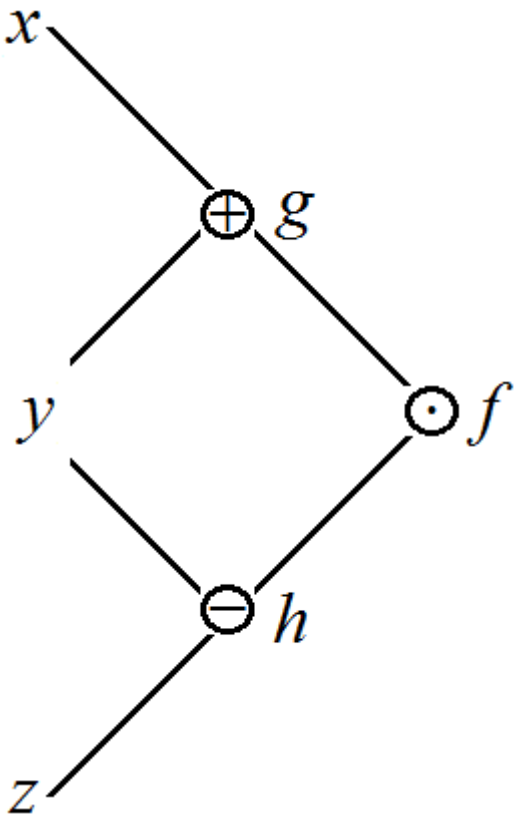
$$\text{ELU}(z) = \begin{cases} z, & z \geq 0, \\ \alpha(e^z - 1), & z < 0. \end{cases}$$



$$\text{Maxout}(z) = \max(w^{\text{T}} z + w_0, v^{\text{T}} z + v_0)$$

Вычисление градиента

$$f = (x+y)\cdot(y-z)$$



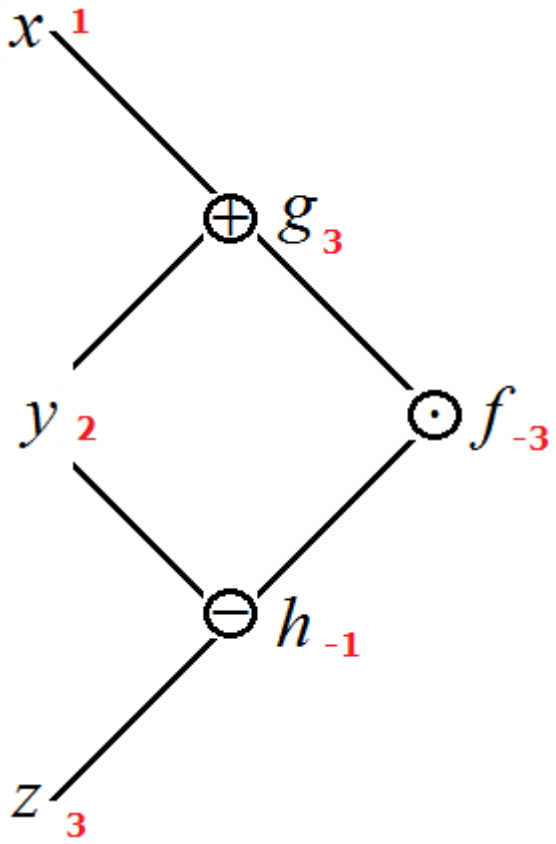
$$f(x,y,z) = \underbrace{(x+y)}_{g(x,y)} \cdot \underbrace{(y-z)}_{h(y,z)}$$

Как проводится вычисление функции?

$$x, y, z = 1, 2, 3$$

Вычисление градиента

$$f = (x+y)\cdot(y-z)$$



$$f(x,y,z) = \underbrace{(x+y)}_{g(x,y)} \cdot \underbrace{(y-z)}_{h(y,z)}$$

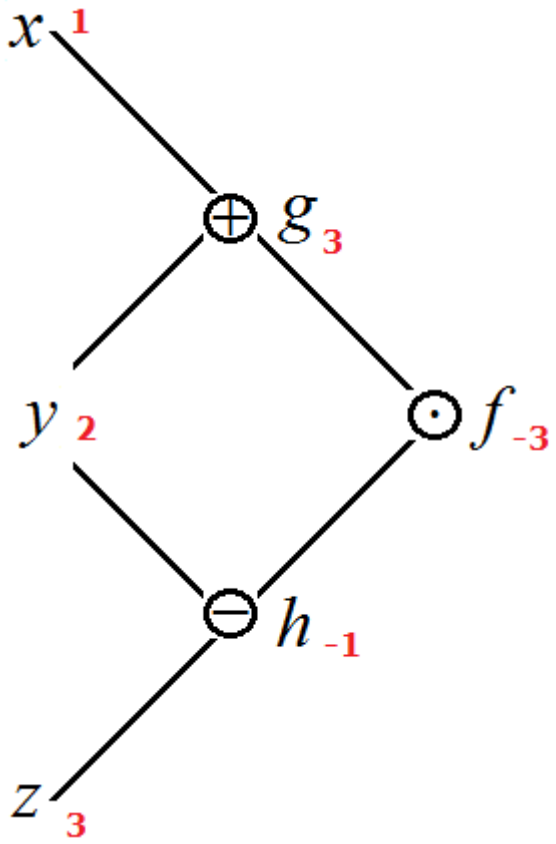
Как проводится вычисление функции?

$$x, y, z = 1, 2, 3$$

«Прямой ход»

Вычисление градиента

$$f = (x+y)\cdot(y-z)$$



$$f(x,y,z) = \underbrace{(x+y)}_{g(x,y)} \cdot \underbrace{(y-z)}_{h(y,z)}$$

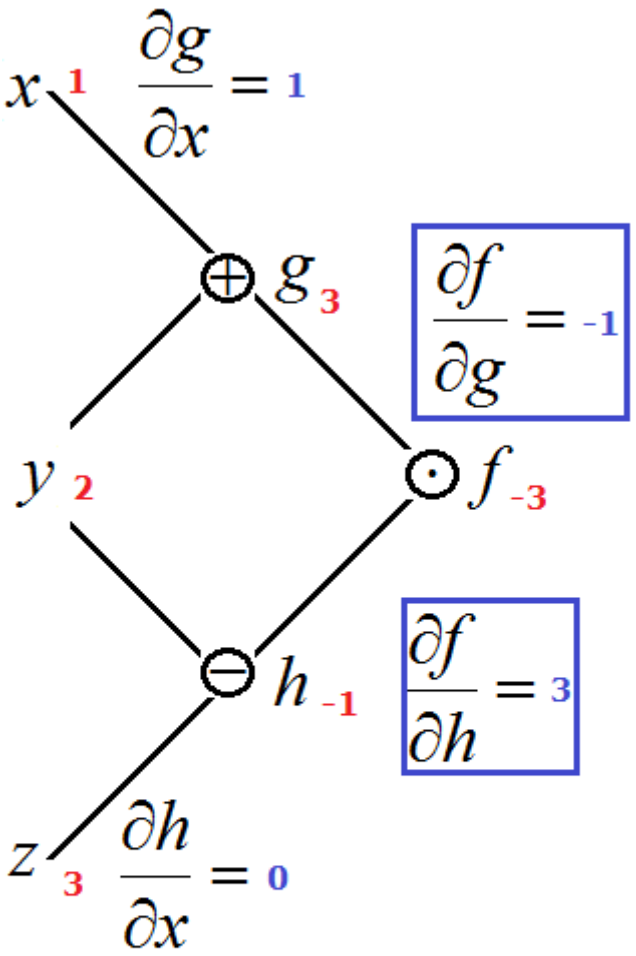
Как проводится вычисление производных?

$$\frac{\partial f}{\partial g} = h, \frac{\partial f}{\partial h} = g$$

$$\frac{\partial g}{\partial x} = 1, \frac{\partial h}{\partial x} = 0$$

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial x} + \frac{\partial f}{\partial h}\frac{\partial h}{\partial x}$$

Вычисление градиента



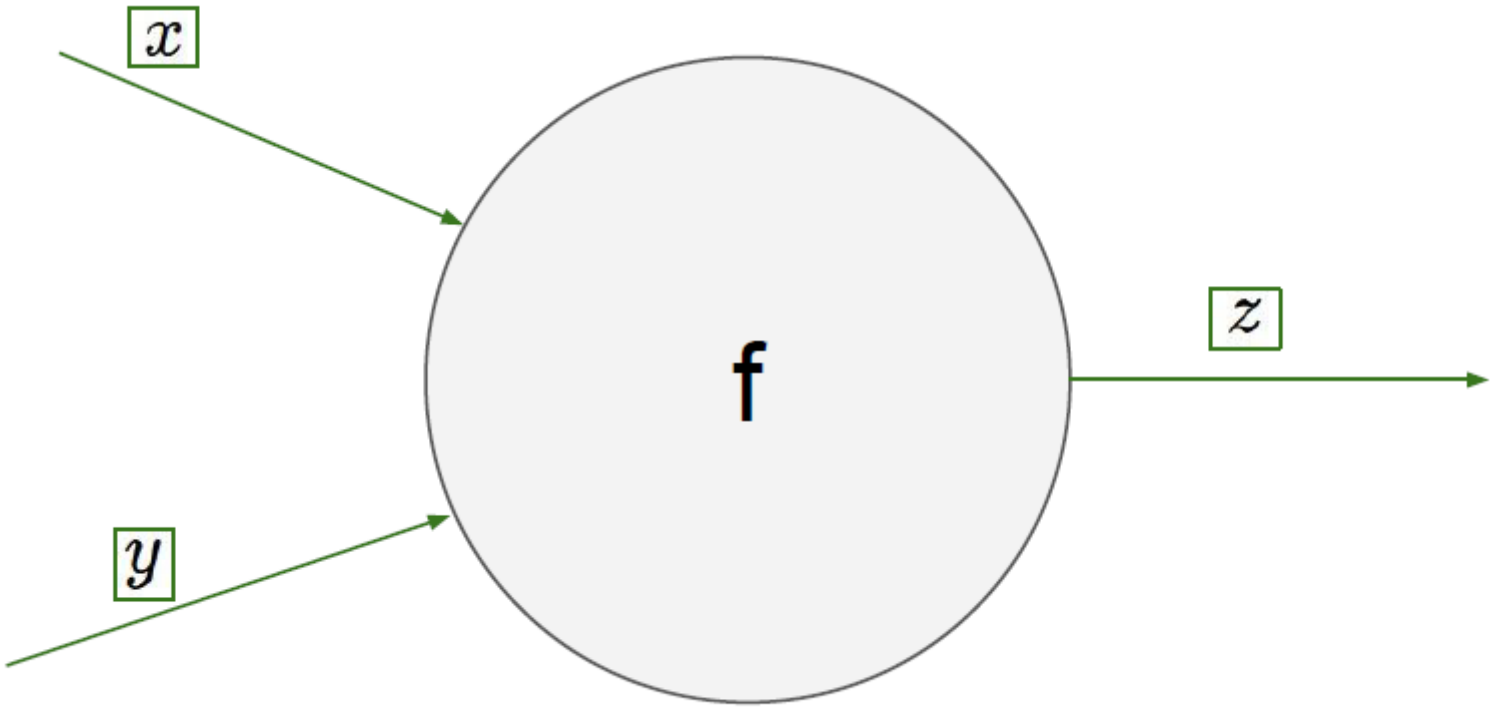
$$\frac{\partial f}{\partial x} = \boxed{\frac{\partial f}{\partial g}}\frac{\partial g}{\partial x} + \boxed{\frac{\partial f}{\partial h}}\frac{\partial h}{\partial x}$$

$$f(x,y,z) = \underbrace{(x+y)}_{g(x,y)} \cdot \underbrace{(y-z)}_{h(y,z)}$$

Как проводится вычисление производных?

«Обратный ход»

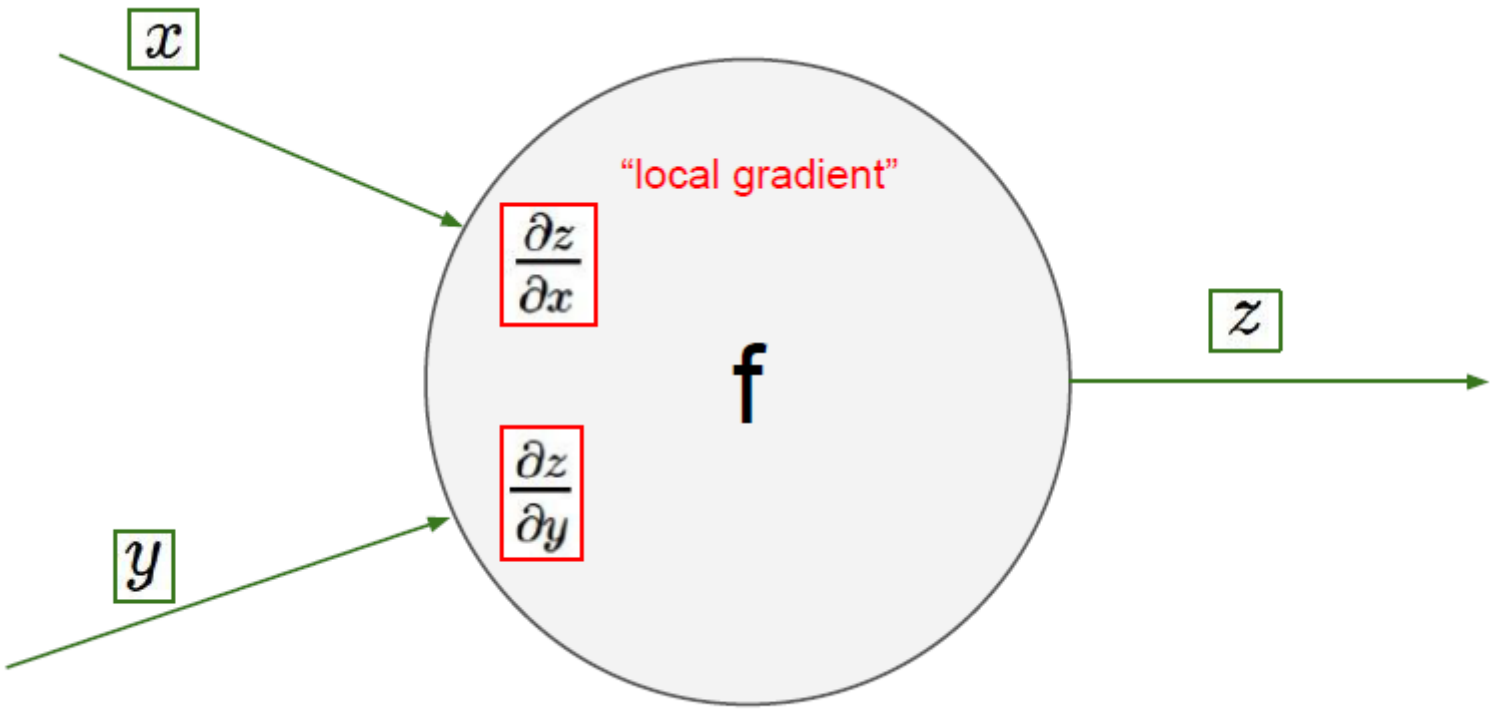
Обратное распространение градиента



http://cs231n.stanford.edu/2017/index.html

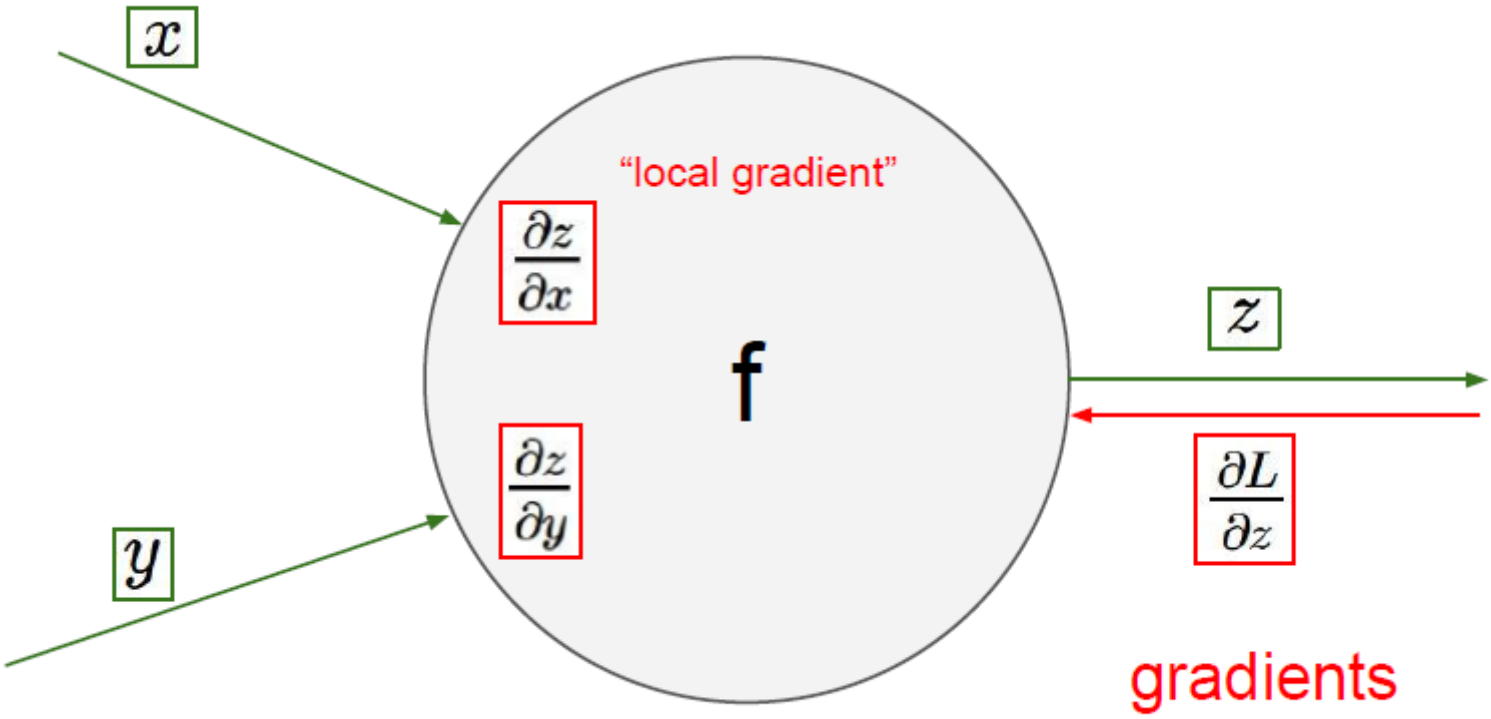
Обратное распространение = SGD + дифференцирование сложных функций

Обратное распространение градиента



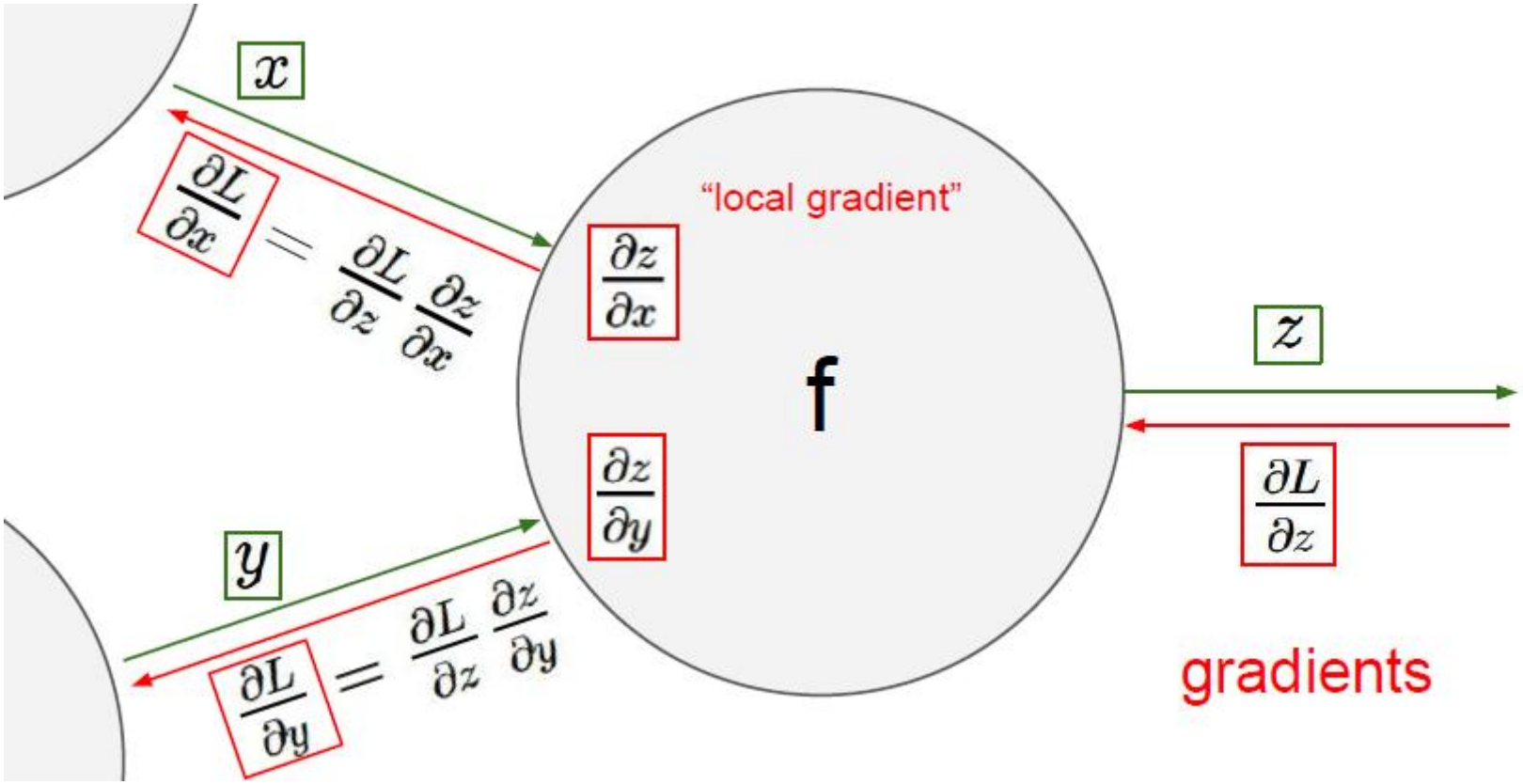
http://cs231n.stanford.edu/2017/index.html

Обратное распространение градиента



http://cs231n.stanford.edu/2017/index.html

Обратное распространение градиента



http://cs231n.stanford.edu/2017/index.html

Борьба с переобучением

Очень много параметров $\Rightarrow$ переобучение

- **Нормировки (Normalization of Data)**
- **Инициализация весов**
- **Верификация – ранний останов (Early Stopping)**
- **Настройка темпа обучения (Learning Rate)**
- **Мини-батчи (Mini-Batches) / Batch-обучение**
- **Продвинутая оптимизация**
- **Регуляризация + Weight Decay**
- **Max-norm-регуляризация**
- **Dropout**
- **Увеличение выборки + Расширение выборки (Data Augmentation)**
- **Обрезка градиентов (Gradient clipping)**

Борьба с переобучением

- **Доучивание уже настроенных нейросетей (Pre-training)**
- **Unsupervised Learning**
- **Техника зануления весов (разреживания НС)**
- **Использование специальных архитектур под задачу**
  (например, использующих локальность – свёрточных НС)
- **зашумление (inject noise)**

Нормировки (Normalization of Data)

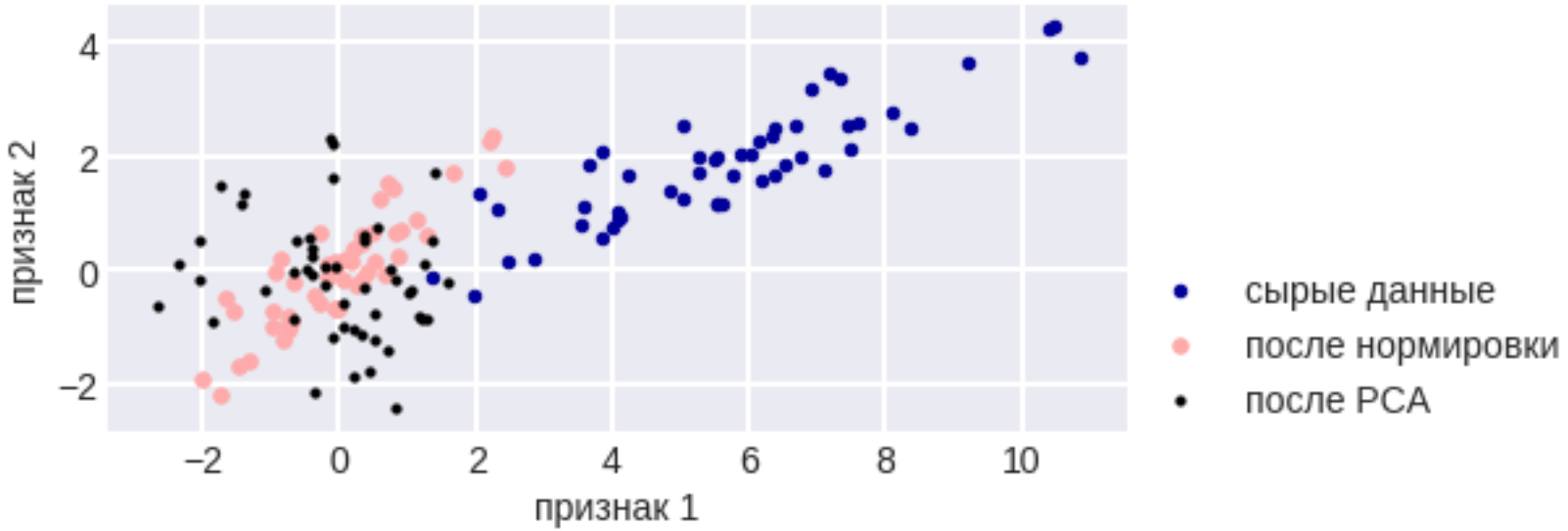
Признак $X = (X_1, \ldots, X_m)$

$$\mu = \frac{1}{m}\sum_{i=1}^{m} X_i$$ **среднее признака**

$$\sigma^2 = \frac{1}{m}\sum_{i=1}^{m} (X_i - \mu)^2$$ **дисперсия признака**

иногда м.б. PCA

$$X = \frac{X - \mu}{\sqrt{\sigma^2}}$$ **нормировка**



Нормировки (Normalization of Data)

Huang et al, «Decorrelated Batch Normalization», arXiv 2018 (Appeared 4/23/2018)

Инициализация весов

- **нарушение симметричности**
  (чтобы нейроны были разные)
- **недопустить «насыщенности» нейрона**
  (почти всегда близок к нулю или 1)

так, чтобы признаки, поступающие в слой имели одинаковую дисперсию – для избегания «насыщения» нейронов

смещения := 0 (зависит от того, где смещение; если на выходе…)

Xavier initialization

$$w_{ij}^{(k)} \sim U\left[-\sqrt{\frac{6}{n_{\text{in}}^{(k)}+n_{\text{out}}^{(k)}}},+\sqrt{\frac{6}{n_{\text{in}}^{(k)}+n_{\text{out}}^{(k)}}}\right]$$

[Glorot & Bengio, 2010]

Распределение, чтобы $Dw_{ij}^{(k)} = \frac{2}{n_{\text{in}}^{(k)}+n_{\text{out}}^{(k)}}$

Формула выведена в предположении, что нет нелинейностей…

$$z^{(k+1)} = f(W^{(k)}z^{(k)}) \equiv W^{(k)}z^{(k)}$$

можно посчитать см.

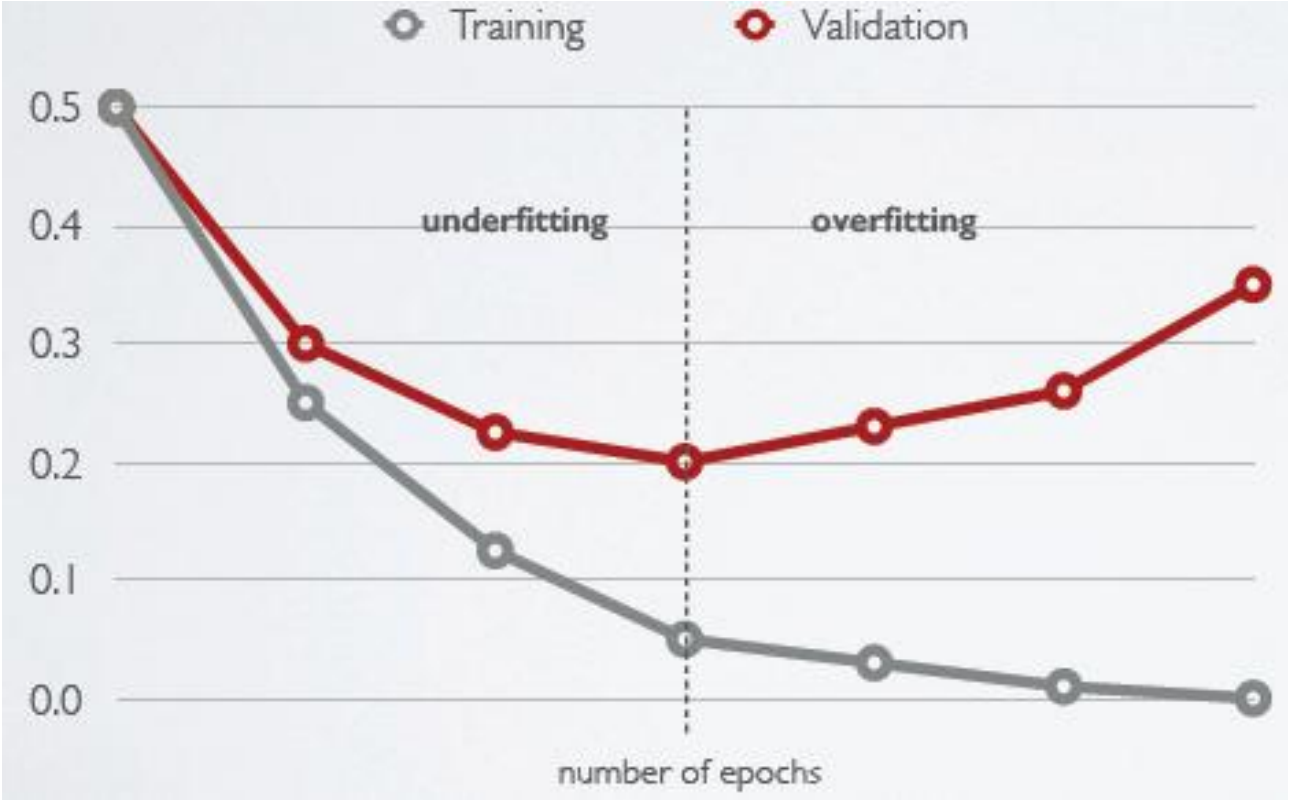
https://www.youtube.com/watch?v=PjS2y8LBMLc&list=PLrCZzMib1e9oOGNLh6_d65HyfdqlJwTQP&index=5

Плохо для ReLu – [He et al., 2015]

Верификация – ранний останов (Early Stopping)

Смотрим ошибку на отложенной выборке!

Выбираем итерацию, на которой наименьшая ошибка.



Настройка темпа обучения (Learning Rate)

Мини-батчи (mini-batches) / Batch-обучение

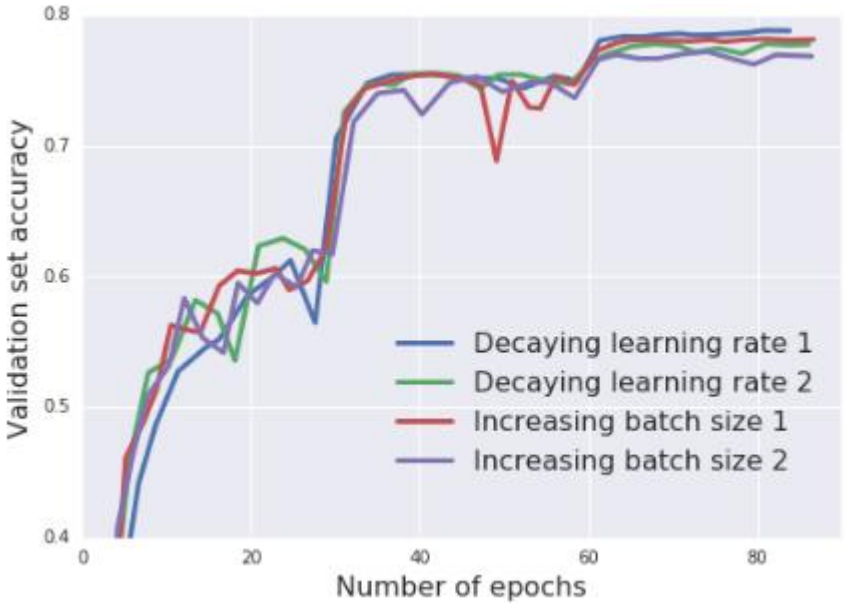
$$w^{(t+1)} = w^{(t)} - \frac{\eta}{|I|}\sum_{i\in I} \nabla[l(a(x_i \mid w^{(t)}), y_i) + \lambda R(w^{(t)})]$$

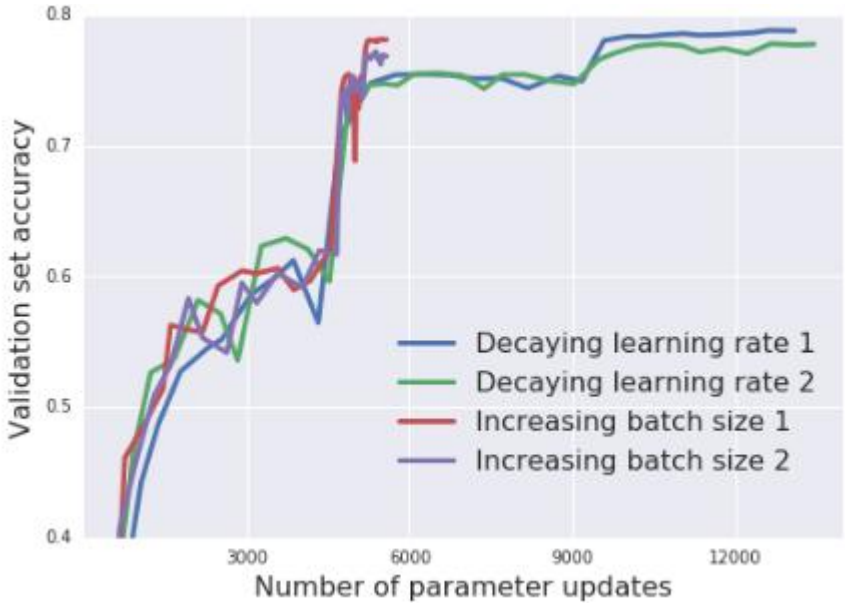
- **Градиенты не такие случайные (оцениваем по подвыборке)**
- **Можно вычислять быстрее (на современных архитектурах)**
  Можно делать максимальный батч, который влезает в память
- **Можно делать нормировку по батчу**
- **Немного противоречит теории**

м.б. специально организовывать батчи

(должны содержать представителей всех классов)

Мини-батчи (mini-batches) / Batch-обучение





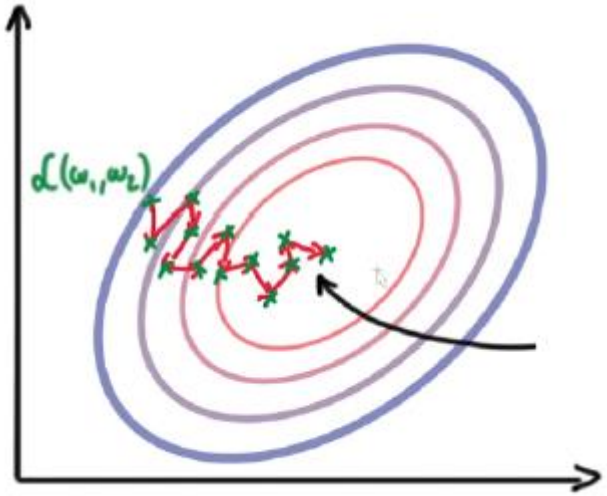
увеличение размера батча – тот же эффект,
что и уменьшение темпа обучения

Samuel L. Smith, Pieter-Jan Kindermans, Chris Ying, Quoc V. Le «Don't Decay the Learning Rate, Increase the Batch Size» https://arxiv.org/abs/1711.00489

Продвинутая оптимизация

Обучение: стохастический градиент

$$w^{(t+1)} = w^{(t)} - \eta \nabla L^{(t)}(w^{(t)})$$



Эпоха – проход по всей обучающей выборке

Надо случайно перемешивать данные перед каждой эпохой

Продвинутая оптимизация

стохастический градиент с моментом (momentum)

$$m^{(t+1)} = \rho m^{(t)} + \nabla L^{(t)}(w^{(t)})$$

$$w^{(t+1)} = w^{(t)} - \eta m^{(t+1)}$$

добавление инерции

метод Нестерова

$$m^{(t+1)} = \rho m^{(t)} + \nabla L^{(t)}(w^{(t)} - \eta m^{(t)})$$

$$w^{(t+1)} = w^{(t)} - \eta m^{(t+1)}$$

- **градиент случаен (зависит от батча)**

добавляем градиенту инертность

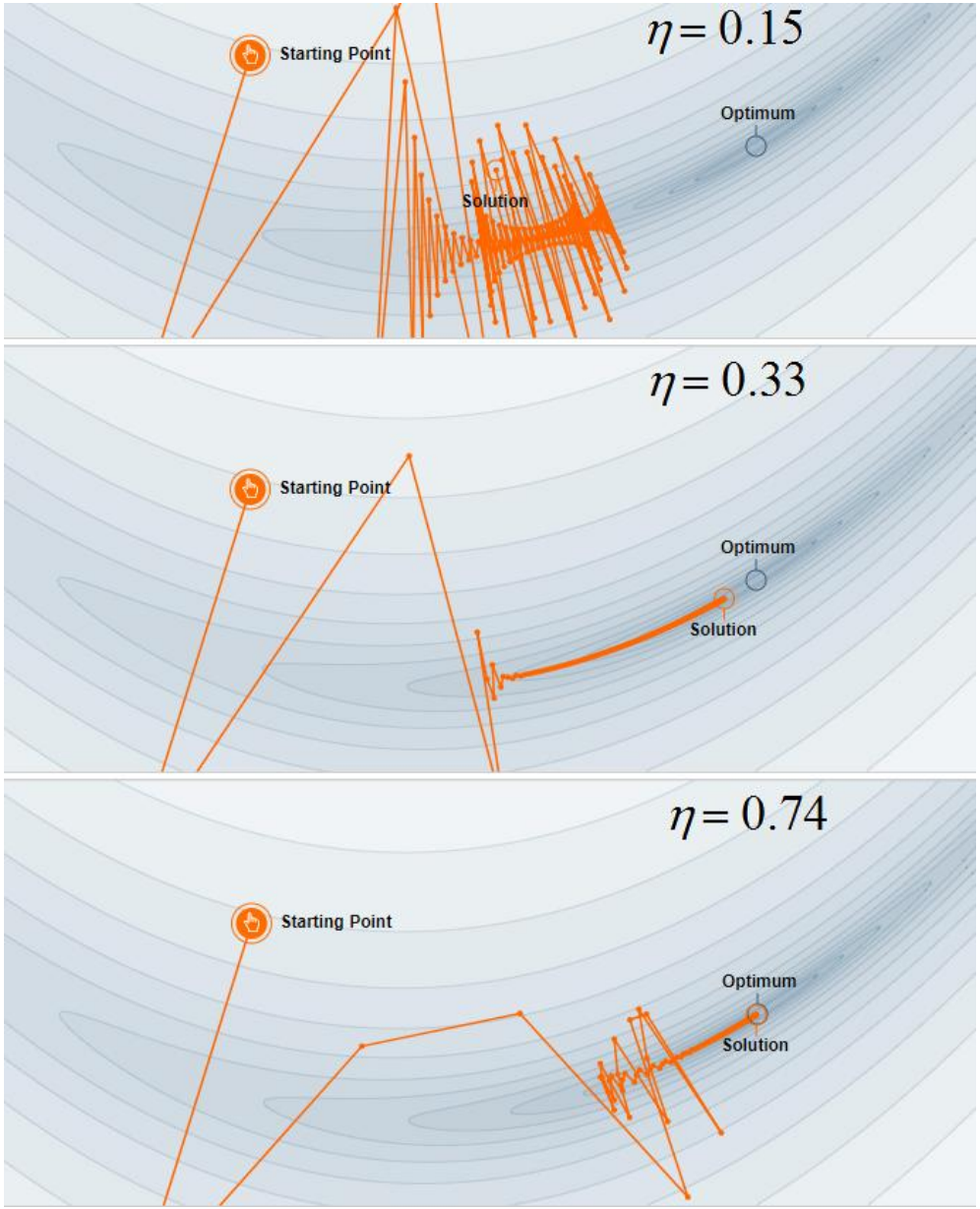
- **уровни функции могут быть сильно вытянуты**

adam

- **все параметры разные**

скорость обучения – для каждого параметра

Иллюстрация СГ с моментом



Добавление инерции может помочь, когда

- **линии уровня вытянуты…**
- **для проскакивания седловых точек**

https://distill.pub/2017/momentum/

Продвинутая оптимизация – Адаптивная

Adagrad [Duchi и др., 2011]

$$v_i^{(t+1)} = v_i^{(t)} + (\nabla_i L^{(t)}(w^{(t)}))^2$$

$$w_i^{(t+1)} = w_i^{(t)} - \frac{\eta}{\sqrt{v_i^{(t+1)} + \varepsilon}} \nabla_i L^{(t)}(w^{(t)})$$

RMSprop = Adagrad + MA [Hinton, 2012]

$$v_i^{(t+1)} = \beta v_i^{(t)} + (1-\beta)(\nabla_i L^{(t)}(w^{(t)}))^2$$

$$w_i^{(t+1)} = w_i^{(t)} - \frac{\eta}{\sqrt{v_i^{(t+1)} + \varepsilon}} \nabla_i L^{(t)}(w^{(t)})$$

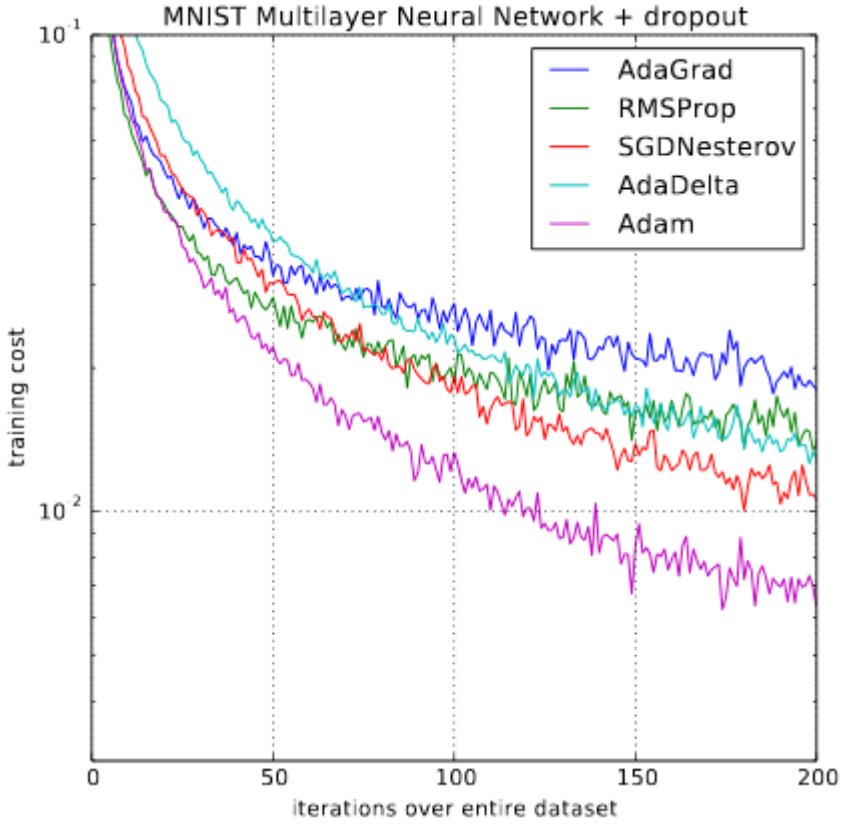
Продвинутая оптимизация – Адаптивная

Adam = RMSprop + momentum

$$m_i^{(t+1)} = \alpha m_i^{(t)} + (1-\alpha)\nabla_i L^{(t)}(w^{(t)})$$

$$v_i^{(t+1)} = \beta v_i^{(t)} + (1-\beta)(\nabla_i L^{(t)}(w^{(t)}))^2$$

$$w_i^{(t+1)} = w_i^{(t)} - \frac{\eta}{\sqrt{v_i^{(t+1)} + \varepsilon}} m_i^{(t)}$$



Adam = «Adaptive Moment Estimation»

18k ссылок, неверное доказательство сходимости

[Kingma, Ba, 2014 https://arxiv.org/abs/1412.6980]

Продвинутая оптимизация

AdaDelta (усл. Adagrad)

$$v_i^{(t+1)} = v_i^{(t)} + (\nabla_i L^{(t)}(w^{(t)}))^2$$

$$\Delta w_i^{(t+1)} = -\frac{\sqrt{\eta_i^{(t)} + \varepsilon}}{\sqrt{v_i^{(t+1)} + \varepsilon}} \nabla_i L^{(t)}(w^{(t)})$$

$$w_i^{(t+1)} = w_i^{(t)} + \Delta w_i^{(t+1)}$$

$$\eta_i^{(t+1)} = \gamma \eta_i^{(t)} + (1-\gamma)(\Delta w_i^{(t+1)})^2$$

Подбор гиперпараметров очень важен!

Подбор метода оптимизации очень важен!

Регуляризация + Weight Decay

Не применяется к весам к константным входам (смещениям)

L2, L1 – регуляризация

Уменьшение весов (вид регуляризации)

$$w^{(t+1)} = (1-\lambda)w^{(t)} - \eta \nabla L^{(t)}(w^{(t)})$$

На самом деле это L2-регуляризация:

$$\nabla(L(w) + \lambda \| w \|^2) = \nabla L(w) + \lambda w$$

$$w^{(t+1)} = w^{(t)} - \eta(\nabla L(w^{(t)}) + \lambda w^{(t)}) = (1-\lambda\eta)w^{(t)} - \eta \nabla L(w^{(t)})$$

```
from keras.regularizers import l2 # L2-regularisation
l2_lambda = 0.0001
conv_1 = Convolution2D(conv_depth, kernel_size, kernel_size,
                       border_mode='same', W_regularizer=l2(l2_lambda),
                       activation='relu')(inp)
```

Max-norm-регуляризация

Для каждого нейрона ограничиваем норму весов:

$$\| w_{\text{neuron}} \| \le c$$

если превысила – проекция

Results from MNIST (handwritten digit recognition)

Method	Unit Type	Architecture	Error %
Standard Neural Net (Simard et al., 2003)	Logistic	2 layers, 800 units	1.60
SVM Gaussian kernel	NA	NA	1.40
Dropout NN	Logistic	3 layers, 1024 units	1.35
Dropout NN	ReLU	3 layers, 1024 units	1.25
Dropout NN + max-norm constraint	ReLU	3 layers, 1024 units	1.06
Dropout NN + max-norm constraint	ReLU	3 layers, 2048 units	1.04
Dropout NN + max-norm constraint	ReLU	2 layers, 4096 units	1.01
Dropout NN + max-norm constraint	ReLU	2 layers, 8192 units	0.95
Dropout NN + max-norm constraint (Goodfellow et al., 2013)	Maxout	2 layers, (5 × 240) units	0.94

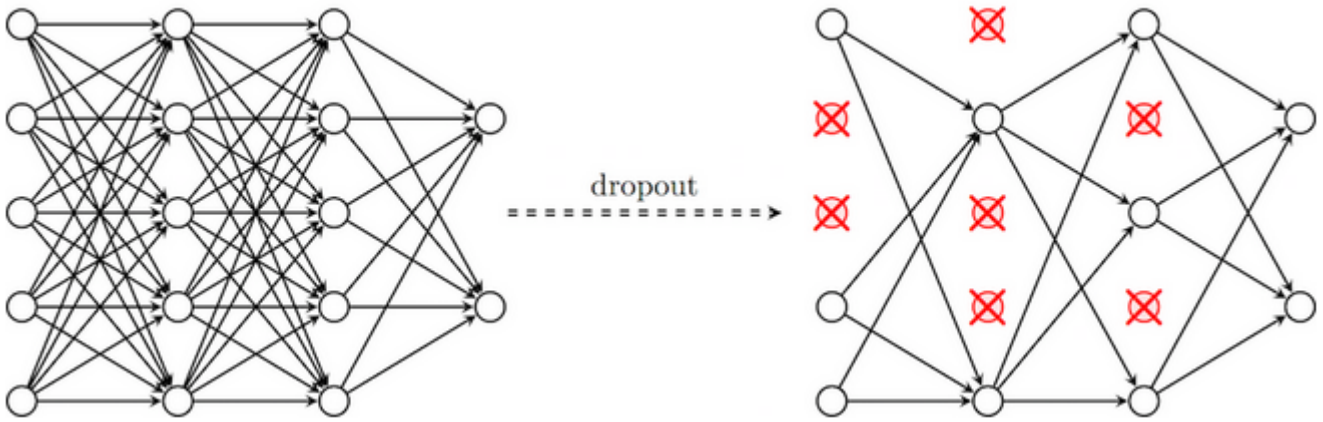
http://www.cs.toronto.edu/~rsalakhu/papers/srivastava14a.pdf

Оптимизаторы

«An overview of gradient descent optimization algorithms»

http://ruder.io/optimizing-gradient-descent/

Dropout



случайное обнуление активаций

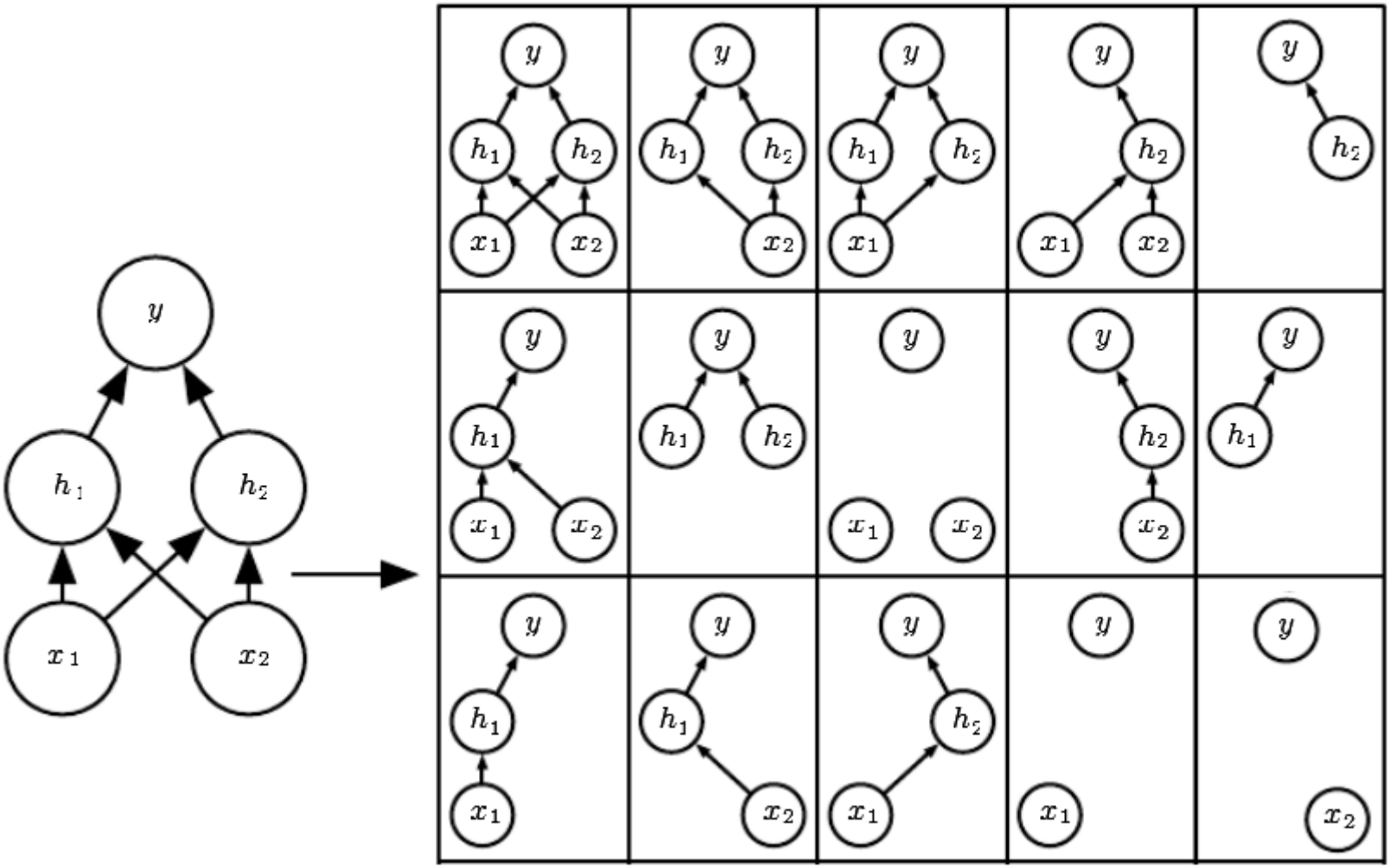
~ выбрасывание нейронов из сети

не выбрасываем из последнего слоя

Обычно в полносвязных слоях!

Отключают в режиме теста – выход умножается на вероятность выбрасывания.

Dropout



В отличие от бэгинга все модели делят параметры (не независимы)

[DLbook]

Inverted Dropout

Во время обучения выход нейрона умножается на

$$\frac{1}{1-p}$$

p – вероятность выбрасывания

Во время тестирования ничего не делаем…

DropConnect

Зануление отдельных весов, а не нейронов

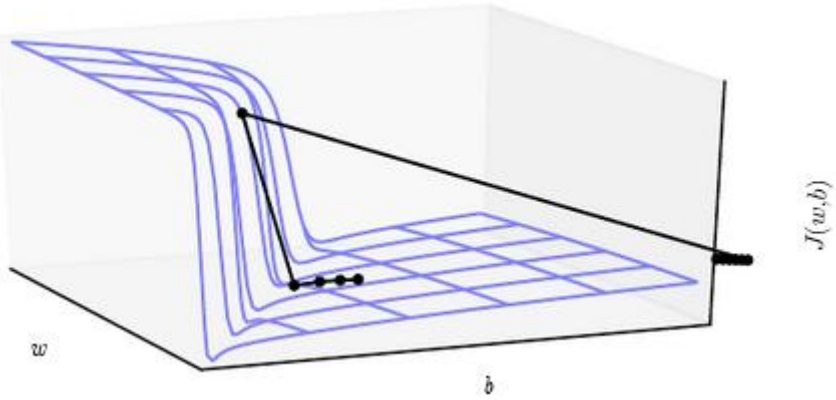
https://cs.nyu.edu/~wanli/dropc/

Обрезка градиентов (Gradient clipping)

$$g = \frac{\partial L}{\partial w}$$

$$g^{\text{new}} = \frac{\min(\theta, \| g \|)}{\| g \|} g$$

Работает пока дисперсия градиента маленькая…



Обрезка помогает, когда попадаем на утёс…

Pascanu, Mikolov, Bengio для RNN

Батч-нормализация (Batch normalization)

– метод адаптивной перепараметризации

Проблема:
градиент – как изменять параметры,
при условии, что вся остальная сеть не меняется

трудно предсказать, насколько изменится какое-то значение
(оно зависит от всех предыдущих в суперпозиции)

Covariate shift – изменение распределений входов во время обучения
Надо уменьшить это изменение в скрытых слоях!

Ioffe and Szegedy, 2015 https://arxiv.org/abs/1502.03167

Батч-нормализация (Batch normalization)

минибатч $\{x_i\}_{i=1}^m$

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i$$ **среднее по мини-батчу**

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2$$ **дисперсия по мини-батчу**

$$x_i^{\text{new}} = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}}$$ **нормировка**

$$y_i = \gamma x_i^{\text{new}} + \beta$$ **растяжение и сдвиг**

Надо определить параметры γ **и** β

Зачем центрировать, а потом смещать?

Так эффективнее обучать: смещение является параметром в чистом виде

При обучении

среднее и дисперсия – по мини-батчу

При тесте

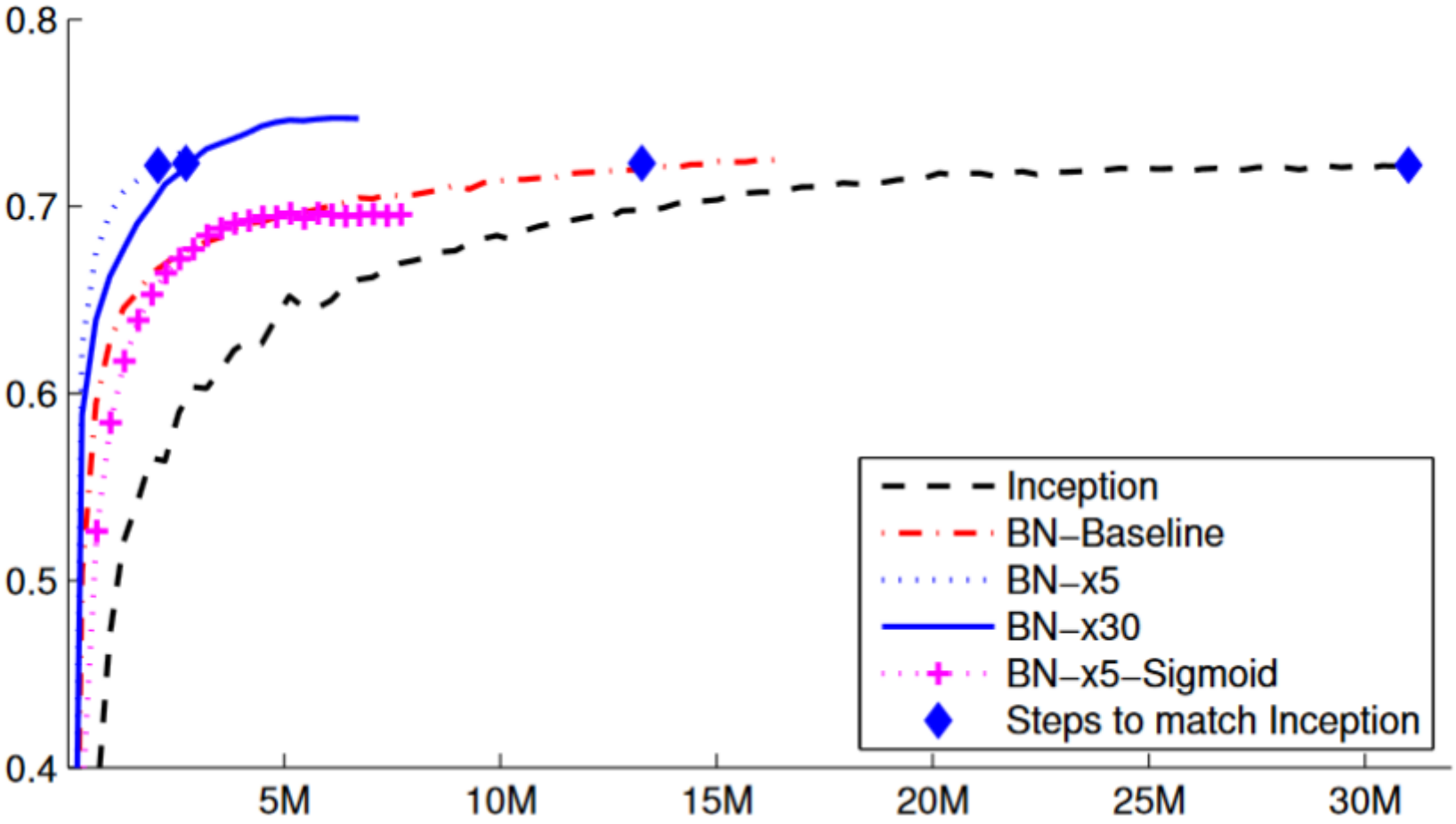
среднее и дисперсия –

- **по обучению**
- **усреднение значений, что были во время обучения**
- **экспоненциальное среднее =*=**

нормализация перед входом в каждый слой (иногда до активации, иногда после)

- **можно увеличить скорость обучения**
- **можно убрать dropout**
- **можно уменьшить регуляризацию**
- **можно использовать более глубокие сети**

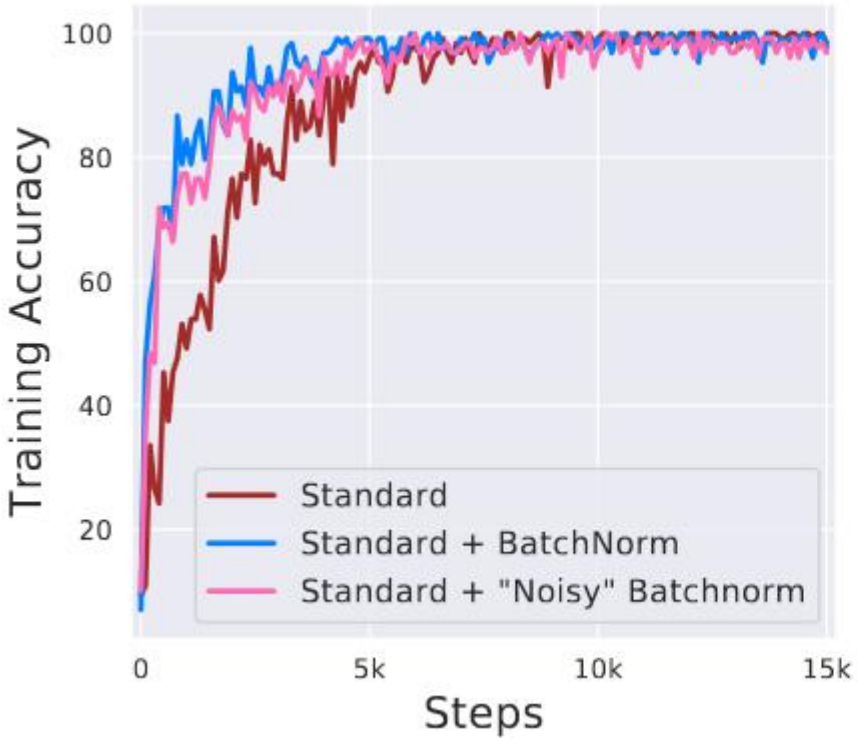
Батч-нормализация (Batch normalization)

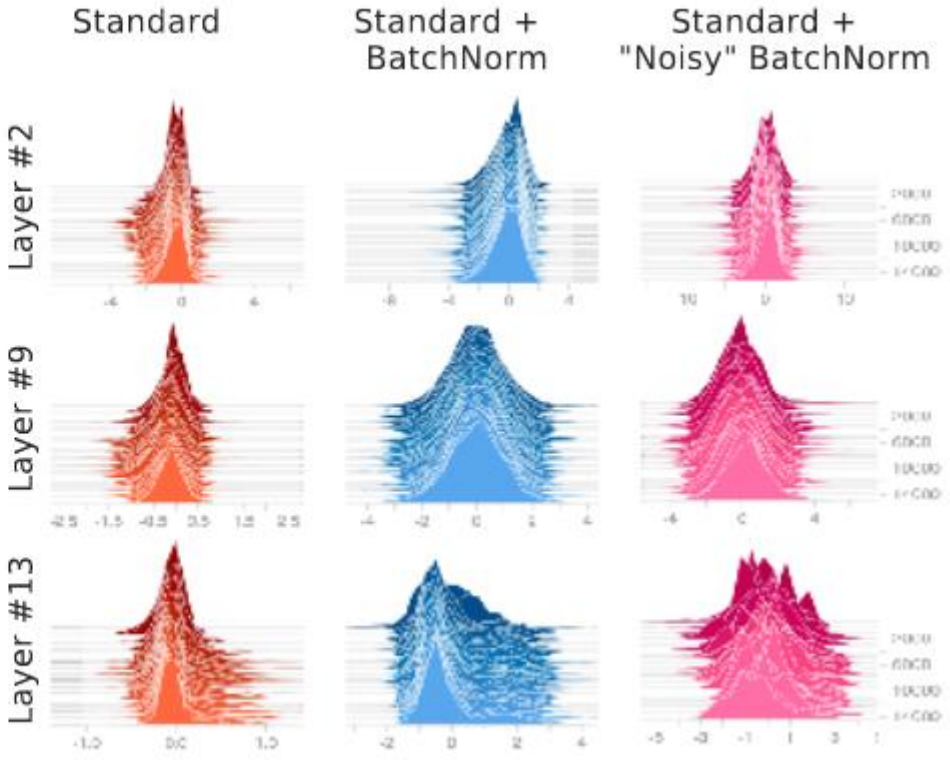


Обучение Inception с / без батч-нормализацией

Батч-нормализация (Batch normalization)

– обоснование эффективности – открытая проблема





дело не в изменении распределения, а в сглаживании функции

Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, Aleksander Madry

How Does Batch Normalization Help Optimization? 2018 // https://arxiv.org/abs/1805.11604

Батч-нормализация (Batch normalization)

```
from keras.layers.normalization import BatchNormalization

inp_norm = BatchNormalization(axis=1)(inp) # применить к inp
# conv_1 = Convolution2D(...)(inp_norm)
conv_1 = BatchNormalization(axis=1)(conv_1) # применить к conv_1
```

Расширение обучающего множества (Data Augmentation)

Аугментация – построение дополнительных данных их исходных

Изображения

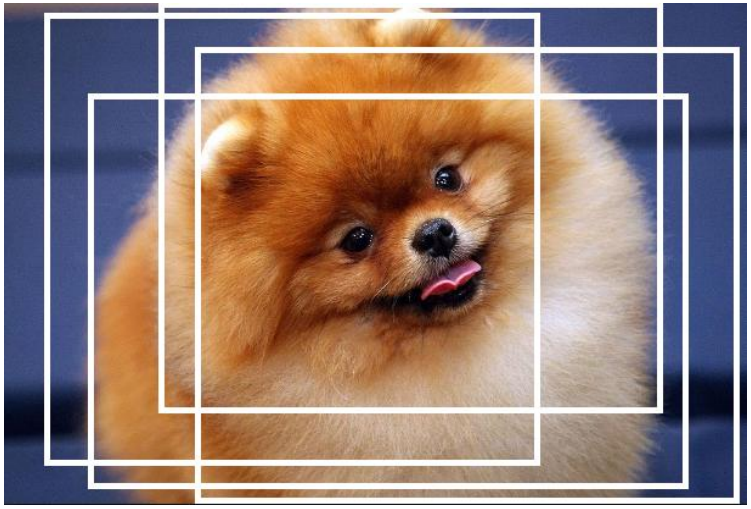
- симметрии (flip)
- вырезки (crop)
- изменение масштаба (rescaling)
- случайные модификации (+шум)
- повороты (rotation)
- сдвиги (shift)
- изменение яркости, контраста, палитры
- эффекты линзы
- перерисовка изображения (ex GAN)

Звук

- +фоновый шум
- тональность

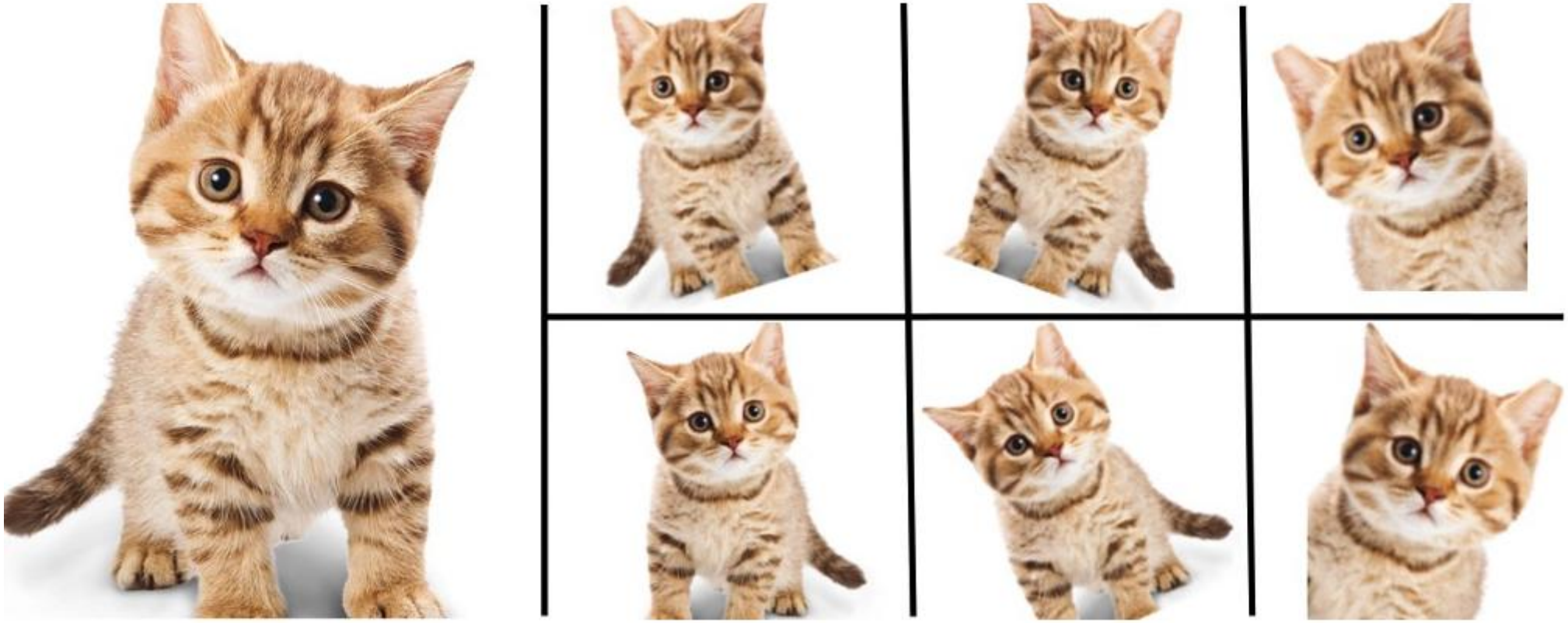
Текст

- замена синонимов



Тонкость: преобразования могут переводить объект в другой класс, например повороты «6» и «9».

Расширение обучающего множества (Data Augmentation)



https://medium.com/nanonets/how-to-use-deep-learning-when-you-have-limited-data-part-2-data-augmentation-c26971dc8ced

https://github.com/aleju/imgaug

Расширение обучающего множества (Data Augmentation)

```
from keras.preprocessing.image import ImageDataGenerator
# после model.compile(...)

datagen = ImageDataGenerator(
            width_shift_range=0.1, # случайные сдвиги
            height_shift_range=0.1) # случайные сдвиги
datagen.fit(X_train)

# обучение на батчах, которые генерирует datagen.flow()

model.fit_generator(datagen.flow(X_train, Y_train,
                        batch_size=batch_size),
                        samples_per_epoch=X_train.shape[0],
                        nb_epoch=num_epochs,
                        validation_data=(X_val, Y_val),
                        verbose=1)
```

Ансамбль нейросетей

- несколько независимых моделей (как всегда +2%)
- усреднение ~ одной НС на разных эпохах обучения (аналог усреднения Поляка)

Loshchilov and Hutter, “SGDR: Stochastic gradient descent with restarts”, arXiv 2016

Huang et al, “Snapshot ensembles: train 1, get M for free”, ICLR 2017

```
from keras.layers import merge # for merging predictions in an ensemble
# ...
ens_models = 3 # we will train three separate models on the data
# ...
inp_norm = BatchNormalization(axis=1)(inp) # Apply BN to the input (N.B. need to rename here)

outs = [] # the list of ensemble outputs
for i in range(ens_models):
    # conv_1 = Convolution2D(...)(inp_norm)
    # ...
    outs.append(Dense(num_classes, init='glorot_uniform', W_regularizer=l2(l2_lambda),
activation='softmax')(drop)) # Output softmax layer

out = merge(outs, mode='ave') # average the predictions to obtain the final output
```

Диагностика проблем с НС

1. Численно проверить градиенты (с помощью конечных разностей)

$$\frac{\partial f(x)}{\partial x} \approx \frac{f(x+\varepsilon) - f(x-\varepsilon)}{2\varepsilon}$$

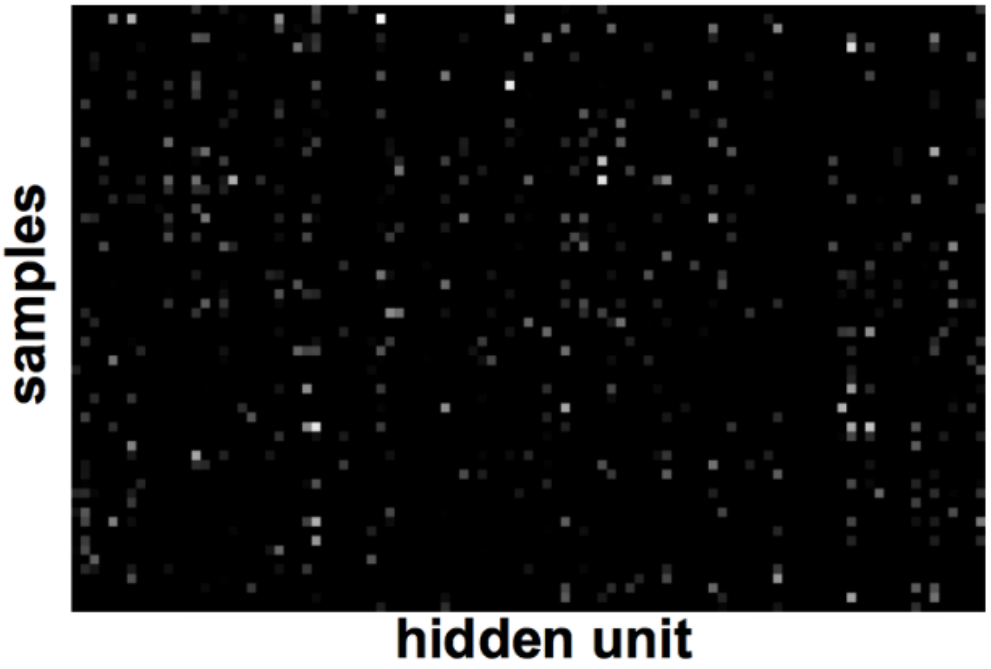
проверка реализации обратного и прямого распространения

Диагностика проблем с НС

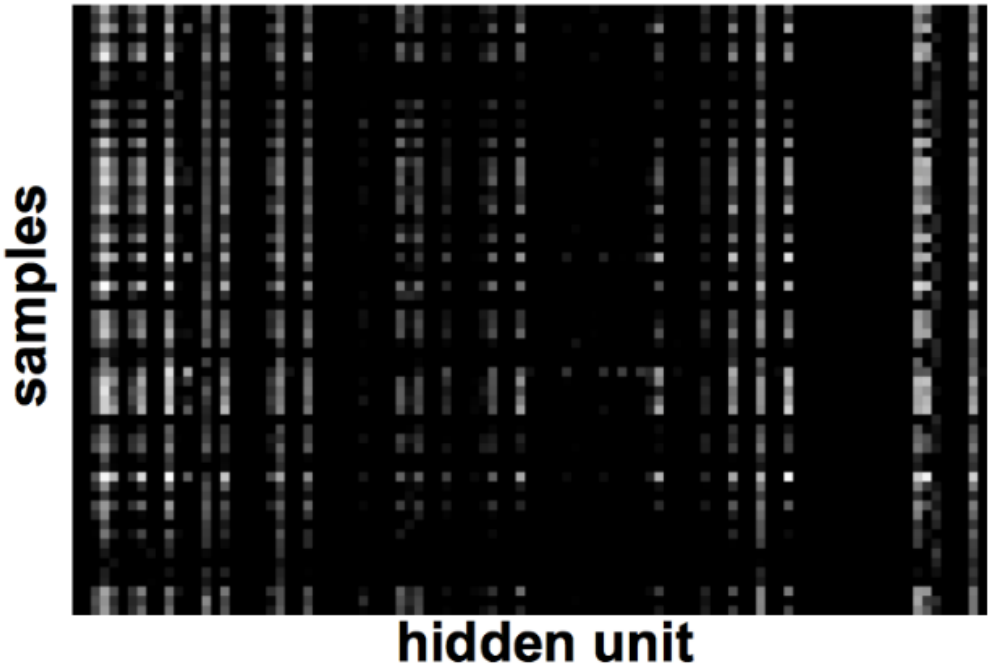
2. Визуализация

Признаки (в общем смысле) должны быть некоррелированными и с большой дисперсией

Хорошо:

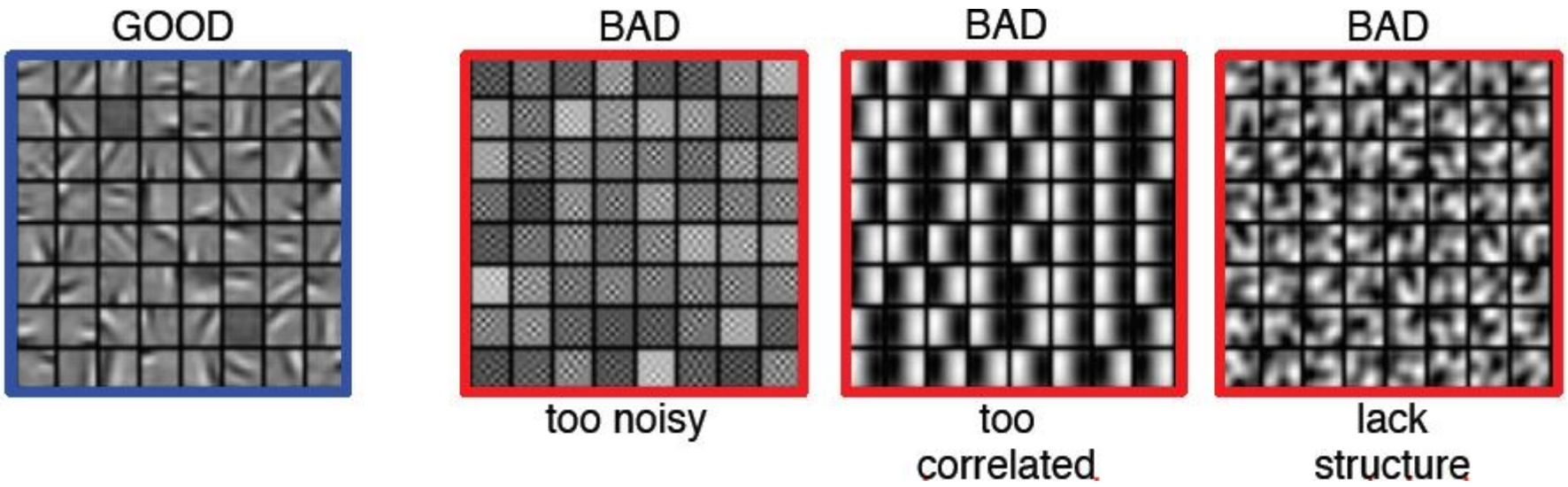


Плохо:



Marc'Aurelio Ranzato, CVPR 2014

Хорошие фильтры имеют структуру и некоррелированные



3. Убедиться, что сеть работает на небольшом куске данных

(~ 100 – 500 объектов)

4. Насыщены ли нейроны ещё до обучения?

Нормировка!

5. Как ведёт себя ошибка обучения

Настроить темп обучения!

6. Насколько меняются веса за итерацию (~ 0.1%)

Недообучение

Другие методы оптимизации

GPU

Переобучение

Регуляризация

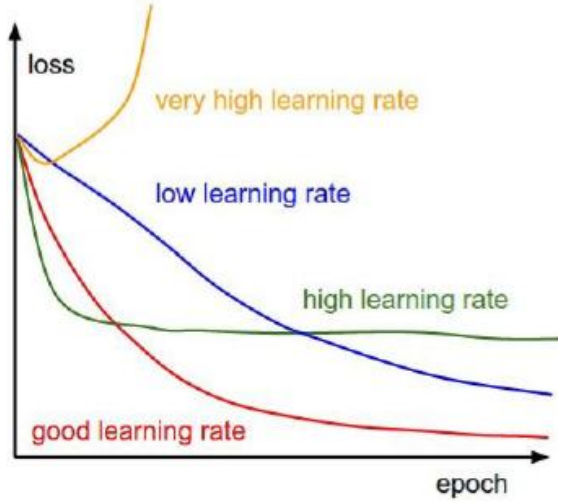
Обучение без учителя

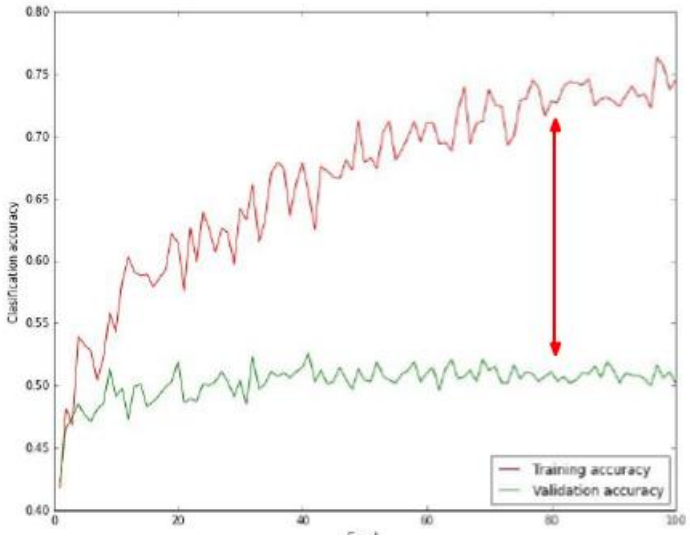
(более сложная задача ⇒ меньше переобучения)

Dropout

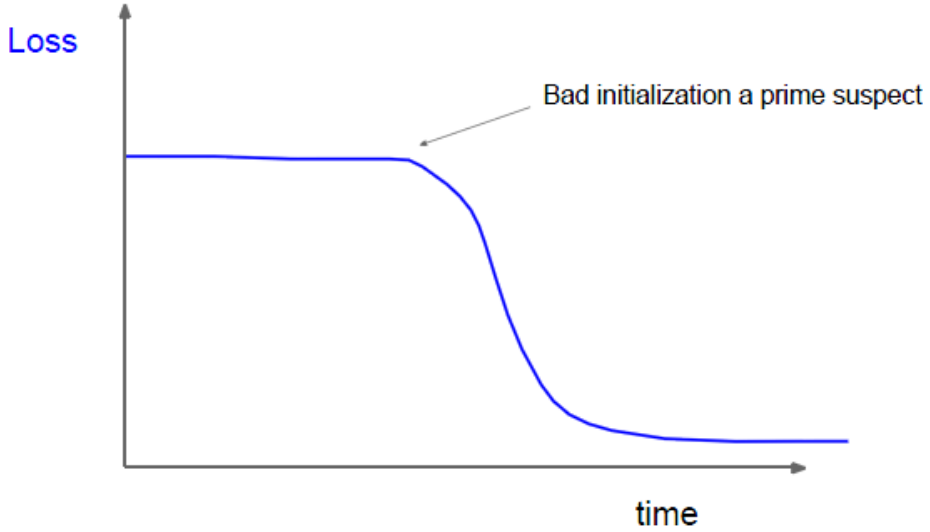
Кривые ошибок

Настройка темпа





Плохая инициализация

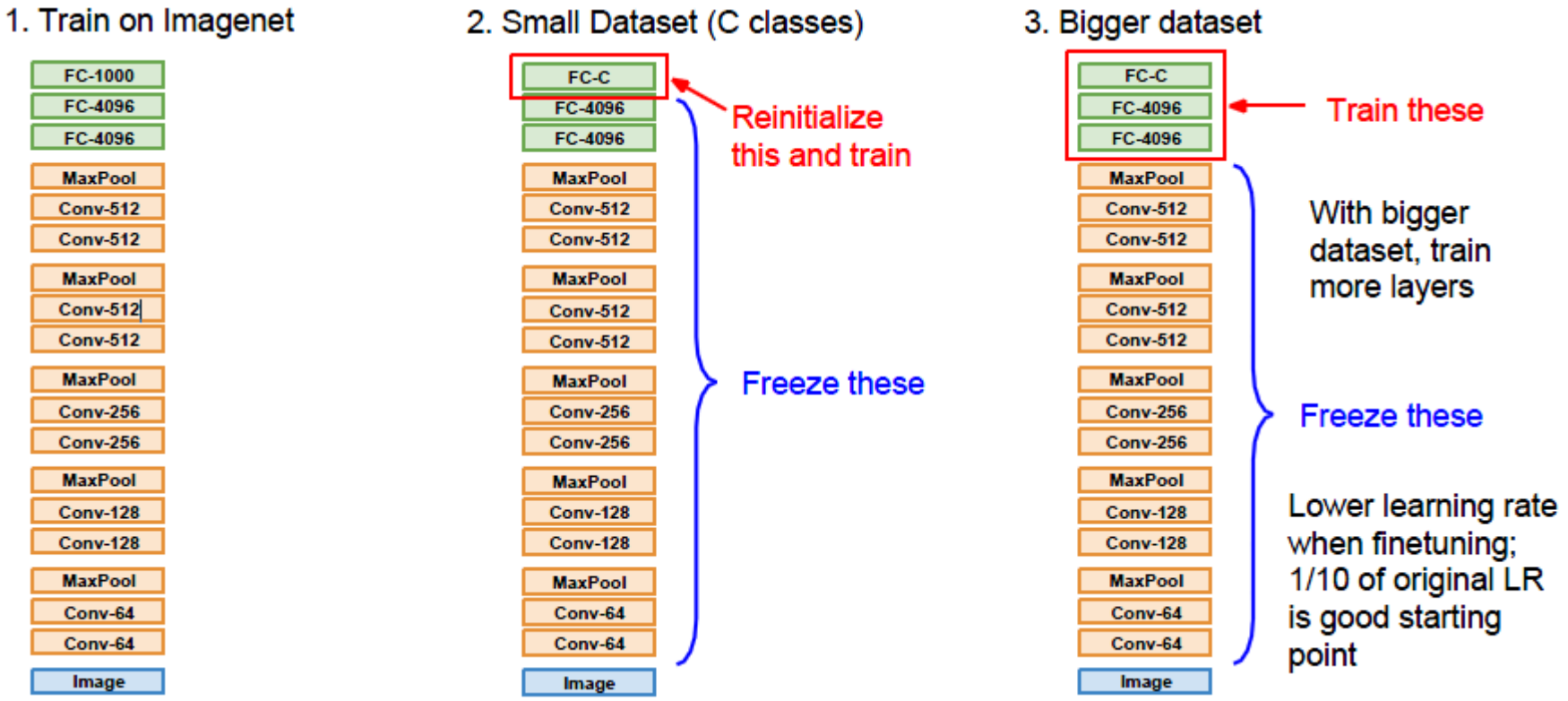


Большой зазор ⇒ переобучение ⇒ усилить регуляризацию

Маленький ⇒ усложнить модель (?)

Transfer Learning

Чтобы решать задачи нужны данные… берём предобученную НС



[cs231]

Transfer Learning

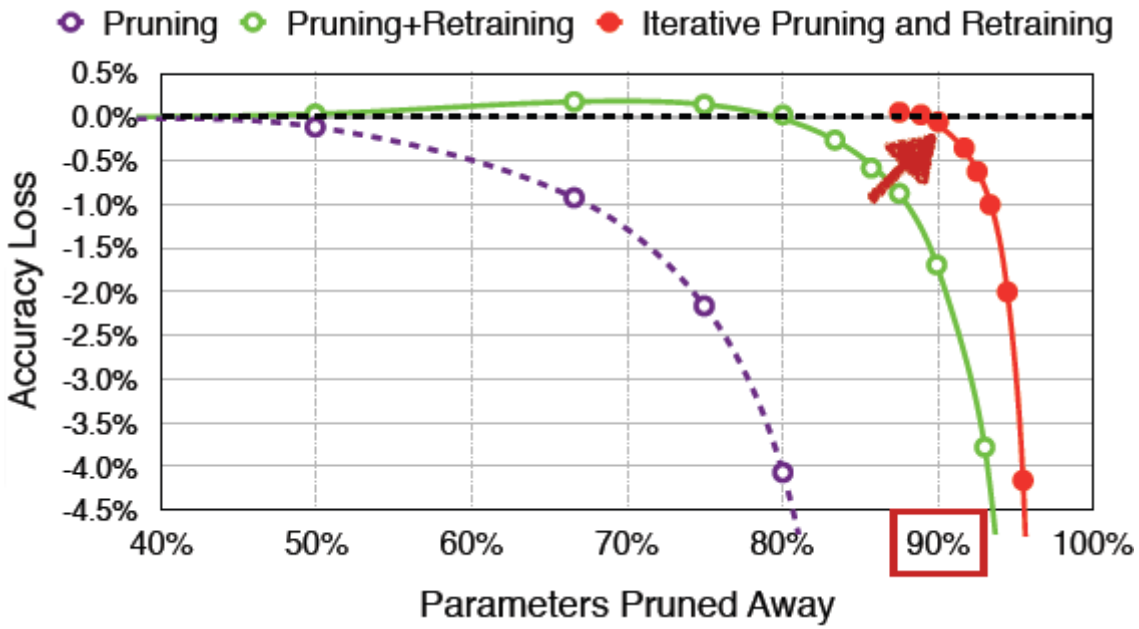
Есть предтренированные НС:

Caffe: https://github.com/BVLC/caffe/wiki/Model-Zoo
TensorFlow: https://github.com/tensorflow/models
PyTorch: https://github.com/pytorch/vision

Есть предобучение с учителем

Узкие глубокие сети учат с помощью уже обученных неглубоких широких

Упрощение НС (Pruning)



[Han et al. NIPS'15]

- **Original** : a man is riding a surfboard on a wave
- **Pruned 90%**: a man in a wetsuit is riding a wave on a beach

- **Original** : a soccer player in red is running in the field
- **Pruned 95%**: a man in a red shirt and black and white black shirt is running through a field

Ещё… нормализации

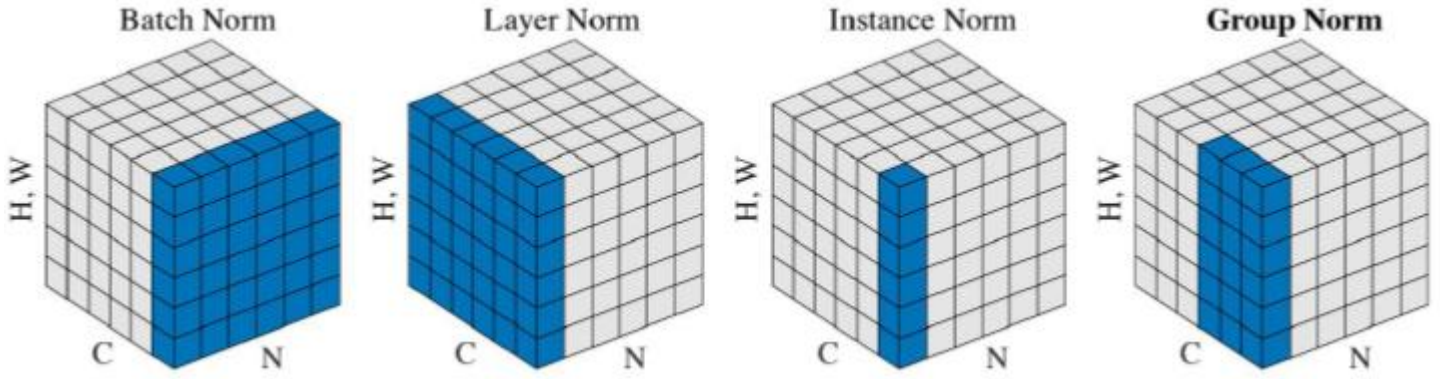
Layer Normalization

Ba, Kiros, and Hinton, «Layer Normalization», arXiv 2016

Instance Normalization

Ulyanov et al, Improved Texture Networks: Maximizing Quality and Diversity in Feed-forward Stylization and Texture Synthesis, CVPR 2017

Wu and He, “Group Normalization”, arXiv 2018



Практические советы

- начинайте с простых архитектур и методов (оптимизации)
- начинайте с небольшого набора данных (для начальных экспериментов)
- выбирайте правильную архитектуру (классификация изображений – CNN, последовательности – LSTM/GRU и т.п.)
- Добавление параметров $\Rightarrow$ усложнение сети (больше времени на обучение, риск переобучения)
- Используйте средства борьбы с переобучением (см. выше)
- Если данных слишком много средства могут не понадобиться. Если можно – собирайте данные!
- Используйте уже натренированные модели.
- Learning rate часто самый важный параметр – лучше уменьшать
- Есть методы настройки параметров лучше, чем structured (grid) search.
- Визуализируйте!
- 1% Rule (???) – веса должны меняться на 1% от своих значений

..