

Чуть глубже в нейросети

ФКН

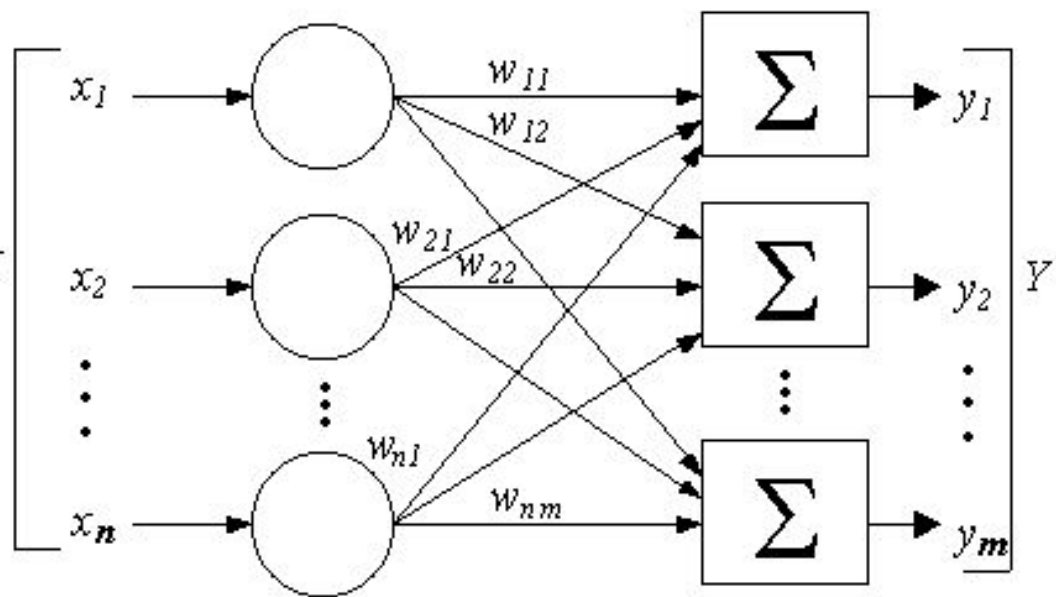
План лекции

- Повторение линейного слоя и нелинейностей
- Инициализация весов сети
 - Инициализация константой
 - Нормальное распределение
 - Xavier
 - Kaiming
- ML common knowledge
- Методы регуляризации сети
 - Модификации функций ошибки
 - Слои нейросетей
 - Оптимизация
 - Работа с данными

Повторение*

Линейный слой

- Матрица весов: $W = \begin{bmatrix} w_{0;1} & \dots & w_{0;m} \\ \vdots & \ddots & \vdots \\ w_{d;1} & \dots & w_{d;m} \end{bmatrix} \in \mathbb{R}^{(d+1) \times m}$.
- $\sigma(z_1, \dots, z_m) \sim (\sigma(z_1), \dots, \sigma(z_m))$.
- Тогда $y = \sigma(x^T * W)$.



Два линейных слоя подряд

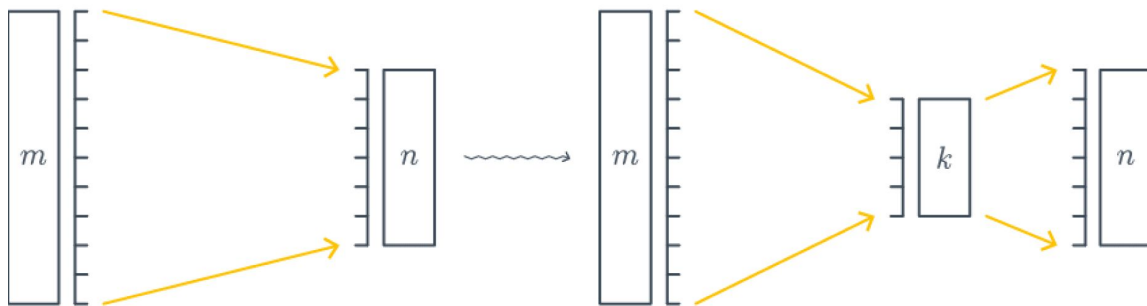
Линейная комбинация линейных отображений есть линейное отображение, то есть два последовательных линейных слоя эквивалентны одному линейному слою.

Но на самом деле бывают ситуации, когда два линейных слоя подряд — это полезно.



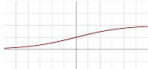
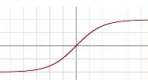

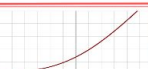


Два линейных слоя подряд

Например, если вы понимаете, что у вас очень много параметров, а информации в данных не так много, вы можете заменить линейный слой, превращающий m -мерные векторы в n -мерные, на два, вставив посередине k -мерное представление, где $k \ll m, n$:

С точки зрения линейной алгебры это примерно то же самое, что потребовать, чтобы матрица исходного линейного слоя имела ранг не выше k . И с точки зрения сужения «информационного канала» это иногда может сработать. Но в любом случае вы должны понимать, что два линейных слоя подряд стоит ставить, только если вы хорошо понимаете, чего хотите добиться.



Нелийности

ACTIVATION FUNCTION	PLOT	EQUATION	DERIVATIVE	RANGE
Linear		$f(x) = x$	$f'(x) = 1$	$(-\infty, \infty)$
Binary Step		$f(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{if } x \neq 0 \\ \text{undefined} & \text{if } x = 0 \end{cases}$	$\{0, 1\}$
Sigmoid		$f(x) = \sigma(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$	$(0, 1)$
Hyperbolic Tangent(tanh)		$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	$f'(x) = 1 - f(x)^2$	$(-1, 1)$
Rectified Linear Unit(ReLU)		$f(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x > 0 \\ \text{undefined} & \text{if } x = 0 \end{cases}$	$[0, \infty)$
Softplus		$f(x) = \ln(1 + e^x)$	$f'(x) = \frac{1}{1 + e^{-x}}$	$(0, 1)$
Leaky ReLU		$f(x) = \begin{cases} 0.01x & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0.01 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$	$(-1, 1)$
Exponential Linear Unit(ELU)		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}$	$f'(x) = \begin{cases} \alpha e^x & \text{if } x < 0 \\ 1 & \text{if } x > 0 \\ 1 & \text{if } x = 0 \text{ and } \alpha = 1 \end{cases}$	$[0, \infty)$

SoftMax

Output
layer

$$\begin{bmatrix} 1.3 \\ 5.1 \\ 2.2 \\ 0.7 \\ 1.1 \end{bmatrix}$$

Softmax
activation function

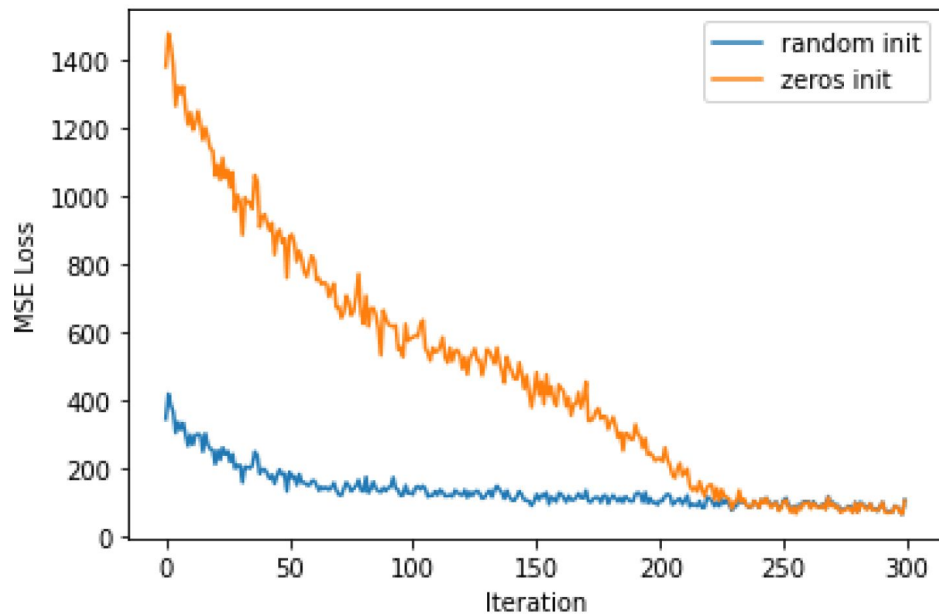
$$\frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}$$

Probabilities

$$\begin{bmatrix} 0.02 \\ 0.90 \\ 0.05 \\ 0.01 \\ 0.02 \end{bmatrix}$$

Инициализация весов

Инициализация константой



Веса будут меняться одинаково внутри слоёв

Xavier

$$y = \mathbf{w}^\top \mathbf{x} + b = \sum_i w_i x_i + b.$$

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

$$\mathrm{Var}(y_i) = \mathrm{Var}(w_i) \mathrm{Var}(x_i)$$

$$\mathrm{Var}(y) = \mathrm{Var} \left(\sum_{i=1}^{n_{\mathrm{out}}} y_i \right) = \sum_{i=1}^{n_{\mathrm{out}}} \mathrm{Var}(w_i x_i) = n_{\mathrm{out}} \mathrm{Var}(w_i) \mathrm{Var}(x_i)$$

$$w_i \sim U \left[-\frac{1}{\sqrt{n_{\mathrm{out}}}}, \frac{1}{\sqrt{n_{\mathrm{out}}}} \right] \quad \mathrm{Var}(w_i) = \frac{1}{12} \left(\frac{1}{\sqrt{n_{\mathrm{out}}}} + \frac{1}{\sqrt{n_{\mathrm{out}}}} \right)^2 = \frac{1}{3n_{\mathrm{out}}}, \quad \text{и} \quad n_{\mathrm{out}} \mathrm{Var}(w_i) = \frac{1}{3}$$

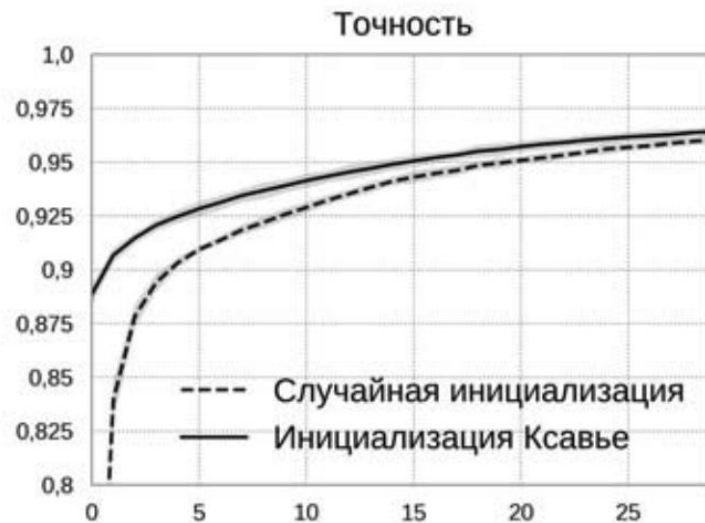
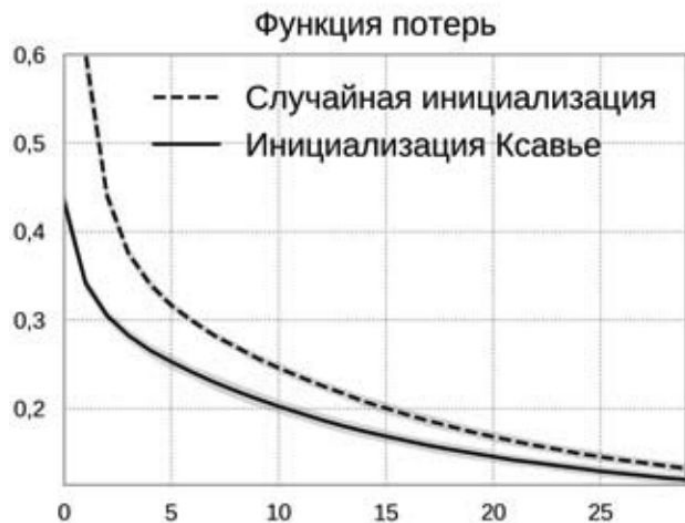
Xavier (backward pass)

$$\frac{\partial L}{\partial y_i^{(l)}} = f'(y_i^{(l)}) \sum_j w_{i,j}^{(l+1)} \frac{\partial L}{\partial y_j^{(l+1)}}$$

$$\text{Var}(w_i) = \frac{2}{n_{\text{in}} + n_{\text{out}}}$$

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

Инициализация просто нормальным распределением



Kaiming (for ReLU)

$$\text{Var}(w_i x_i) = \mathbb{E}[x_i]^2 \text{Var}(w_i) + \mathbb{E}[w_i]^2 \text{Var}(x_i) + \text{Var}(w_i) \text{Var}(x_i)$$

$$\text{Var}(w_i x_i) = \mathbb{E}[x_i]^2 \text{Var}(w_i) + \text{Var}(w_i) \text{Var}(x_i) = \text{Var}(w_i) \mathbb{E}[x_i^2]$$

$$\text{Var}(y^{(l)}) = n_{\text{in}}^{(l)} \text{Var}(w^{(l)}) \mathbb{E}\left[(x^{(l)})^2\right] \quad \mathbb{E}\left[(x^{(l)})^2\right] = \frac{1}{2} \text{Var}(y^{(l-1)})$$

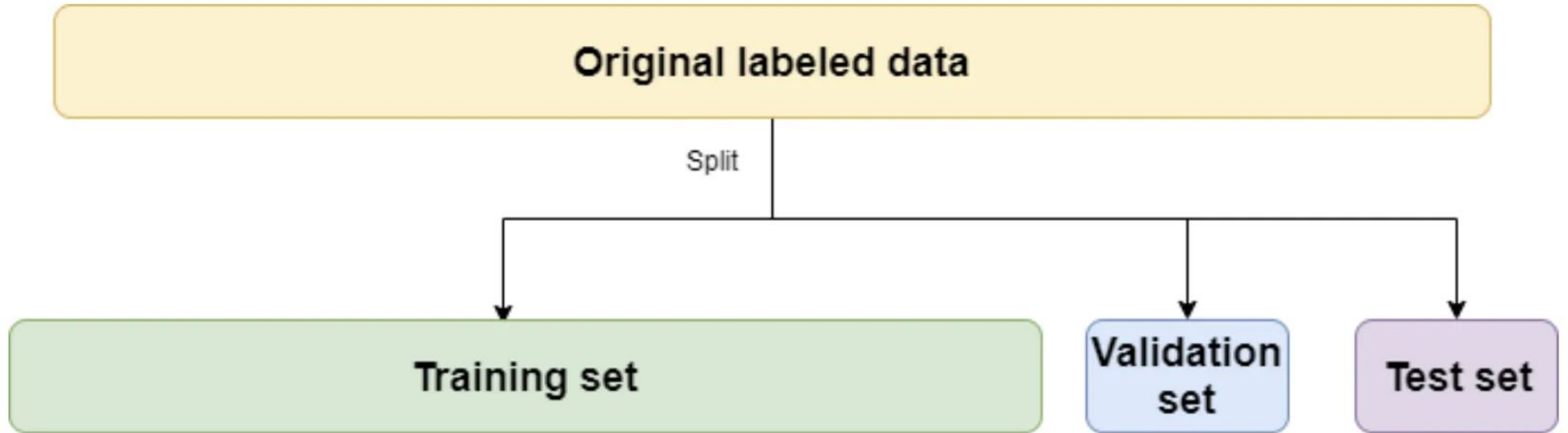
$$\text{Var}(y^{(l)}) = \frac{n_{\text{in}}^{(l)}}{2} \text{Var}(w^{(l)}) \text{Var}(y^{(l-1)})$$

$$\frac{n_{\text{in}}^{(l)}}{2} \text{Var}(w^{(l)}) = 1 \quad \text{и} \quad \text{Var}(w_i) = \frac{2}{n_{\text{in}}^{(l)}}$$

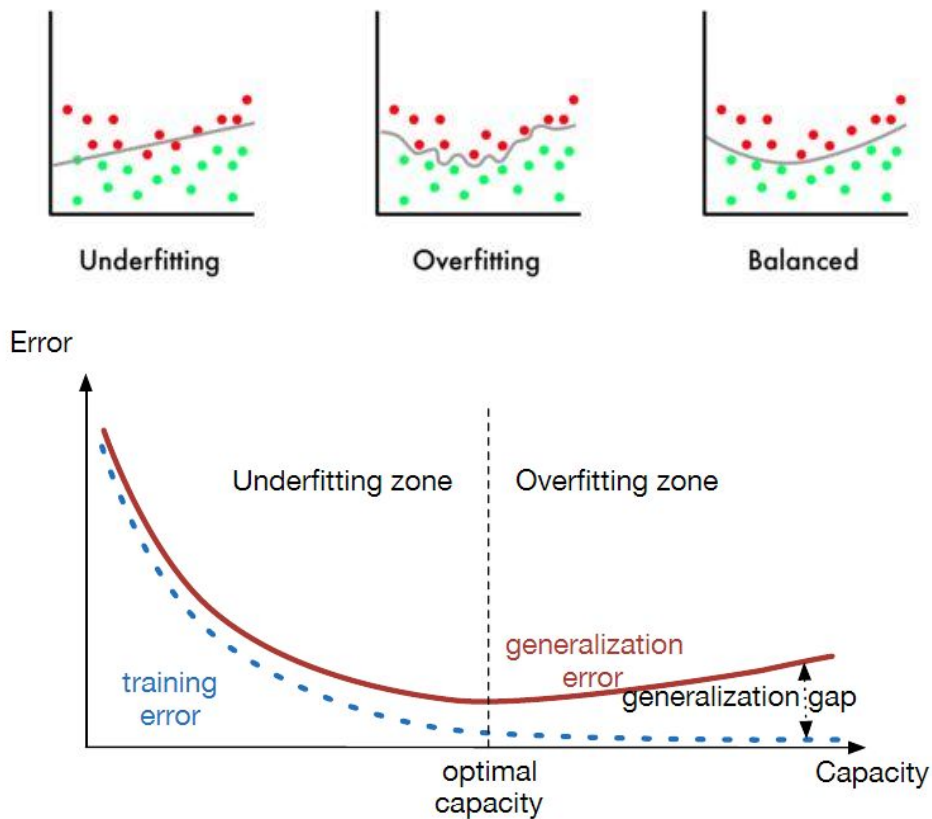
$$w_i \sim \mathcal{N}\left(0, \sqrt{\frac{2}{n_{\text{in}}^{(l)}}}\right)$$

ML common knowledge

Train, Validation, Test



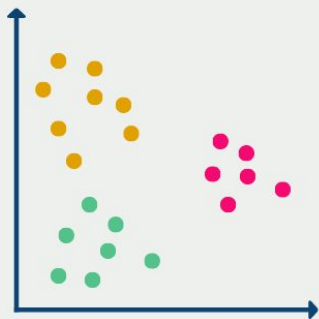
Переобучение



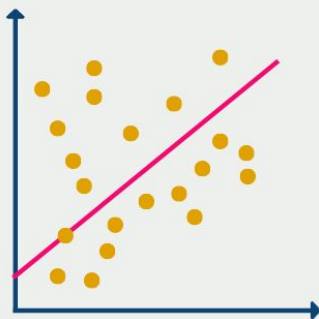
Виды обучений

Machine Learning

Unsupervised
Learning



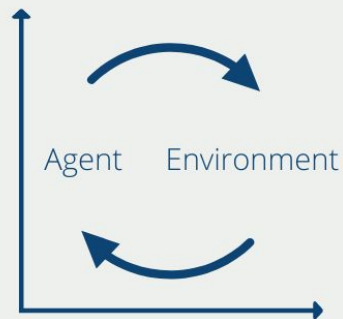
Supervised
Learning



Semi-Supervised
Learning



Reinforcement
Learning



Регуляризация нейросетей

Регуляризация нейросетей

Техники для борьбы с переобучением и для получения более подходящего решения с точки зрения эксперта. В нейронных сетях можно добиться изменением

- функции потерь
- структуры сети
- процесса оптимизации
- данных

Функции потерь

Модифицируем loss, по которому пускаем расчёт градиента

$$Loss_{result} = L_{original} + L_{regularization}$$

Добавка может быть любой, подходящей под задачу, но есть пара наиболее распространённых примеров

L2-регуляризация

- L2-норма на параметры

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^N \ell(f(x_i, \theta), y_i) + \frac{\lambda}{2} \|\theta\|_2^2$$

- Препятствует росту параметров
- Обычно встроена в оптимизатор (в SGD – это weight decay)
$$\theta_{t+1} \leftarrow \theta_t(1 - \lambda) - \gamma \nabla_{\theta} \ell_t(\theta_t)$$
- Обычные значения 10^{-3} , 10^{-4} (по умолчанию – 0)
- L1 – примерно такой же эффект

Модификации архитектуры

- Специальные слои (нормализации и DropOut)
- Дистилляция
- Квантизация
- Пруннинг

Нормализации

- BatchNorm
- LayerNorm
- InstanceNorm

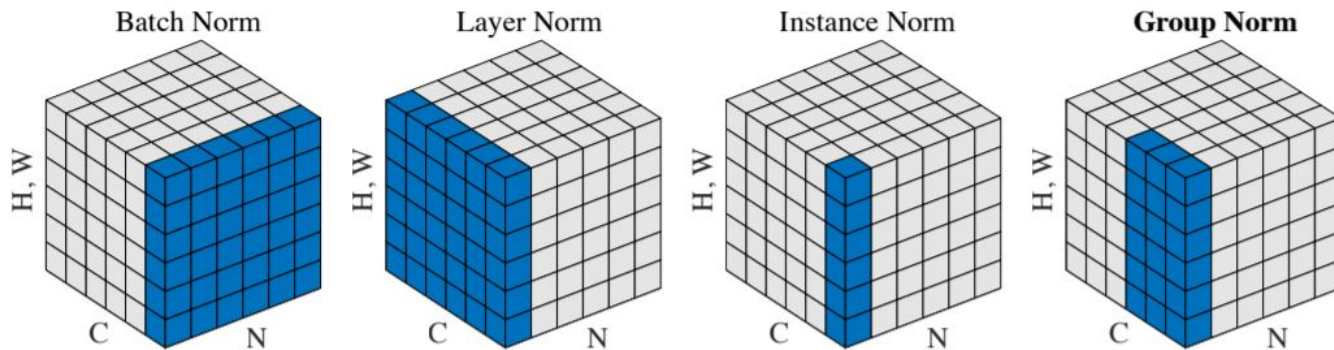


Figure 2. **Normalization methods.** Each subplot shows a feature map tensor, with N as the batch axis, C as the channel axis, and (H, W) as the spatial axes. The pixels in blue are normalized by the same mean and variance, computed by aggregating the values of these pixels.

BatchNorm

- Internal Covariate Shift = изменение распределения активаций, вызванное изменением параметров
- Whitening – поворот на i.i.d. с 0-mean и 1-std
 - Оценка матрицы ковариаций Σ , $W = \text{chol}(\Sigma^{-1})$, $x := W x$
 - Упрощение: вычитание среднего и деление на std
 - Улучшает обучение (всегда в предобработке данных)
- Batchnorm – упрощенный whitening на слоях
 - Mean, std оцениваются по батчу
 - Mean, std – дифференцируемы

BatchNorm

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_1 \dots x_m\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \text{BN}_{\gamma, \beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^m x_i \quad // \text{ mini-batch mean}$$
$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2 \quad // \text{ mini-batch variance}$$
$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \quad // \text{ normalize}$$
$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i) \quad // \text{ scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

[Ioffe&Szegedy, 2015]

На тесте работает по другому!

Используются предпосчитанные оценки средних и дисперсий.

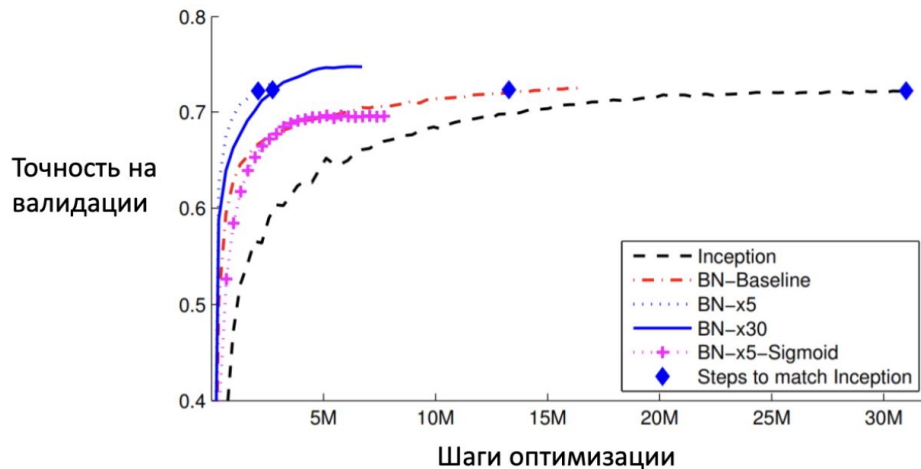
Оценки средних и дисперсий вычисляются скользящим средним во время обучения.

- Объекты перестают быть i.i.d.!

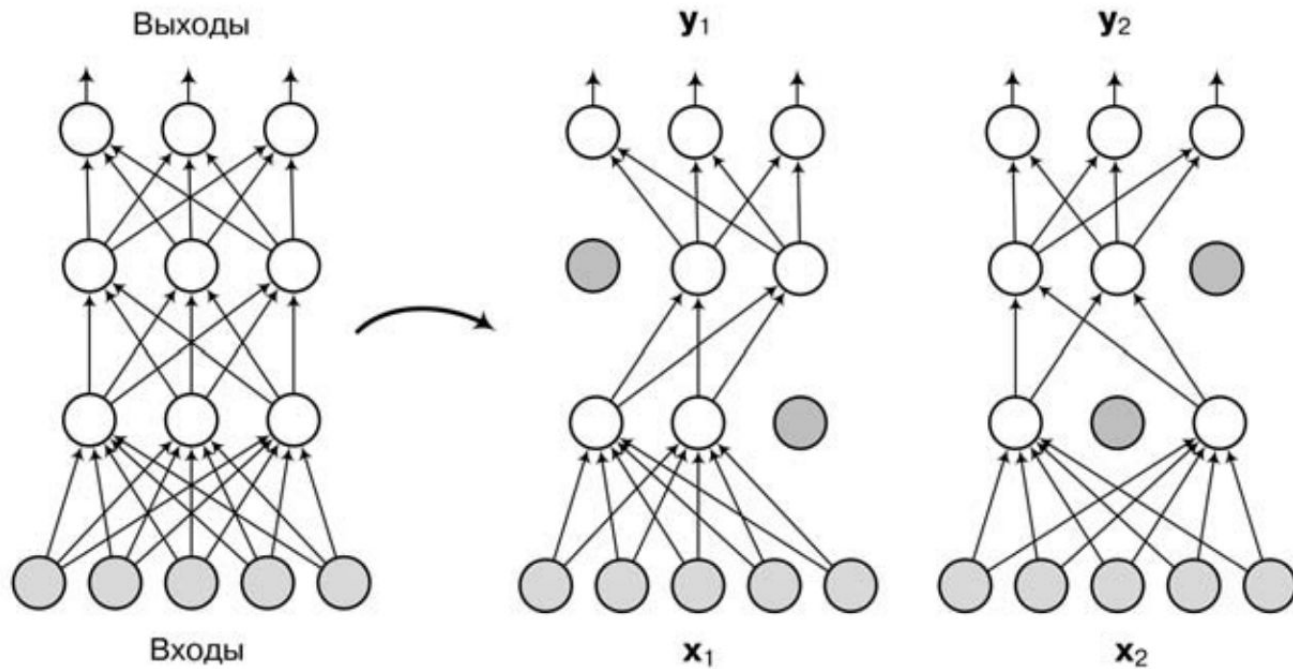
BatchNorm

[Ioffe&Szegedy, 2015]

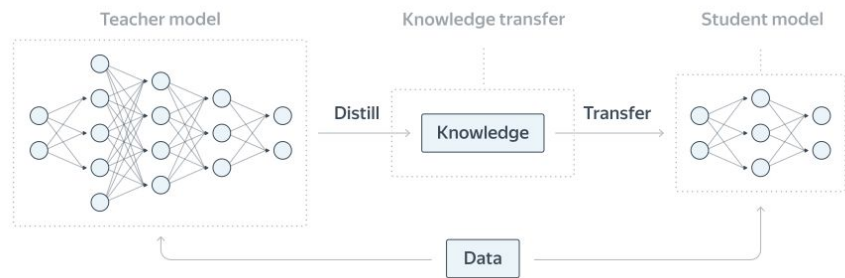
- Можно увеличить длины шага (LR)
- Можно убрать dropout
- Уменьшить L2-регуляризацию
- Позволяет обучать очень глубокие модели (ResNet)



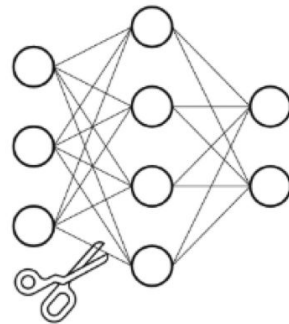
DropOut



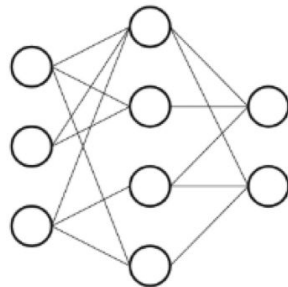
Дистилляция, квантизация, пруннинг



До:



После:

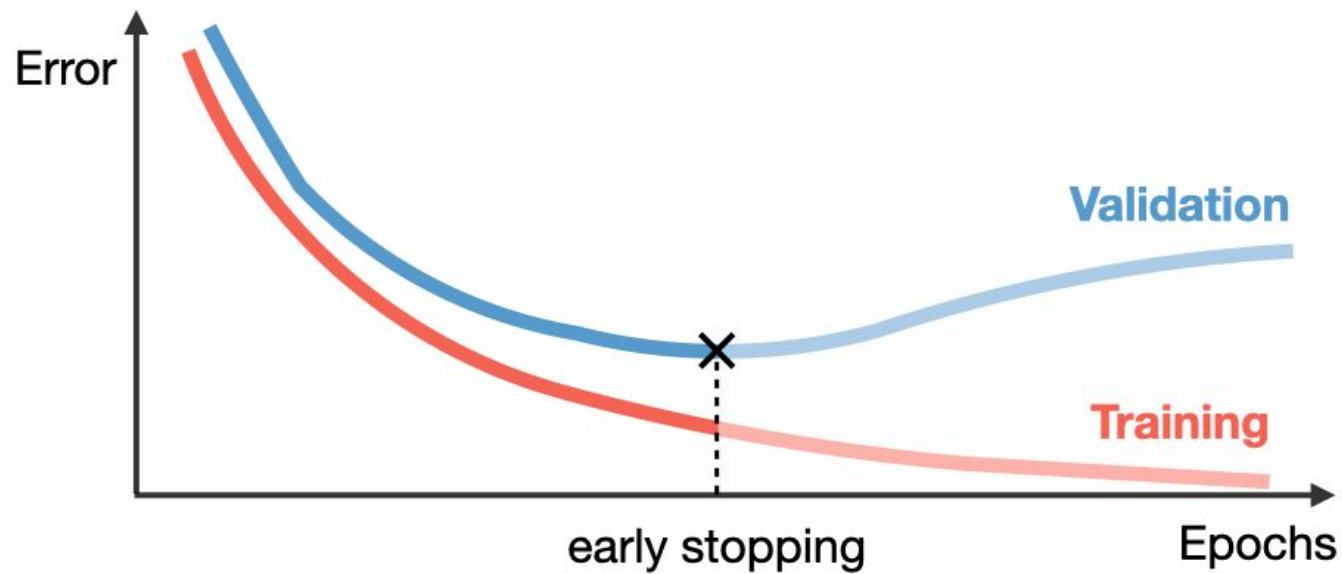


Подробнее об этих методах через одну лекцию 😊

Оптимизация

- Early stopping
- Learning rate scheduling
- Простые оптимизаторы

Early stopping



Источники

- <https://education.yandex.ru/handbook/ml>
- https://github.com/aosokin/dl_cshse_ami/tree/master/2020-fall/lectures
- Глубокое обучение. Погружение в мир нейронных сетей – Николенко С., Кадурын А.
- Глубокое обучение – Гудфеллоу Я., Бенджио И., Курвилль А.