

# Глубокое обучение и вообще

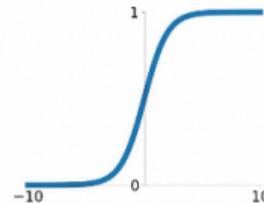
Ульянкин Филипп

**Посиделка 7:** Нейросети – конструктор LEGO (часть 2)

# В предыдущих сериях: активация

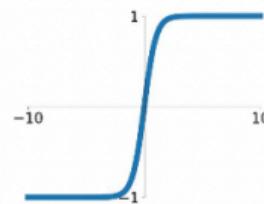
## Sigmoid

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



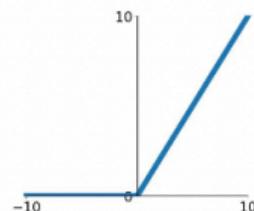
## tanh

$$\tanh(x)$$



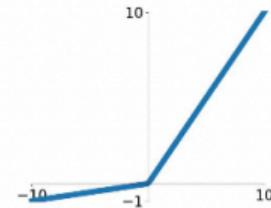
## ReLU

$$\max(0, x)$$



## Leaky ReLU

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

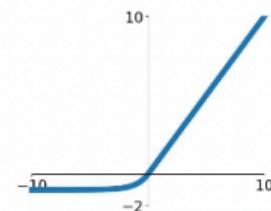


## Maxout

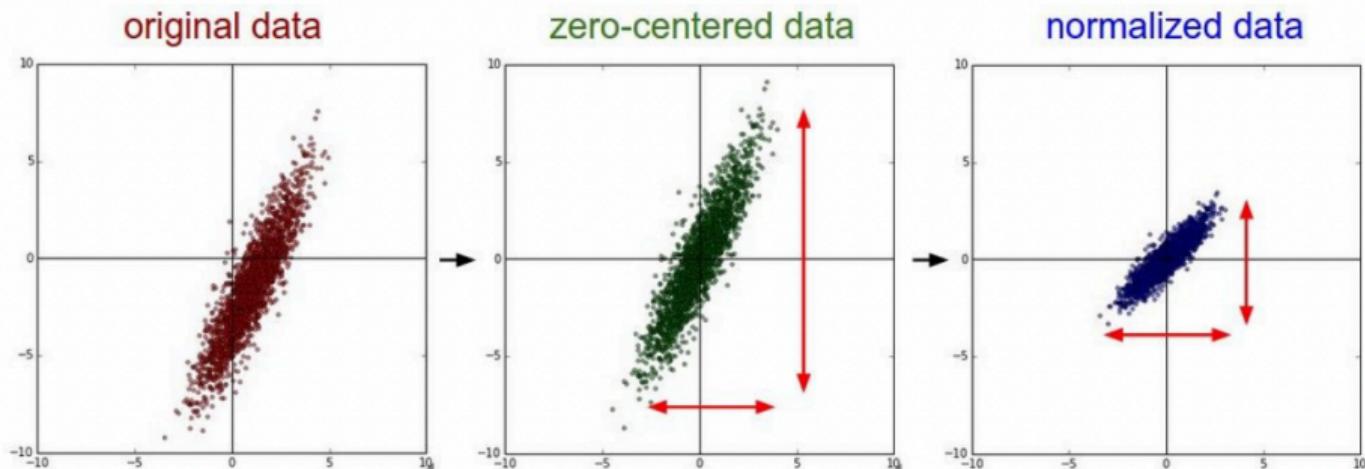
$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

## ELU

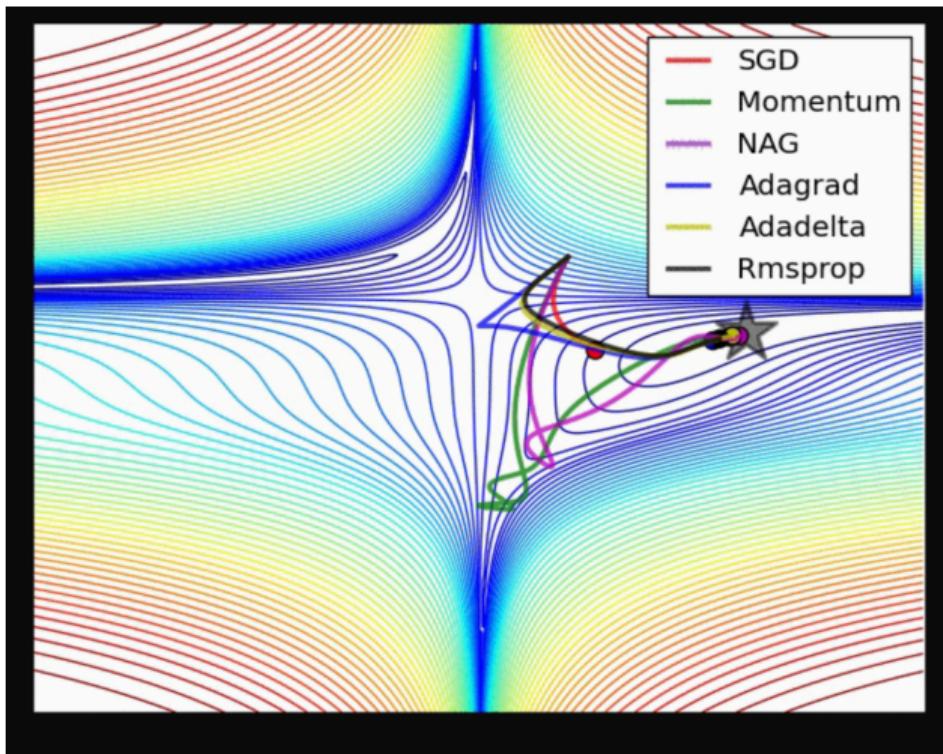
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



# В предыдущих сериях: предобработка



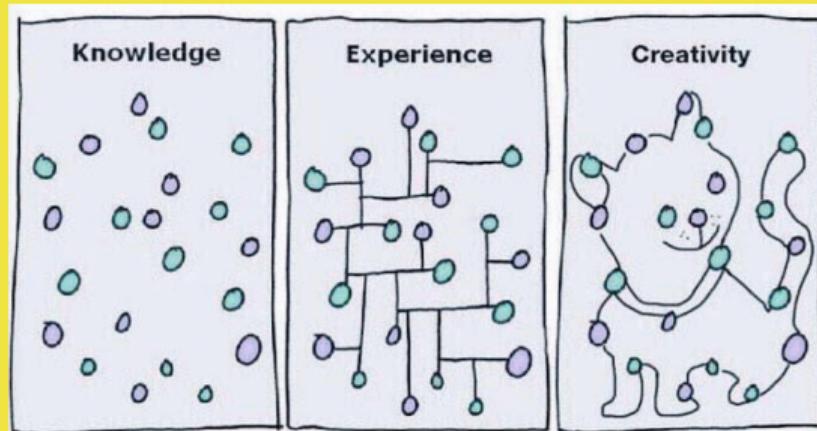
В предыдущих сериях: 50 оттенков градиентного спуска



# Agenda

- Переобучение нейронных сетей
- Дропаут и ранняя остановка
- Регуляризация и weight decay
- Learning rate (LR) scheduling
- Инициализация весов
- Нормализация по батчам
- Организация DL-экспериментов

# Переобучение нейронных сетей



# Что такое асимптотика?

- $n$  — число наблюдений
- Обучение почти любой ML-модели — максимизация правдоподобия
- При  $n \rightarrow \infty$  модель работает хорошо

# Что такое асимптотика?

- $n$  — число наблюдений
- Обучение почти любой ML-модели — максимизация правдоподобия
- При  $n \rightarrow \infty$  модель работает хорошо

# Что такое асимптотика?

- $n$  — число наблюдений  $k$  — число параметров
- Обучение почти любой ML-модели — максимизация правдоподобия
- При  $n \rightarrow \infty$  модель работает хорошо
- При  $\frac{n}{k} \rightarrow \infty$  модель работает хорошо

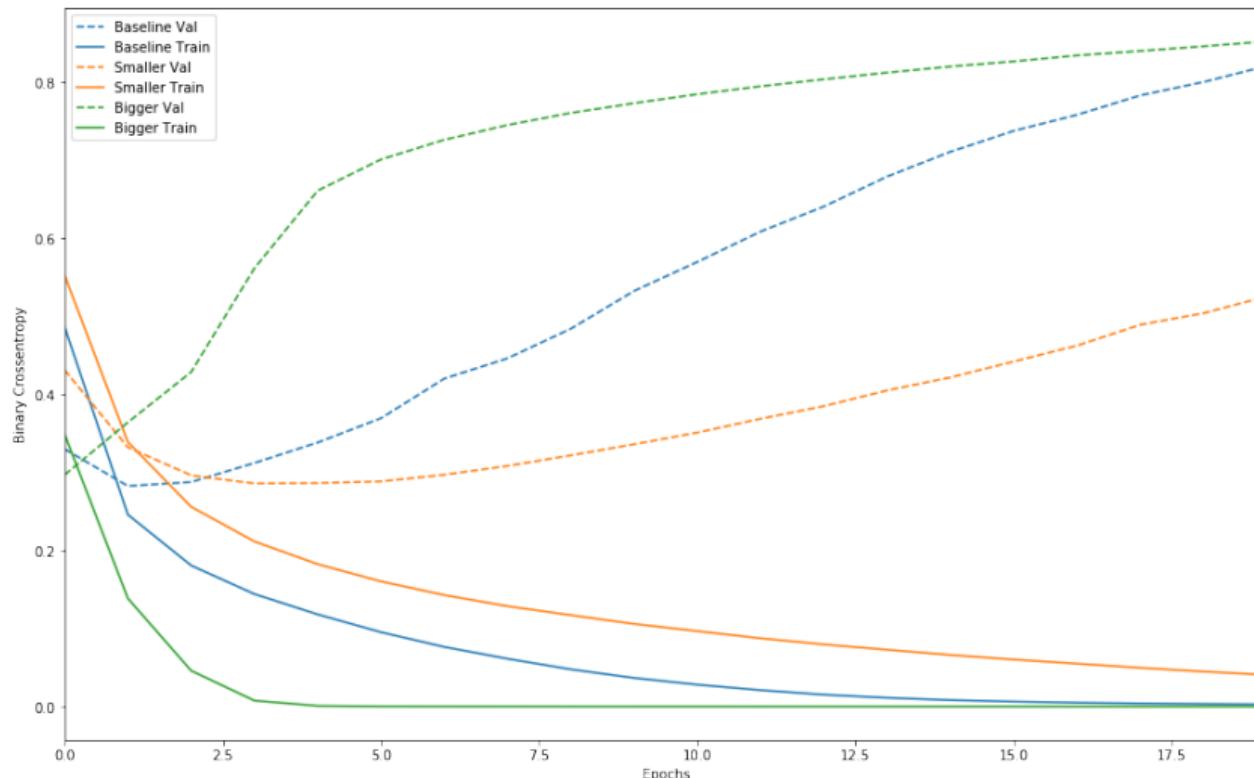
# Что такое асимптотика?

- $n$  — число наблюдений  $k$  — число параметров
- Обучение почти любой ML-модели — максимизация правдоподобия
- При  $n \rightarrow \infty$  модель работает хорошо
- При  $\frac{n}{k} \rightarrow \infty$  модель работает хорошо
- Мы живём в эпоху перепараметризованных моделей,  
 $k$  сильно больше  $n$
- У таких моделей вскрывается много интересных свойств [1]

[1] <https://www.youtube.com/watch?v=SKYXBPCJHCg&t>



# Размеры сеток и переобучение



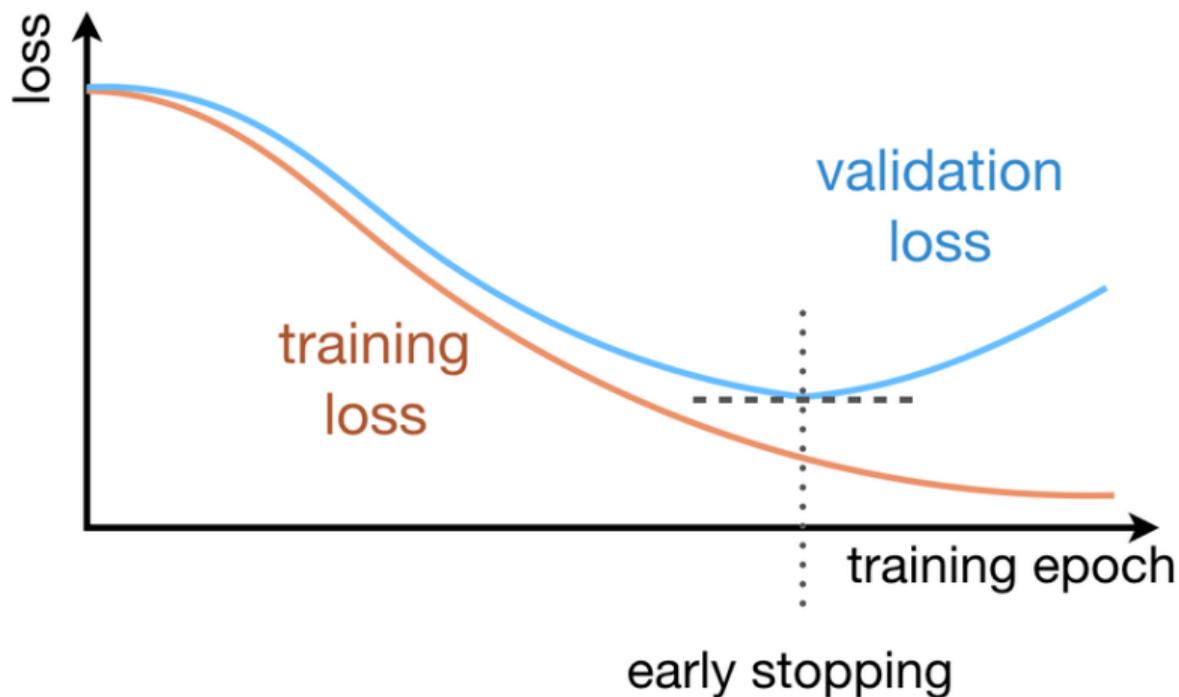
# Борьба с переобучением

- Граница сильно размыта
- Сокращение числа параметров, изобретение более эффективных архитектур
- Регуляризация, разные способы мешать модели подгоняться под обучающую выборку
- Более эффективные методы оптимизации

# Регуляризация

- Любой дополнительный шум в данных — регуляризация
- $L_2$ : приплюсовываем к функции потерь  $\lambda \cdot \sum w_i^2$
- $L_1$ : приплюсовываем к функции потерь  $\lambda \cdot \sum |w_i|$
- Вместо классической регуляризации обычно используют weight decay

# Ранняя остановка (early stopping)



# Ранняя остановка (early stopping)

- Будем останавливать обучение, когда качество на валидации начинает падать
- Ранняя остановка действует примерно также, как регуляризаторы
- Например, в книге Гудфеллоу "Глубокое обучение" на стр. 218 можно найти, что ранняя остановка для линейных моделей эквивалентна  $l_2$  регуляризации с MSE, обучаемой SGD

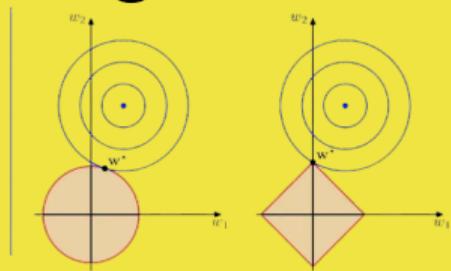
# Регуляризация и Dropout

- На практике обычно используют Dropout
- Он работает похожим на регуляризаторы образом
- Например, в [1] написано:

«We show that the dropout regularizer is first-order equivalent to an L2 regularizer applied after scaling the features by an estimate of the inverse diagonal Fisher information matrix»

[1] <https://arxiv.org/abs/1307.1493>

# Weight decay



# Momentum SGD

$$Q_\lambda(w) = \frac{1}{n} \cdot \sum_{i=1}^n L_i(w) + \frac{1}{2} \lambda \cdot \|w\|_2^2$$

$$\begin{cases} g_t = \nabla_w Q(w_{t-1}) + \lambda \cdot w_{t-1} \\ m_t = \mu \cdot m_{t-1} + g_t \\ w_t = w_{t-1} - \eta_t \cdot m_t \end{cases}$$

Подставим первую строку во вторую, а вторую в третью:

$$\begin{aligned} w_t &= w_{t-1} - \eta_t \cdot (\mu \cdot m_{t-1} + \nabla_w Q(w_{t-1}) + \lambda \cdot w_{t-1}) = \\ &= \underbrace{(1 - \eta_t \cdot \lambda)}_{<1} \cdot w_{t-1} - \eta_t \cdot (\mu \cdot m_{t-1} + \nabla_w Q(w_{t-1})) \end{aligned}$$

## Weight decay для momentum SGD

- мы шагаем по старому градиенту (без регуляризатора), но из новых весов

$$w_t = \underbrace{(1 - \eta_t \cdot \lambda)}_{<1} \cdot w_{t-1} - \eta_t \cdot (\mu \cdot m_{t-1} + \nabla_w Q(w_{t-1}))$$

- $\lambda$  — weight decay
- типичные значения  $\lambda$  для нейронок:  $1e-4, 1e-5$

# Adam

$$Q_\lambda(w) = \frac{1}{n} \cdot \sum_{i=1}^n L_i(w) + \frac{1}{2} \lambda \cdot \|w\|_2^2$$

$$\begin{cases} g_t = \nabla_w Q(w_{t-1}) + \lambda \cdot w_{t-1} \\ m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t \\ v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2 \\ \hat{m}_t = \frac{1}{1 - \beta_1^t} \cdot m_t \\ \hat{v}_t = \frac{1}{1 - \beta_2^t} v_t \\ w_t = w_{t-1} - \eta_t \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \end{cases}$$

Гиперпараметры от авторов (обычно их не меняют):

$$\beta_1 = 0.9, \beta_2 = 0.999$$

# Adam

$$\begin{aligned} w_t &= w_{t-1} - \eta_t \cdot \frac{m_t}{1 - \beta_1^t} \cdot \frac{1}{\sqrt{\hat{v}_t} + \varepsilon} = \\ &= w_t = w_{t-1} - \eta_t \cdot \frac{\beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t}{1 - \beta_1^t} \cdot \frac{1}{\sqrt{\hat{v}_t} + \varepsilon} = \\ &= w_t = w_{t-1} - \eta_t \cdot \frac{\beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot (\nabla_w Q(w_{t-1}) + \lambda \cdot w_{t-1})}{1 - \beta_1^t} \cdot \frac{1}{\sqrt{\hat{v}_t} + \varepsilon} = \\ &= w_{t-1} \cdot \left( \underbrace{\frac{1}{\text{вектор единиц}}}_{\text{вектор единиц}} - \frac{\eta_t \cdot \lambda \cdot (1 - \beta_1)}{1 - \beta_1^t} \cdot \frac{1}{\sqrt{\hat{v}_t} + \varepsilon} \right) - \dots \end{aligned}$$

Регуляризация работает по-разному — веса штрафуются по-разному, мы хотели от регуляризации не этого!

## AdamW

Это модификация Adam, в которой мы делаем честный weight decay, из-за этого сетка должна меньше переобучаться

$$\begin{cases} g_t = \nabla_w Q(w_{t-1}) \\ m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t \\ v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2 \\ \hat{m}_t = \frac{1}{1-\beta_1^t} \cdot m_t \\ \hat{v}_t = \frac{1}{1-\beta_2^t} v_t \\ w_t = (1 - \eta_t \cdot \lambda) \cdot w_{t-1} - \eta_t \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon} \end{cases}$$

<https://arxiv.org/pdf/1711.05101.pdf>

# Ландшафт функции потерь

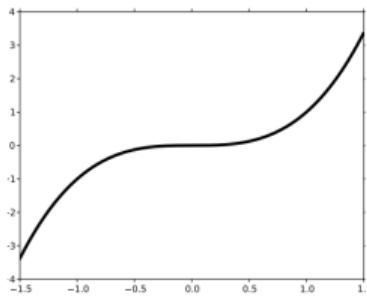


# Боб чилит в локальном минимуме

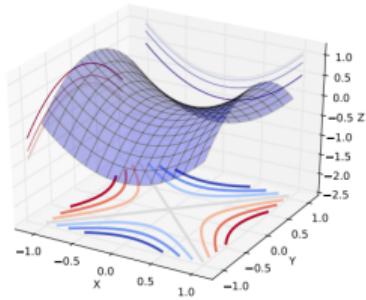


<https://hackernoon.com/life-is-gradient-descent-880c60ac1be8>

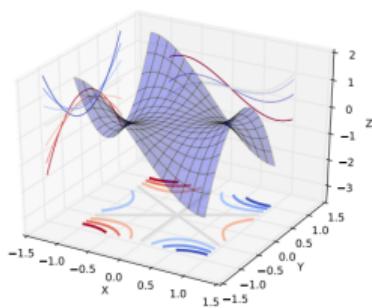
# Седловые точки



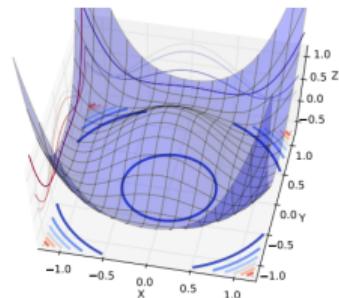
(a)



(b)



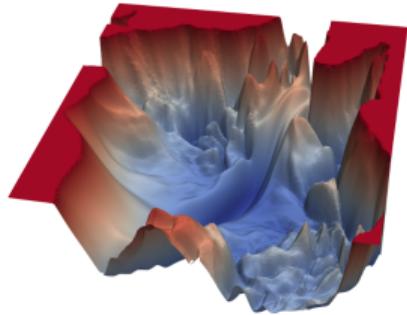
(c)



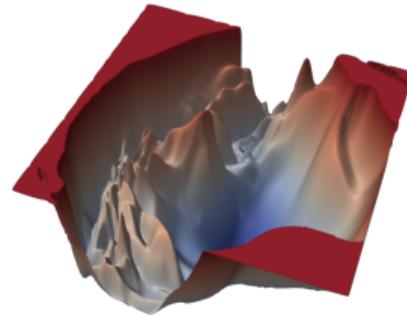
(d)

# Визуализация потерь

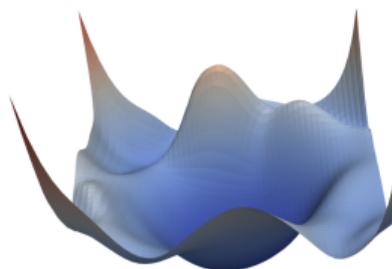
VGG-56



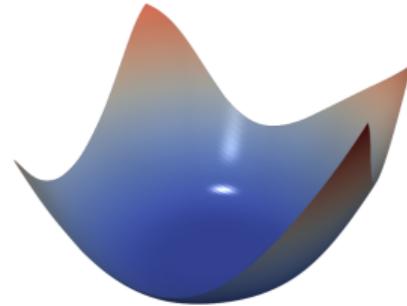
VGG-110



Renset-56



Densenet-121

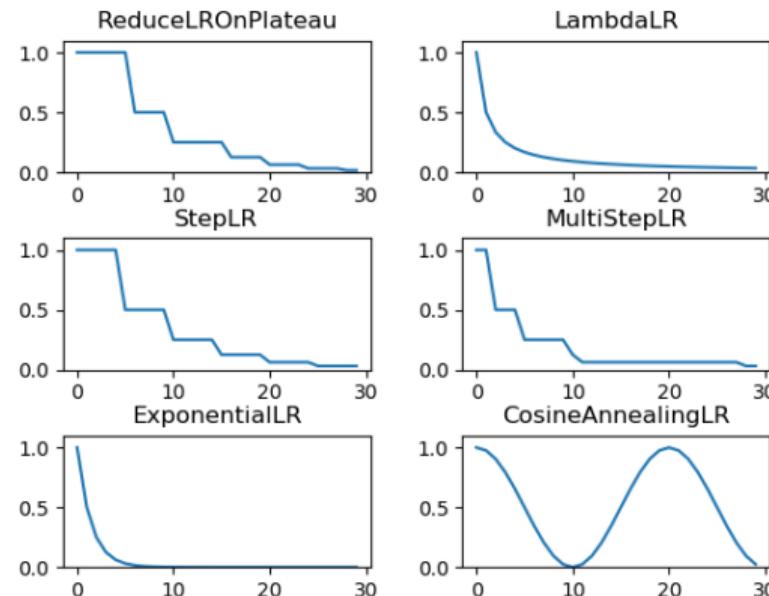


<https://arxiv.org/pdf/1712.09913.pdf>

<https://github.com/tomgoldstein/loss-landscape>

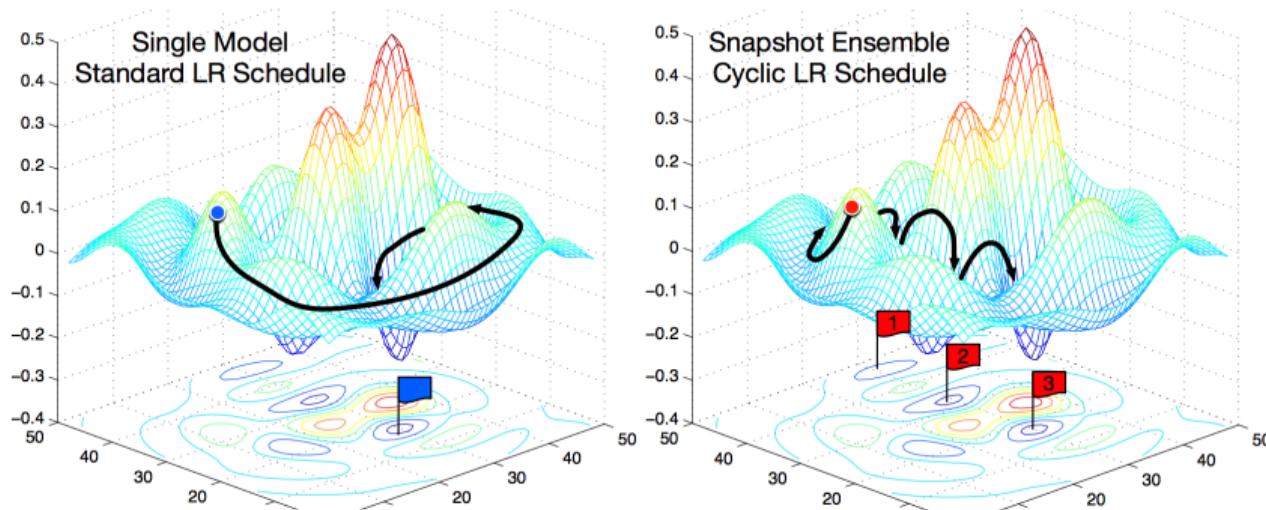
# LR scheduling

- Процедура изменения learning rate по какому-то правилу
- LR во время обучения всегда надо изменять (теряем проценты в качестве)

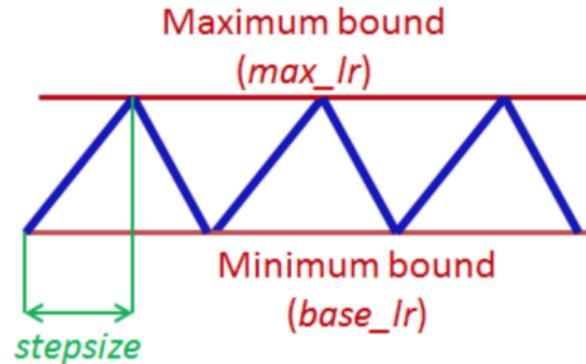
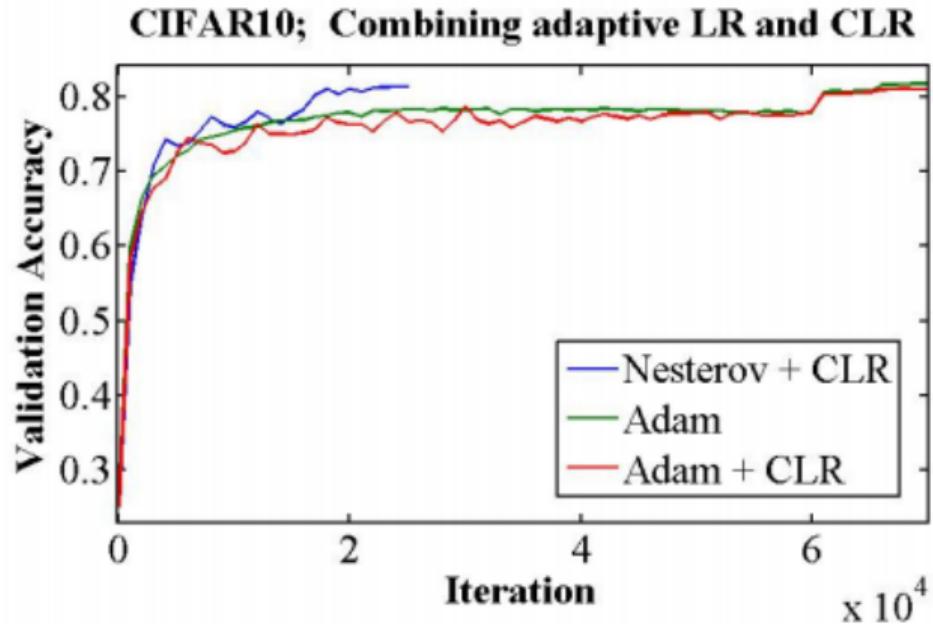


# Циклическая скорость обучения (CLR)

- Хочется, чтобы был шанс вылезти из локального минимума, а также шанс сползти с седла  $\Rightarrow$  давайте менять глобальную скорость обучения циклически



# Циклическая скорость обучения (CLR)



Нестеров с CLR отработал быстрее и лучше Adam

Нет одного правильного алгоритма на все случаи

Всегда надо экспериментировать

<https://arxiv.org/pdf/1506.01186.pdf>

<https://openreview.net/pdf?id=BJYwwY9II>

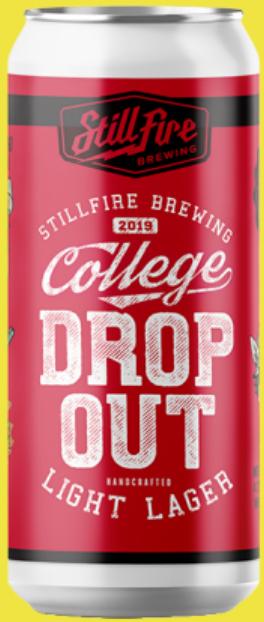
# LRFinder

- Эвристика для подбора стартового значения learning rate
- Увеличиваем learning rate после каждого батча, остановка перед взрывом значений loss-функции
- Есть готовые реализации (в том числе в pytorch и tensorflow)
- Аналогично можно искать оптимальное значение для momentum

<https://sgugger.github.io/how-do-you-find-a-good-learning-rate.html>

<https://arxiv.org/abs/1506.01186>

<https://arxiv.org/abs/1803.09820>



## Хинтон и банк

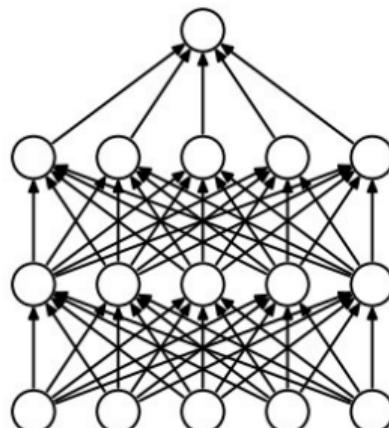
Посещая свой банк я заметил, что операционисты, обслуживающие меня, часто меняются. Я спросил одного из них, почему так происходит. Он сказал, что не знает, но им часто приходится переходить с места на место. Я предположил, что это делается для исключения мошенническогоговора с сотрудником банка.

Это навело меня на мысль, что **удаление случайно выбранного подмножества нейронов** из каждого примера может помочь **предотвратить заговор модели с исходными данными** и ослабить эффект переобучения.

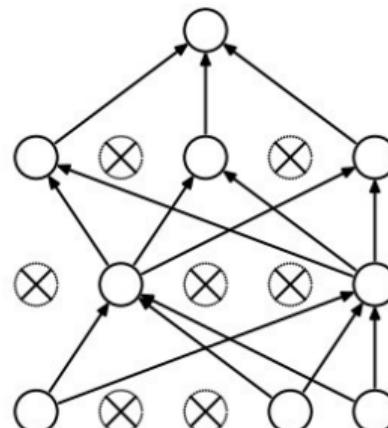
[https://www.reddit.com/r/MachineLearning/comments/4w6tsv/ama\\_we\\_are\\_the\\_google\\_brain\\_team\\_wed\\_love\\_to](https://www.reddit.com/r/MachineLearning/comments/4w6tsv/ama_we_are_the_google_brain_team_wed_love_to)

# Dropout

- При каждом прямом проходе мы зануляем выход нейрона с вероятностью  $p$
- Делает нейроны более устойчивыми к случайным возмущениям



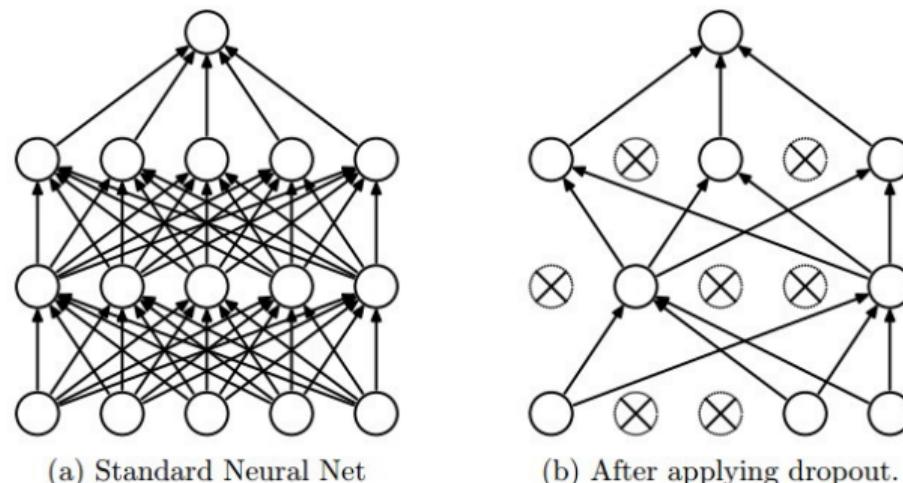
(a) Standard Neural Net



(b) After applying dropout.

# Dropout

- Борьба с ко-адаптацией, не все соседи похожи друг на друга, не все дети похожи на родителей
- Dropout эквивалентен обучению ансамбля из  $2^n$  сетей



## Dropout в формулах

- Можно определить как слой  $d(h)$
- Параметров у слоя нет, есть только гиперпараметр  $p$

## Dropout в формулах

- Можно определить как слой  $d(h)$
- Параметров у слоя нет, есть только гиперпараметр  $p$
- forward pass без dropout:

$$o = f(XW)$$

- forward pass с dropout:

$$o = d \cdot f(XW), \quad d = (d_1, \dots, d_k) \sim iidBern(1 - p)$$

## Dropout в формулах

- Можно определить как слой  $d(h)$
- Параметров у слоя нет, есть только гиперпараметр  $p$
- forward pass без dropout:

$$o = f(XW)$$

- forward pass с dropout:

$$o = d \cdot f(XW), \quad d = (d_1, \dots, d_k) \sim iidBern(1 - p)$$

$$o_i = d_i \cdot f(wx_i^T) = \begin{cases} f(wx_i^T), & 1 - p \\ 0, & p \end{cases}$$

Дропаут — это просто небольшая модификация функции активации

# Dropout в формулах

- forward pass:

$$o = f(X \cdot W)$$

$$o = d \cdot f(X \cdot W), \quad d = (d_1, \dots, d_k) \sim iidBern(1 - p)$$

- backward pass:

$$grad = f'(h) \cdot W \cdot grad$$

$$grad = d \cdot f'(h) \cdot W \cdot grad$$

# Dropout

- При обучении мы домножаем часть выходов на  $d_i$ , тем самым мы изменяем только часть параметров и нейроны учатся более независимо  $\Rightarrow$  регуляризация
- Дропаут выплёвывает случайные выходы, что делать на стадии тестирования?

# Dropout

- При обучении мы домножаем часть выходов на  $d_i$ , тем самым мы изменяем только часть параметров и нейроны учатся более независимо  $\Rightarrow$  **регуляризация**
- Дропаут выплёвывает случайные выходы, **что делать на стадии тестирования?**
- Нам надо сымитировать работу такого ансамбля: можно отключать по очереди все возможные комбинации нейронов, получить  $2^n$  прогнозов и усреднить их

# Dropout

- При обучении мы домножаем часть выходов на  $d_i$ , тем самым мы изменяем только часть параметров и нейроны учатся более независимо  $\Rightarrow$  регуляризация
- Дропаут выплёвывает случайные выходы, что делать на стадии тестирования?
- Нам надо сымитировать работу такого ансамбля: можно отключать по очереди все возможные комбинации нейронов, получить  $2^n$  прогнозов и усреднить их
- Но лучше просто брать по дропауту математическое ожидание:

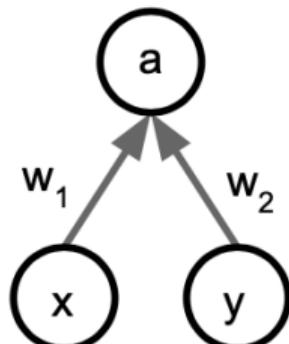
$$o = (1 - p) \cdot f(X \cdot W)$$

# Dropout на тестовой выборке

- Но лучше просто брать по дропауту математическое ожидание:

$$o = (1 - p) \cdot f(X \cdot W)$$

**Пример:** используем дропаут с вероятностью 0.5:



$$\begin{aligned}\mathbb{E}(a) &= \frac{1}{4} \cdot (w_1 x + w_2 y) + \frac{1}{4} \cdot (w_1 x + 0y) + \\ &+ \frac{1}{4} \cdot (0x + w_2 y) + \frac{1}{4} \cdot (0x + 0y) = \frac{1}{2} \cdot (w_1 \cdot x + w_2 \cdot y)\end{aligned}$$

Достаточно домножить выход на 0.5

# Обратный Dropout

- На тесте ищем математическое ожидание:

$$o = (1 - p) \cdot f(X \cdot W)$$

- Это неудобно! Надо переписывать функцию для прогнозов!

# Обратный Dropout

- На тесте ищем математическое ожидание:

$$o = (1 - p) \cdot f(X \cdot W)$$

- Это неудобно! Надо переписывать функцию для прогнозов!
- Давайте лучше будем домножать на  $\frac{1}{1-p}$  на этапе обучения:

train: 
$$o = \frac{1}{1-p} \cdot d \cdot f(X \cdot W)$$

test: 
$$o = f(X \cdot W)$$

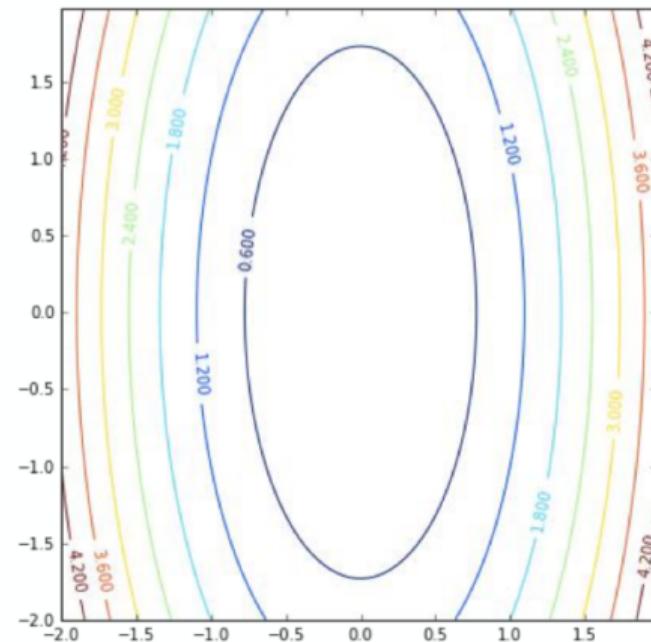
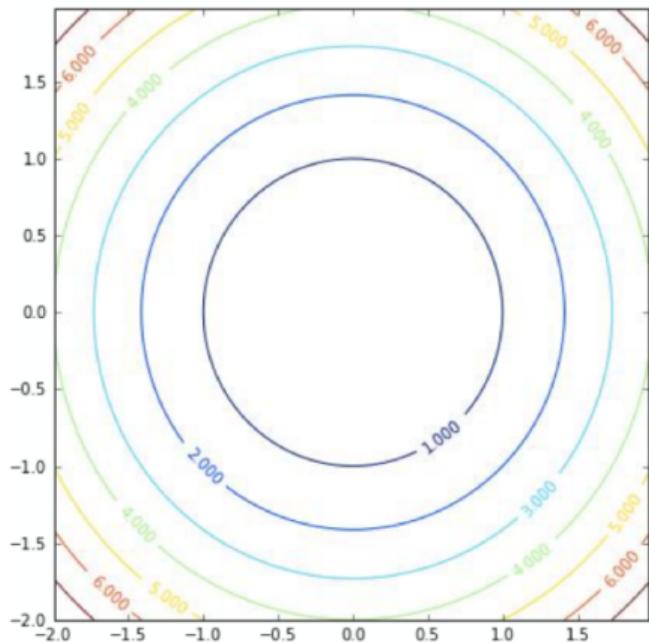
# Интерпретация

- Мы обучаем все возможные архитектуры нейросетей, которые получаются из исходной выбрасыванием отдельных нейронов
- У всех этих архитектур общие веса
- На этапе применения усредняем прогнозы всех этих архитектур

# Батч-нормализация

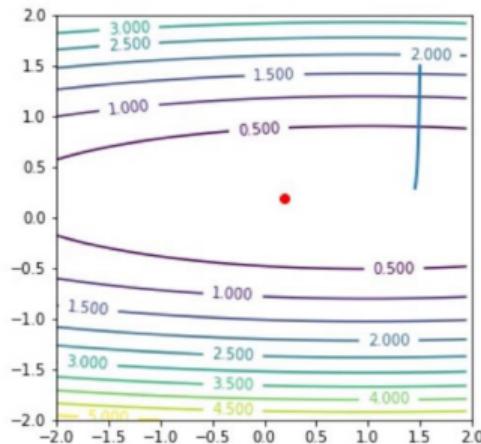
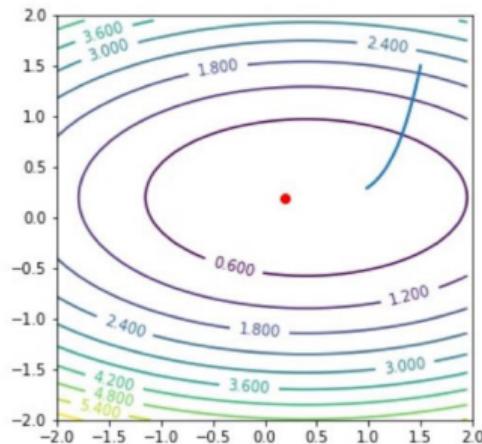
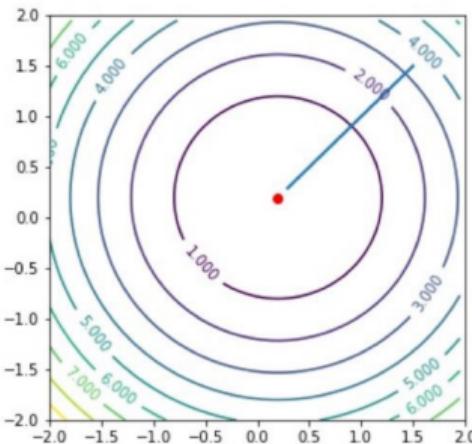


# Стандартизация



Какая из ситуаций лучше для градиентного спуска?

# Стандартизация



Градиентные методы быстрее сходятся, если градиенты в одном масштабе

<https://github.com/hse-aml/intro-to-dl/blob/master/week2/v2/ill-conditioned-demo.ipynb>

# Стандартизация

Будем подавать на вход в нейросетку стандартизованные данные:

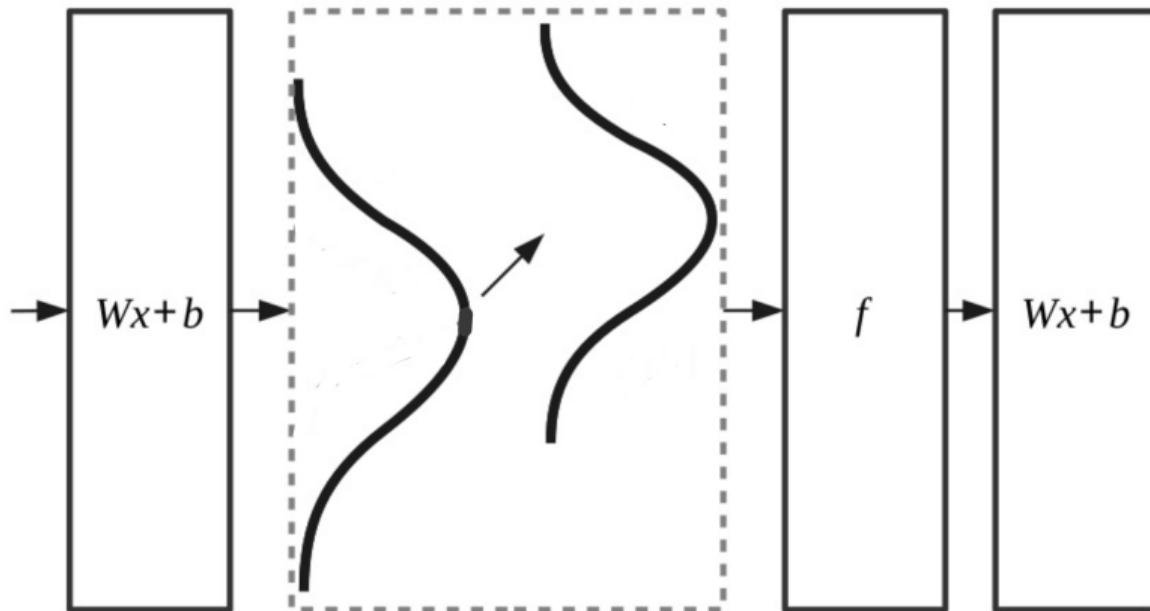
$$\hat{x}_{ij} = \frac{x_{ij} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}$$

$$\mu_j = \frac{1}{n} \sum_{i=1}^n x_{ij}$$

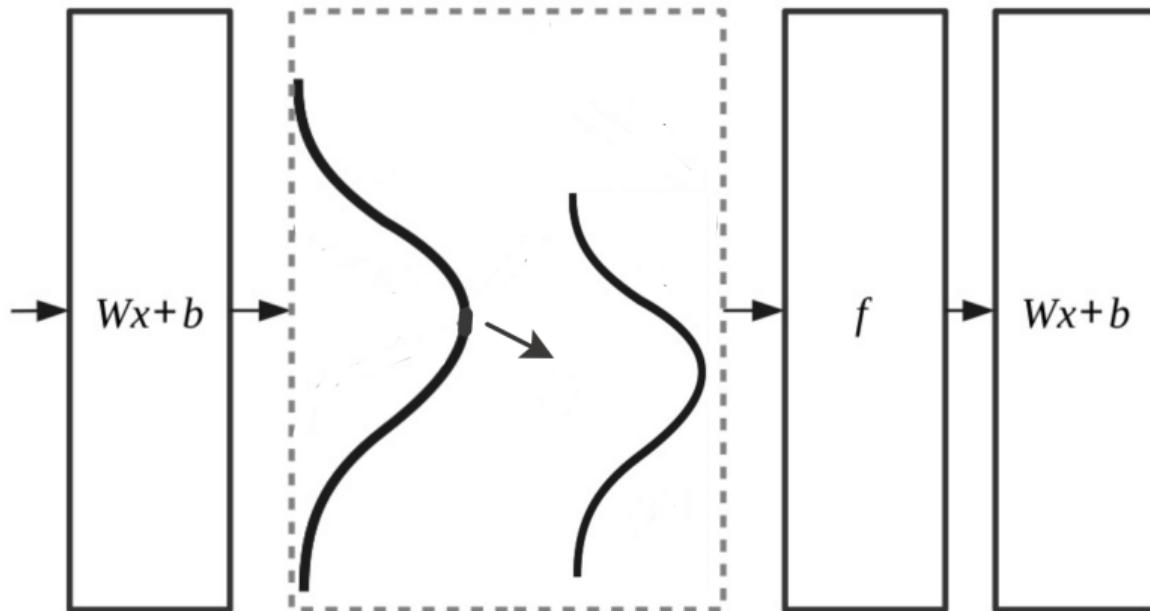
$$\sigma_j^2 = \frac{1}{n} \sum_{i=1}^n (x_{ij} - \mu_j)^2$$

Это помогает градиентному спуску лучше работать

# А что внутри нейросетки?



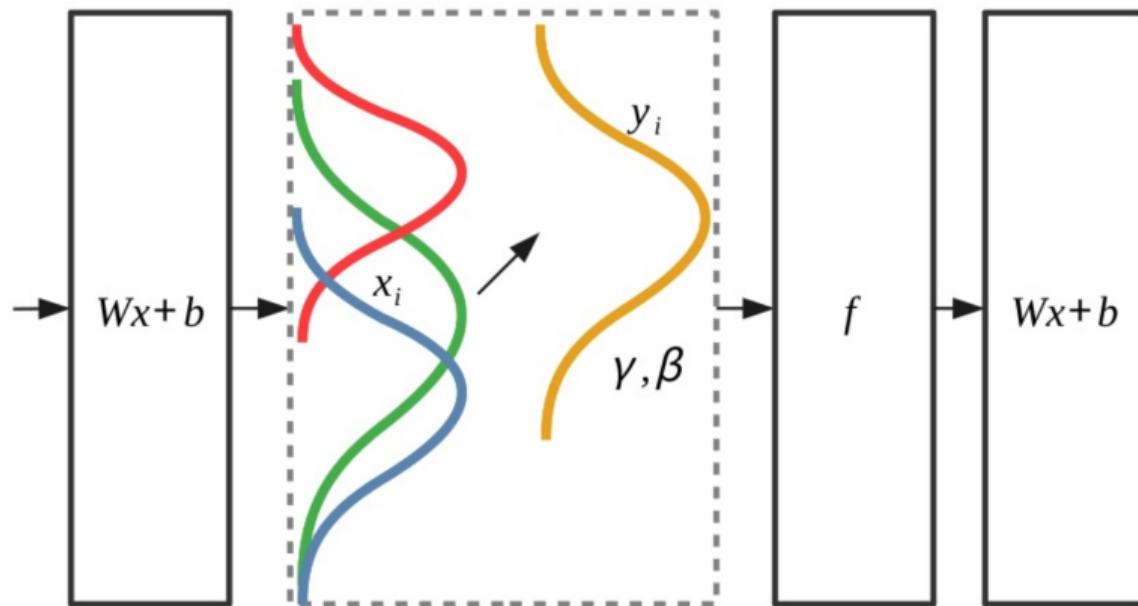
# А что внутри нейросетки?



# Проблема

- Даже если мы стандартизовали вход  $X$ , внутри сетки может произойти несчастье и скрытый слой окажется нестандартизован
- Скрытые представления  $h = f(XW)$  могут менять своё распределение в процессе обучения, это усложняет его
- Давайте на каждом слое вместо  $h$  использовать  $\hat{h} = \frac{h - \hat{\mu}_h}{\hat{\sigma}_h}$
- На выход будем выдавать  $\beta \cdot \hat{h} + \gamma$ , для того, чтобы у нас было больше свободы, параметры  $\beta$  и  $\gamma$  учим как параметры полносвязного слоя

# Batch Normalization (2015)



# Forward pass

**Input:** Values of  $x$  over a mini-batch:  $\mathcal{B} = \{x_1 \dots m\}$ ;

Parameters to be learned:  $\gamma, \beta$

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

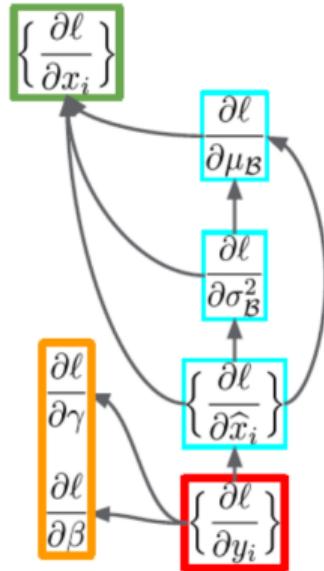
$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^m x_i \quad // \text{mini-batch mean}$$

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

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

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

# Backward pass



$$\frac{\partial \ell}{\partial \hat{x}_i} = \frac{\partial \ell}{\partial y_i} \cdot \gamma$$

$$\frac{\partial \ell}{\partial \sigma_B^2} = \sum_{i=1}^m \frac{\partial \ell}{\partial \hat{x}_i} \cdot (x_i - \mu_B) \cdot \frac{-1}{2} (\sigma_B^2 + \epsilon)^{-3/2}$$

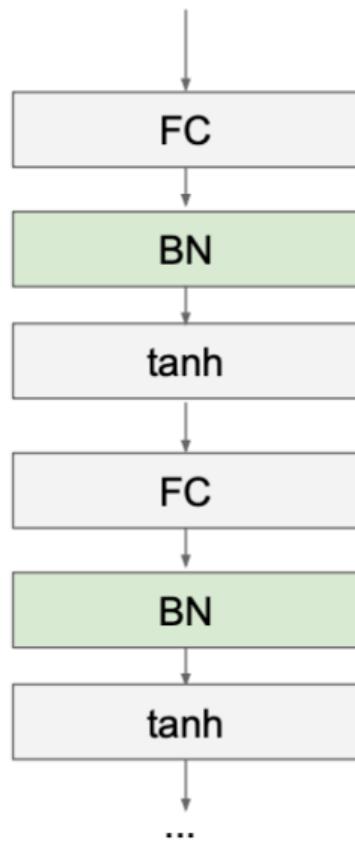
$$\frac{\partial \ell}{\partial \mu_B} = \left( \sum_{i=1}^m \frac{\partial \ell}{\partial \hat{x}_i} \cdot \frac{-1}{\sqrt{\sigma_B^2 + \epsilon}} \right) + \frac{\partial \ell}{\partial \sigma_B^2} \cdot \frac{\sum_{i=1}^m -2(x_i - \mu_B)}{m-1}$$

$$\frac{\partial \ell}{\partial x_i} = \frac{\partial \ell}{\partial \hat{x}_i} \cdot \frac{1}{\sqrt{\sigma_B^2 + \epsilon}} + \frac{\partial \ell}{\partial \sigma_B^2} \cdot \frac{2(x_i - \mu_B)}{m-1} + \frac{\partial \ell}{\partial \mu_B} \cdot \frac{1}{m}$$

$$\frac{\partial \ell}{\partial \gamma} = \sum_{i=1}^m \frac{\partial \ell}{\partial y_i} \cdot \hat{x}_i$$

$$\frac{\partial \ell}{\partial \beta} = \sum_{i=1}^m \frac{\partial \ell}{\partial y_i}$$

# Batch Normalization (2015)



- Делает обучение глубоких сетей проще
- Обеспечивает более высокие темпы обучения и более быструю сходимость
- Сети становятся более устойчивыми к инициализации
- Действует как регуляризатор во время обучения
- Во время обучения и тестирования работает по-разному, это является источником большого числа багов и ошибок

# Batch norm (2015)

- Как считать  $\mu_h$  и  $\sigma_h$ ?
- Оценить по текущему батчу!

$$\mu_h = \alpha \cdot \bar{x}_{batch} + (1 - \alpha) \cdot \mu_h$$
$$\sigma_h^2 = \alpha \cdot \sigma_{batch}^2 + (1 - \alpha) \cdot \sigma_h^2$$

- Коэффициенты  $\beta$  и  $\gamma$  оцениваются в ходе обратного распространения ошибки,  $\mu$  и  $\sigma$  не обучаются
- Обучение довольно сильно ускоряется, сходимость улучшается

<https://arxiv.org/pdf/1502.03167.pdf>

# Batch Normalization for convolutional networks

Batch Normalization for  
**fully-connected** networks

$$\begin{aligned} \mathbf{x}: & N \times D \\ \text{Normalize} & \downarrow \\ \boldsymbol{\mu}, \sigma: & 1 \times D \\ \gamma, \beta: & 1 \times D \\ \mathbf{y} = \gamma(\mathbf{x} - \boldsymbol{\mu}) / \sigma + \beta & \end{aligned}$$

Batch Normalization for  
**convolutional** networks  
(Spatial Batchnorm, BatchNorm2D)

$$\begin{aligned} \mathbf{x}: & N \times C \times H \times W \\ \text{Normalize} & \downarrow \quad \downarrow \quad \downarrow \\ \boldsymbol{\mu}, \sigma: & 1 \times C \times 1 \times 1 \\ \gamma, \beta: & 1 \times C \times 1 \times 1 \\ \mathbf{y} = \gamma(\mathbf{x} - \boldsymbol{\mu}) / \sigma + \beta & \end{aligned}$$

Все скрины спёр у Стэнфорда, влом было набирать

# Layer Normalization (2016)

## Layer Normalization

Batch Normalization for  
fully-connected networks

Layer Normalization for  
fully-connected networks  
Same behavior at train and test!  
Can be used in recurrent networks

$\mathbf{x}: N \times D$

Normalize



$\mu, \sigma: 1 \times D$

$\gamma, \beta: 1 \times D$

$$\mathbf{y} = \gamma(\mathbf{x} - \mu) / \sigma + \beta$$

$\mathbf{x}: N \times D$

Normalize



$\mu, \sigma: N \times 1$

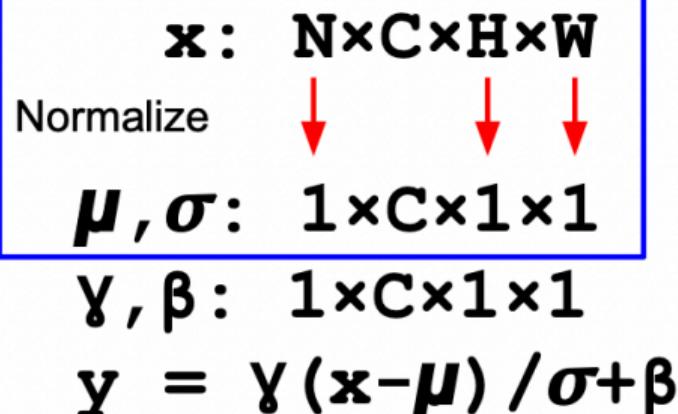
$\gamma, \beta: 1 \times D$

$$\mathbf{y} = \gamma(\mathbf{x} - \mu) / \sigma + \beta$$

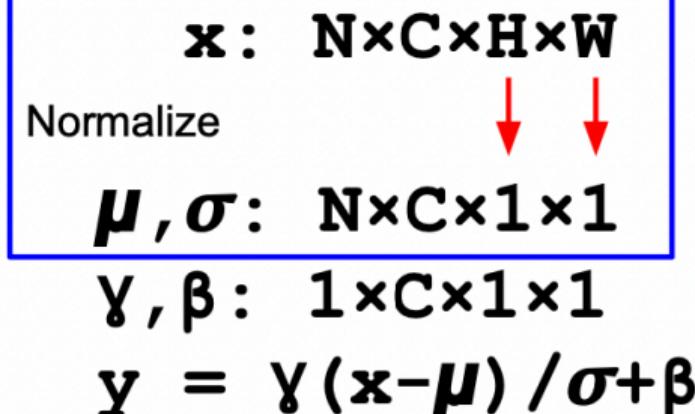
# Instance Normalization (2017)

## Instance Normalization

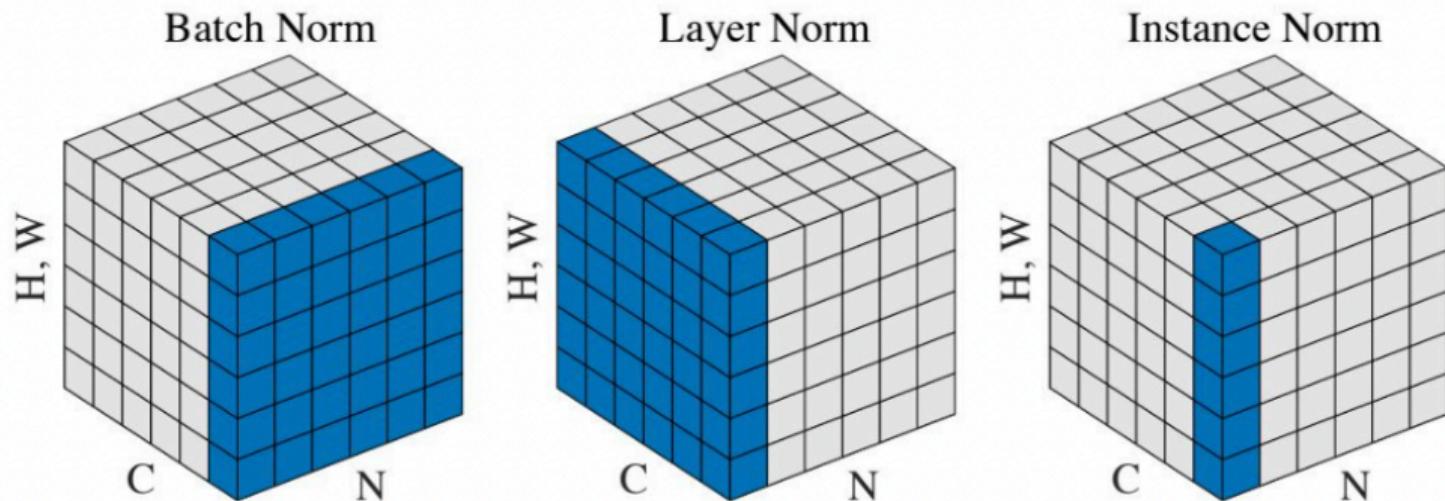
**Batch Normalization** for  
convolutional networks



**Instance Normalization** for  
convolutional networks  
Same behavior at train / test!



# Сравнение слоёв для нормализации



# Почему это помогает при обучении

---

## How Does Batch Normalization Help Optimization?

---

Shibani Santurkar\*  
MIT  
shibani@mit.edu

Dimitris Tsipras\*  
MIT  
tsipras@mit.edu

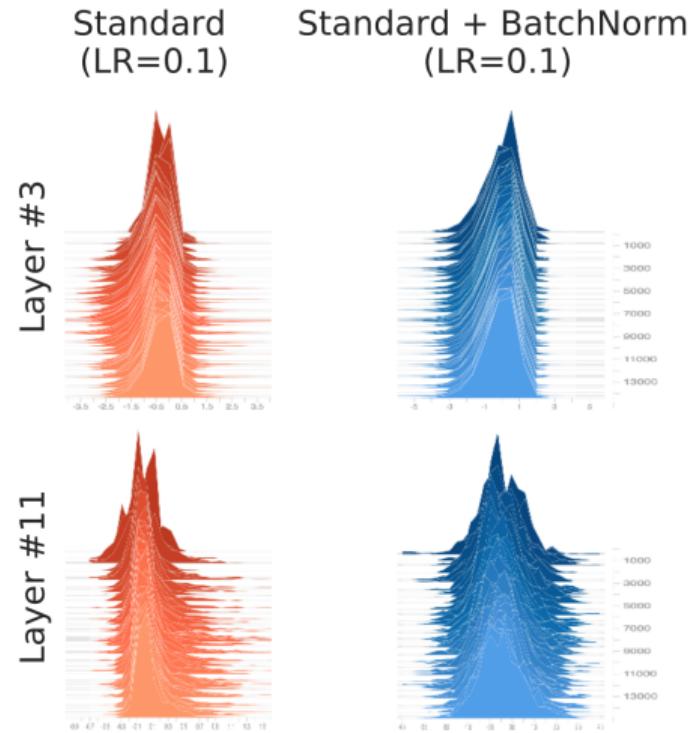
Andrew Ilyas\*  
MIT  
ailyas@mit.edu

Aleksander Mądry  
MIT  
madry@mit.edu

### Abstract

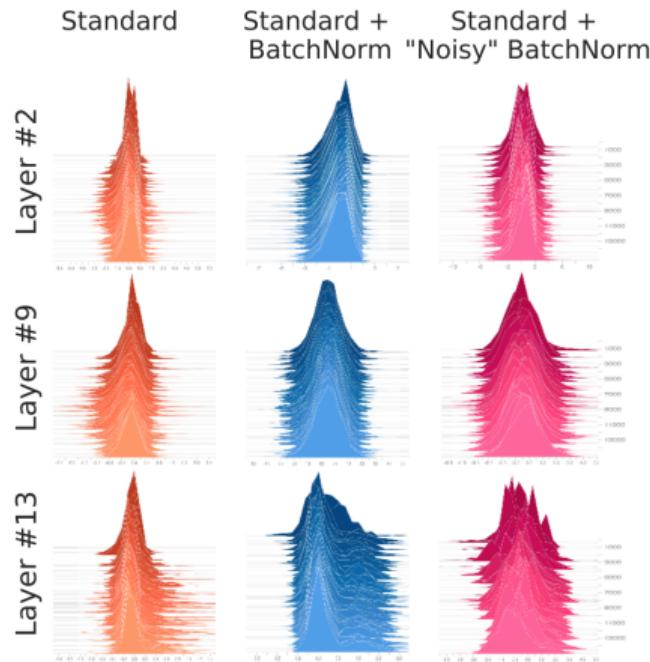
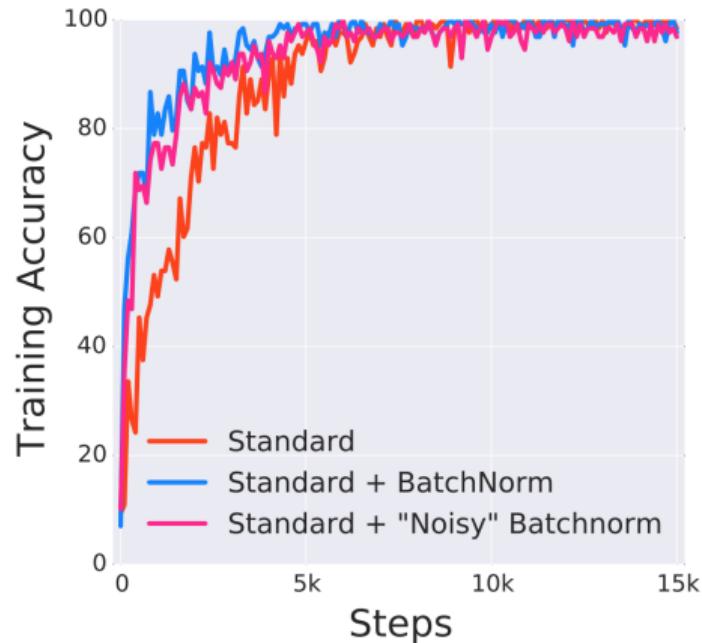
Batch Normalization (BatchNorm) is a widely adopted technique that enables faster and more stable training of deep neural networks (DNNs). Despite its pervasiveness, the exact reasons for BatchNorm's effectiveness are still poorly understood. The popular belief is that this effectiveness stems from controlling the change of the layers' input distributions during training to reduce the so-called "internal covariate shift". In this work, we demonstrate that such distributional stability of layer inputs has little to do with the success of BatchNorm. Instead, we uncover a more fundamental impact of BatchNorm on the training process: it makes the optimization landscape significantly smoother. This smoothness induces a more predictive and stable behavior of the gradients, allowing for faster training.

# Почему это помогает при обучении



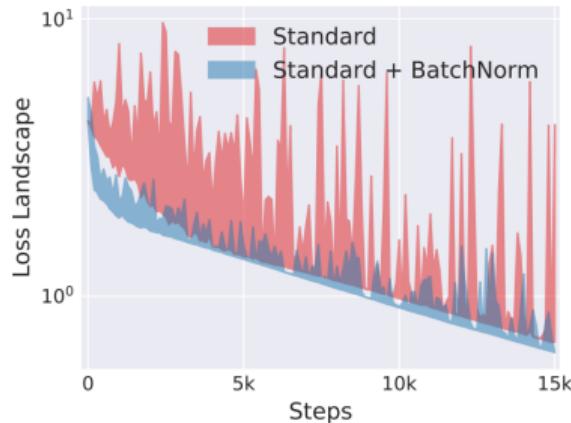
<https://arxiv.org/abs/1805.11604>

# Почему это помогает при обучении

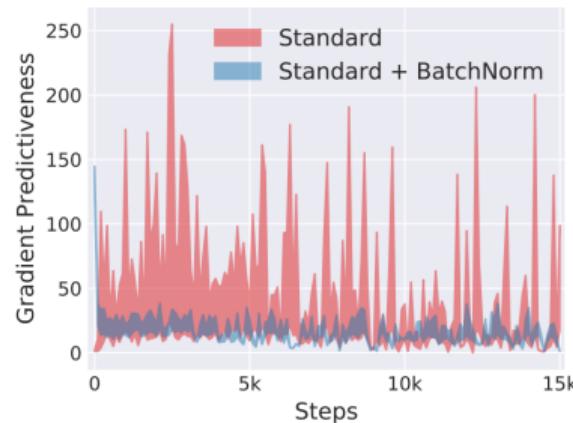


<https://arxiv.org/abs/1805.11604>

# Почему это помогает при обучении



(a) loss landscape



(b) gradient predictiveness

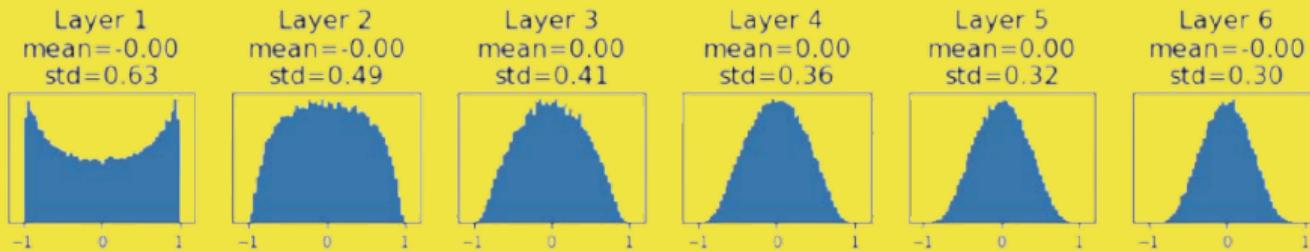
Батчнорм сглаживает ландшафт функции потерь и из-за этого градиентный спуск идёт более гладко.

## Трюки

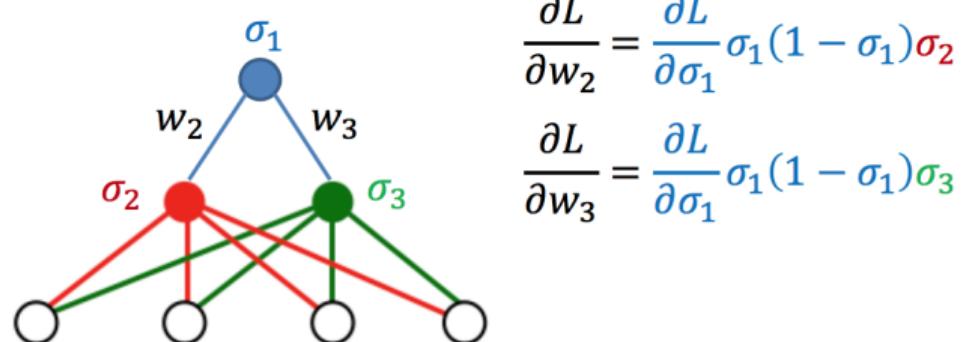
- С батч-нормализацией нужно уменьшить силу Dropout и регуляризацию
- Батч-нормализация и Dropout могут конфликтовать
- Не забывайте перемешивать обучающую выборку перед каждой новой эпохой, чтобы батчи были разнообразными
- Существует довольно много техник нормализации: Layer Normalization, Weight Normalization, Batch Renormalization, Adaptive Instance Normalization, Group Normalization etc.

[http://openaccess.thecvf.com/content\\_CVPR\\_2019/papers/Li\\_Understanding\\_the\\_Discrepancy\\_Between\\_Dropout\\_and\\_Batch\\_Normalization\\_by\\_Variance\\_CVPR\\_2019\\_paper.pdf](http://openaccess.thecvf.com/content_CVPR_2019/papers/Li_Understanding_the_Discrepancy_Between_Dropout_and_Batch_Normalization_by_Variance_CVPR_2019_paper.pdf)

# Инициализация весов



# Инициализация весов

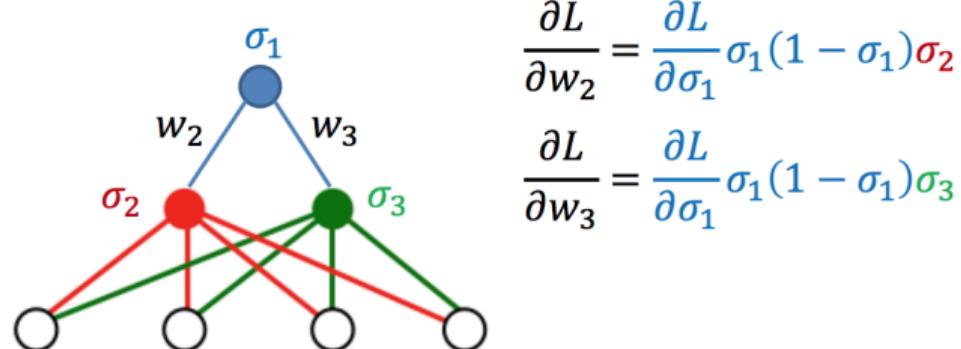


$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial \sigma_1} \sigma_1 (1 - \sigma_1) \color{red}{\sigma_2}$$

$$\frac{\partial L}{\partial w_3} = \frac{\partial L}{\partial \sigma_1} \sigma_1 (1 - \sigma_1) \color{red}{\sigma_3}$$

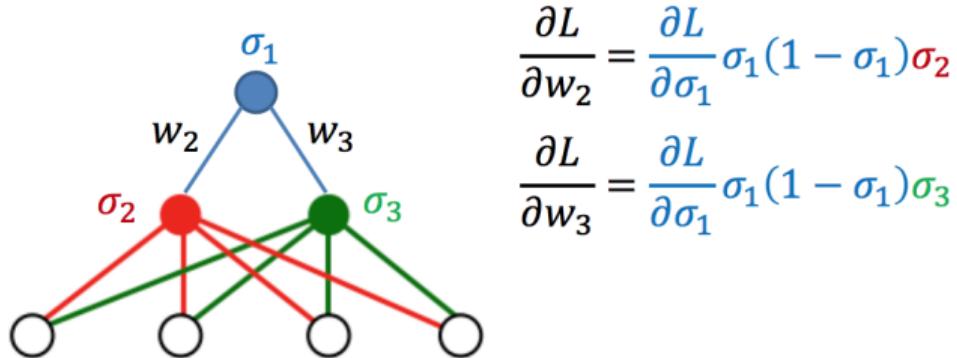
- Что будет, если инициализировать веса нулями?

# Инициализация весов



- Что будет, если инициализировать веса нулями?
- $\sigma_2$  и  $\sigma_3$  будут обновляться одинаково

# Инициализация весов



- Хочется уничтожить симметрию
- Можно инициализировать маленькими рандомными числами из какого-то распределения (нормальное, равномерное)
- Это будет плохо работать для глубоких сетей

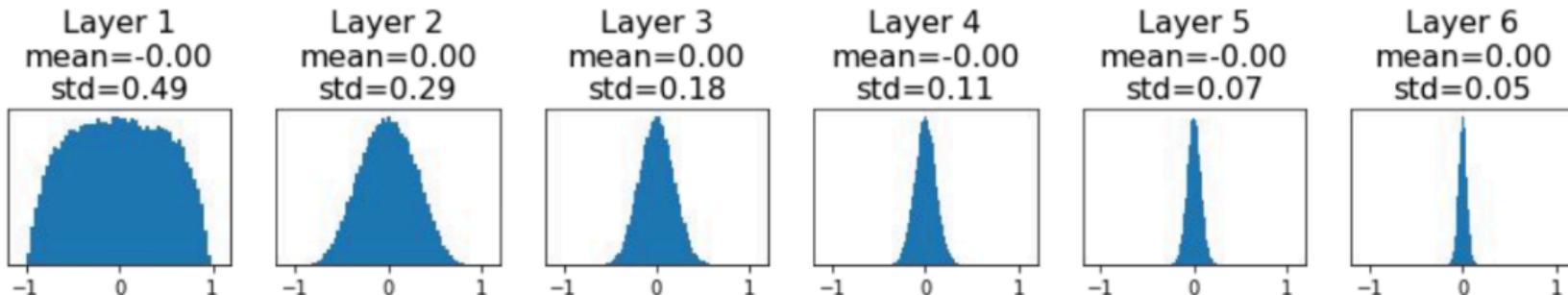
# Инициализация весов

- Прямой проход через 6 слоёв нейросети
- Что произойдёт с активацией к последнему слою?

```
dims = [4096] * 7
hs = []
x = np.random.randn( 16, dims[ 0 ] )
for Din, Dout in zip(dims[: -1], dims[ 1 : ]):
    W = 0.01 * np.random.randn(Din, Dout)
    x = np.tanh(x.dot(W))
    hs.append(x)
```

# Инициализация весов

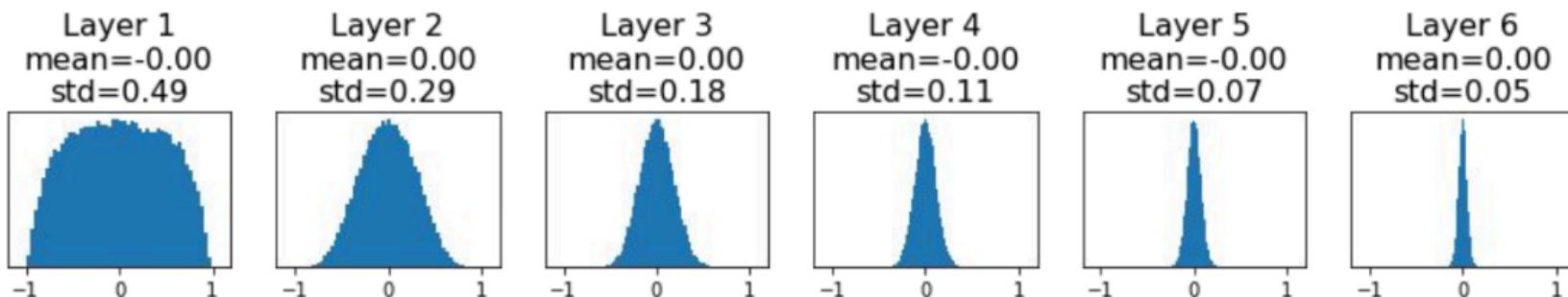
- Прямой проход через 6 слоёв нейросети
  - Что произойдёт с активацией к последнему слою?
  - Активация постепенно стягивается к нулю
- ```
dims = [4096] * 7
hs = []
x = np.random.randn( 16, dims[ 0 ])
for Din, Dout in zip(dims[: -1], dims[ 1:]):
    W = 0.01 * np.random.randn(Din, Dout)
    x = np.tanh(x.dot(W))
    hs.append(x)
```



# Инициализация весов

- Активация постепенно стягивается к нулю
- А что будет происходить с градиентами?

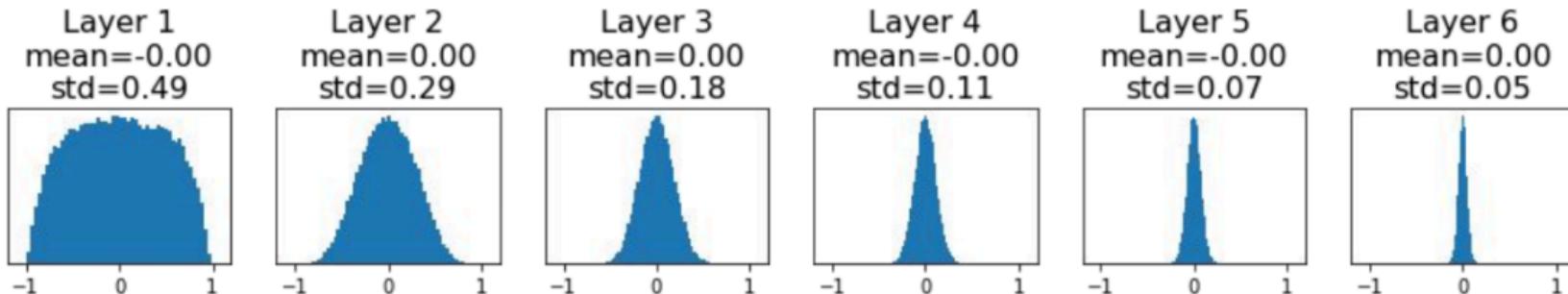
```
dims = [4096] * 7
hs = []
x = np.random.randn( 16, dims[ 0 ])
for Din, Dout in zip(dims[: -1], dims[ 1 :]):
    W = 0.01 * np.random.randn(Din, Dout)
    x = np.tanh(x.dot(W))
    hs.append(x)
```



# Инициализация весов

- Активация постепенно стягивается к нулю
- А что будет происходить с градиентами?
- Все будут нулевыми :(

```
dims = [4096] * 7
hs = []
x = np.random.randn( 16, dims[ 0 ] )
for Din, Dout in zip(dims[: -1], dims[ 1 :]):
    W = 0.01 * np.random.randn(Din, Dout)
    x = np.tanh(x.dot(W))
    hs.append(x)
```



# Инициализация весов

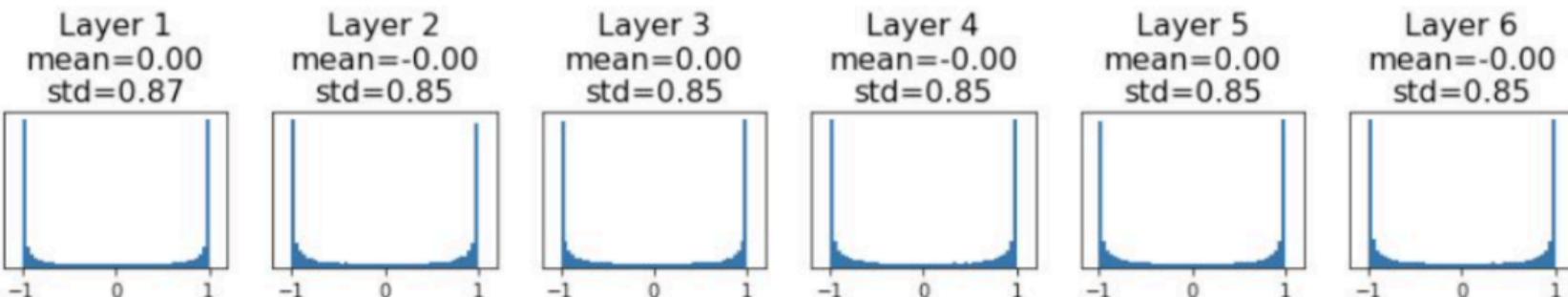
- Увеличим стандартное отклонение в инициализации с 0.01 до 0.05
- Что произойдёт с активацией к последнему слою?

```
dims = [4096] * 7
hs = []
x = np.random.randn( 16, dims[ 0 ] )
for Din, Dout in zip(dims[: -1], dims[ 1 :]):  
    W = 0.05 * np.random.randn(Din, Dout)
    x = np.tanh(x.dot(W))
    hs.append(x)
```

# Инициализация весов

- Постепенно происходит насыщение, градиенты опять занулятся

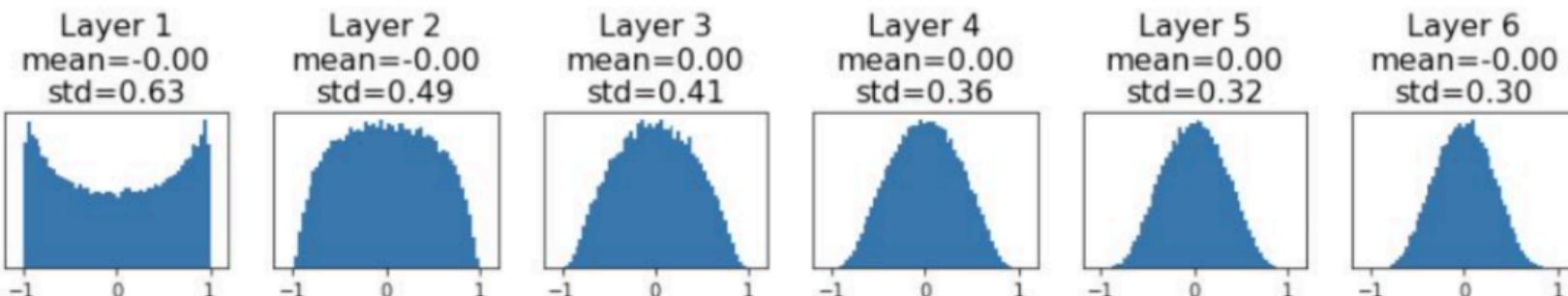
```
dims = [4096] * 7
hs = []
x = np.random.randn( 16, dims[ 0 ])
for Din, Dout in zip(dims[ : -1 ], dims[ 1 : ]):
    W = 0.05 * np.random.randn(Din, Dout)
    x = np.tanh(x.dot(W))
    hs.append(x)
```



# Инициализация весов

- Распределение активаций не меняется, с градиентами всё хорошо

```
dims = [4096] * 7
hs = []
x = np.random.randn( 16, dims[ 0 ] )
for Din, Dout in zip(dims[: -1], dims[ 1 : ]):
    W = 1/np.sqrt(Din) * np.random.randn(Din, Dout)
    x = np.tanh(x.dot(W))
    hs.append(x)
```



# Инициализация весов

- Посмотрим на выход нейрона перед активацией:

$$h_i = \sum_{i=1}^{n_{in}} w_i x_i$$

- Дисперсия  $h_i$  выражается через дисперсии  $x$  и  $w$
- Будем считать, что веса  $w_1, \dots, w_k \sim iid$ , наблюдения  $x_1, \dots, x_n \sim iid$ , а ещё  $x_i$  и  $w_i$  независимы между собой

## Инициализация весов (симметричная активация)

$$\begin{aligned}\text{Var}(h_i) &= \text{Var}\left(\sum_{i=1}^{n_{in}} w_i x_i\right) = \sum_{i=1}^{n_{in}} \text{Var}(w_i x_i) = \\ &= \sum_{i=1}^{n_{in}} [\mathbb{E}(x_i)]^2 \cdot \text{Var}(w_i) + [\mathbb{E}(w_i)]^2 \cdot \text{Var}(x_i) + \text{Var}(x_i) \cdot \text{Var}(w_i) =\end{aligned}$$

[https://en.wikipedia.org/wiki/Variance#Product\\_of\\_independent\\_variables](https://en.wikipedia.org/wiki/Variance#Product_of_independent_variables)

## Инициализация весов (симметричная активация)

$$\begin{aligned}\text{Var}(h_i) &= \text{Var}\left(\sum_{i=1}^{n_{in}} w_i x_i\right) = \sum_{i=1}^{n_{in}} \text{Var}(w_i x_i) = \\ &= \sum_{i=1}^{n_{in}} [\mathbb{E}(x_i)]^2 \cdot \text{Var}(w_i) + [\mathbb{E}(w_i)]^2 \cdot \text{Var}(x_i) + \text{Var}(x_i) \cdot \text{Var}(w_i) =\end{aligned}$$

- Если функция активации симметричная, тогда  $\mathbb{E}(x_i) = 0$ . Будем инициализировать веса с нулевым средним, тогда  $\mathbb{E}(w_i) = 0$ .

## Инициализация весов (симметричная активация)

$$\begin{aligned}\text{Var}(h_i) &= \text{Var}\left(\sum_{i=1}^{n_{in}} w_i x_i\right) = \sum_{i=1}^{n_{in}} \text{Var}(w_i x_i) = \\ &= \sum_{i=1}^{n_{in}} [\mathbb{E}(x_i)]^2 \cdot \text{Var}(w_i) + [\mathbb{E}(w_i)]^2 \cdot \text{Var}(x_i) + \text{Var}(x_i) \cdot \text{Var}(w_i) = \\ &= \sum_{i=1}^{n_{in}} \text{Var}(x_i) \cdot \text{Var}(w_i)\end{aligned}$$

- Предполагаем, что  $x_i$  инициализируются из одного распределения,  $w_i$  сами инициализируем из одного распределения.

## Инициализация весов (симметричная активация)

$$\begin{aligned}\text{Var}(h_i) &= \text{Var}\left(\sum_{i=1}^{n_{in}} w_i x_i\right) = \sum_{i=1}^{n_{in}} \text{Var}(w_i x_i) = \\ &= \sum_{i=1}^{n_{in}} [\mathbb{E}(x_i)]^2 \cdot \text{Var}(w_i) + [\mathbb{E}(w_i)]^2 \cdot \text{Var}(x_i) + \text{Var}(x_i) \cdot \text{Var}(w_i) = \\ &= \sum_{i=1}^{n_{in}} \text{Var}(x_i) \cdot \text{Var}(w_i) = \text{Var}(x) \cdot [n_{in} \cdot \text{Var}(w)]\end{aligned}$$

- Хотим, чтобы зелёная штука была равна единице, тогда у потока будет всегда постоянная дисперсия, совпадающая с  $\text{Var}(x)$ .

## Инициализация весов (симметричная активация)

$$\begin{aligned}\mathbf{Var}(h_i) &= \mathbf{Var}\left(\sum_{i=1}^{n_{in}} w_i x_i\right) = \sum_{i=1}^{n_{in}} \mathbf{Var}(w_i x_i) = \\ &= \sum_{i=1}^{n_{in}} [\mathbf{E}(x_i)]^2 \cdot \mathbf{Var}(w_i) + [\mathbf{E}(w_i)]^2 \cdot \mathbf{Var}(x_i) + \mathbf{Var}(x_i) \cdot \mathbf{Var}(w_i) = \\ &= \sum_{i=1}^{n_{in}} \mathbf{Var}(x_i) \cdot \mathbf{Var}(w_i) = \mathbf{Var}(x) \cdot \underbrace{[n_{in} \cdot \mathbf{Var}(w)]}_{=1}\end{aligned}$$

# Плохая инициализация весов

Пусть

$$w_i \sim U \left[ -\frac{1}{\sqrt{n_{in}}}; \frac{1}{\sqrt{n_{in}}} \right],$$

тогда

$$\text{Var}(w_i) = \frac{1}{12} \cdot \left( \frac{1}{\sqrt{n_{in}}} + \frac{1}{\sqrt{n_{in}}} \right)^2 = \frac{1}{3n_{in}} \Rightarrow \text{Var}(h_i) = \frac{1}{3}$$

Получаем затухание!

# Инициализация Ксавье/Глорота (2010)

Пусть

$$w_i \sim U \left[ -\frac{\sqrt{3}}{\sqrt{n_{in}}}; \frac{\sqrt{3}}{\sqrt{n_{in}}} \right],$$

тогда

$$\text{Var}(w_i) = \frac{1}{12} \cdot \left( \frac{\sqrt{3}}{\sqrt{n_{in}}} + \frac{\sqrt{3}}{\sqrt{n_{in}}} \right)^2 = \frac{1}{n_{in}} \Rightarrow \text{Var}(h_i) = 1$$

## Инициализация Ксавье/Глорота (2010)

Пусть

$$w_i \sim U \left[ -\frac{\sqrt{3}}{\sqrt{n_{in}}}; \frac{\sqrt{3}}{\sqrt{n_{in}}} \right],$$

тогда

$$\text{Var}(w_i) = \frac{1}{12} \cdot \left( \frac{\sqrt{3}}{\sqrt{n_{in}}} + \frac{\sqrt{3}}{\sqrt{n_{in}}} \right)^2 = \frac{1}{n_{in}} \Rightarrow \text{Var}(h_i) = 1$$

При forward pass на вход идёт  $n_{in}$  наблюдений, при backward pass на вход идёт  $n_{out}$  градиентов  $\Rightarrow$  канал с дисперсией может быть непостоянным, если число весов от слоя к слою сильно колеблется

# Инициализация Ксавье/Глорота (2010)

- Для неодинаковых размеров слоёв невозможно удовлетворить обоим условиям, поэтому обычно усредняют и пытаются инициализировать веса с дисперсией

$$\frac{2}{n_{in} + n_{out}}$$

- Такая инициализация называется **инициализацией Ксавье (или Глорота)**

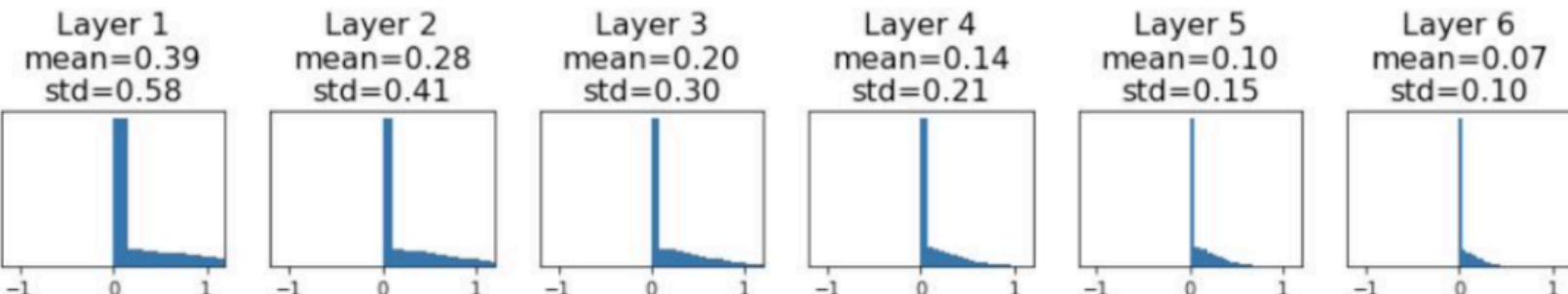
$$w_i \sim U \left[ -\frac{\sqrt{6}}{\sqrt{n_{out} + n_{in}}}; \frac{\sqrt{6}}{\sqrt{n_{out} + n_{in}}} \right],$$

<http://proceedings.mlr.press/v9/glorot10a/glorot10a.pdf>

# А что с ReLU?

Инициализация  
Ксавье предполагает  
симметричную  
функцию активации

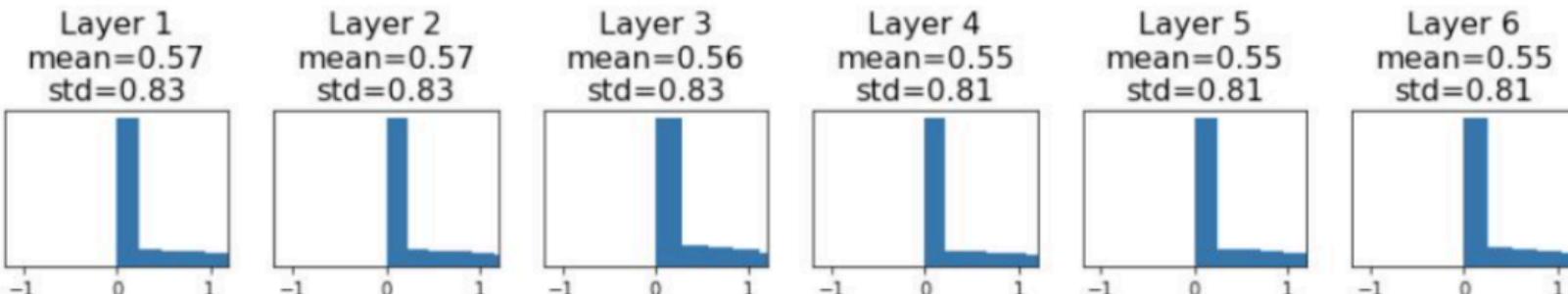
```
dims = [4096] * 7
hs = []
x = np.random.randn( 16, dims[ 0 ] )
for Din, Dout in zip(dims[ : -1 ], dims[ 1 : ]):
    W = 1/np.sqrt(Din) * np.random.randn(Din, Dout)
    x = np.max(0, x.dot(W))
    hs.append(x)
```



# А что с ReLU?

Инициализация  
Ксавье предполагает  
симметричную  
функцию активации

```
dims = [4096] * 7
hs = []
x = np.random.randn( 16, dims[ 0 ] )
for Din, Dout in zip(dims[: -1], dims[ 1 :]):
    W = np.sqrt(2/Din) * np.random.randn(Din, Dout)
    x = np.max(0, x.dot(W))
    hs.append(x)
```



# Инициализация Xe (2015)

$$\begin{aligned}\text{Var}(h_i) &= \text{Var}\left(\sum_{i=1}^{n_{in}} w_i x_i\right) = \sum_{i=1}^{n_{in}} \text{Var}(w_i x_i) = \\ &= \sum_{i=1}^{n_{in}} [\mathbb{E}(x_i)]^2 \cdot \text{Var}(w_i) + [\mathbb{E}(w_i)]^2 \cdot \text{Var}(x_i) + \text{Var}(x_i) \cdot \text{Var}(w_i)\end{aligned}$$

- Когда нет симметрии, можно занулить только второе слагаемое

# Инициализация Xe (2015)

$$\begin{aligned}\text{Var}(h_i) &= \text{Var}\left(\sum_{i=1}^{n_{in}} w_i x_i\right) = \sum_{i=1}^{n_{in}} \text{Var}(w_i x_i) = \\ &= \sum_{i=1}^{n_{in}} [\mathbb{E}(x_i)]^2 \cdot \text{Var}(w_i) + [\mathbb{E}(w_i)]^2 \cdot \text{Var}(x_i) + \text{Var}(x_i) \cdot \text{Var}(w_i) = \\ &= \sum_{i=1}^{n_{in}} [\mathbb{E}(x_i)]^2 \cdot \text{Var}(w_i) + \text{Var}(x_i) \cdot \text{Var}(w_i) = \sum_{i=1}^{n_{in}} \text{Var}(w_i) \cdot E(x_i^2)\end{aligned}$$

- Когда нет симметрии, можно занулить только второе слагаемое

# Инициализация Xe (2015)

$$\begin{aligned}\mathbf{Var}(h_i) &= \mathbf{Var}\left(\sum_{i=1}^{n_{in}} w_i x_i\right) = \sum_{i=1}^{n_{in}} \mathbf{Var}(w_i x_i) = \\ &= \sum_{i=1}^{n_{in}} [\mathbf{E}(x_i)]^2 \cdot \mathbf{Var}(w_i) + [\mathbf{E}(w_i)]^2 \cdot \mathbf{Var}(x_i) + \mathbf{Var}(x_i) \cdot \mathbf{Var}(w_i) = \\ &= \sum_{i=1}^{n_{in}} [\mathbf{E}(x_i)]^2 \cdot \mathbf{Var}(w_i) + \mathbf{Var}(x_i) \cdot \mathbf{Var}(w_i) = \sum_{i=1}^{n_{in}} \mathbf{Var}(w_i) \cdot E(x_i^2) = \\ &= E(x^2) \cdot [n_{in} \cdot \mathbf{Var}(w)]\end{aligned}$$

# Инициализация Xe (2015)

$$\begin{aligned}\mathsf{Var}(h_i) &= E(x_i^2) \cdot [n_{in} \cdot \mathsf{Var}(w)] \\ x_i &= \max(0; h_{i-1})\end{aligned}$$

# Инициализация Xe (2015)

$$\begin{aligned}\text{Var}(h_i) &= E(x_i^2) \cdot [n_{in} \cdot \text{Var}(w)] \\ x_i &= \max(0; h_{i-1})\end{aligned}$$

Пусть  $w_{i-1}$  распределены симметрично относительно нуля, тогда  $h_{i-1}$  тоже будут симметрично распределены, тогда:

# Инициализация Xe (2015)

$$\begin{aligned}\text{Var}(h_i) &= E(x_i^2) \cdot [n_{in} \cdot \text{Var}(w)] \\ x_i &= \max(0; h_{i-1})\end{aligned}$$

Пусть  $w_{i-1}$  распределены симметрично относительно нуля, тогда  $h_{i-1}$  тоже будут симметрично распределены, тогда:

$$E(x_i^2) = \frac{1}{2} \cdot \text{Var}(h_{i-1})$$

# Инициализация Xe (2015)

$$\begin{aligned}\text{Var}(h_i) &= E(x_i^2) \cdot [n_{in} \cdot \text{Var}(w)] \\ x_i &= \max(0; h_{i-1})\end{aligned}$$

Пусть  $w_{i-1}$  распределены симметрично относительно нуля, тогда  $h_{i-1}$  тоже будут симметрично распределены, тогда:

$$E(x_i^2) = \frac{1}{2} \cdot \text{Var}(h_{i-1})$$

$$\text{Var}(h_i) = \frac{1}{2} \cdot \text{Var}(h_{i-1}) \cdot [n_{in} \cdot \text{Var}(w)] \Rightarrow \text{Var}(w_i) = \frac{2}{n_{in}}$$

## TLDR: что на практике

- Для симметричных функций с нулевым средним используйте инициализацию Ксавье (Глорота)
- Для ReLU и им подобным инициализацию Хе
- Эти две инициализации корректируют параметры распределений в зависимости от входа и выхода слоя так, чтобы поддерживать дисперсию равной единице

# Тема инициализации сегодня активно исследуется

- Understanding the difficulty of training deep feedforward neural networks by Glorot and Bengio, 2010
- Exact solutions to the nonlinear dynamics of learning in deep linear neural networks by Saxe et al, 2013
- Random walk initialization for training very deep feedforward networks by Sussillo and Abbott, 2014
- Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification by He et al., 2015
- Data-dependent Initializations of Convolutional Neural Networks by Krähenbühl et al., 2015
- All you need is a good init, Mishkin and Matas, 2015
- Fixup Initialization: Residual Learning Without Normalization, Zhang et al, 2019
- The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks, Frankle and Carbin, 2019

Попробуем новые трюки на  
практике?