Инициализация весов и регуляризация

ФКН

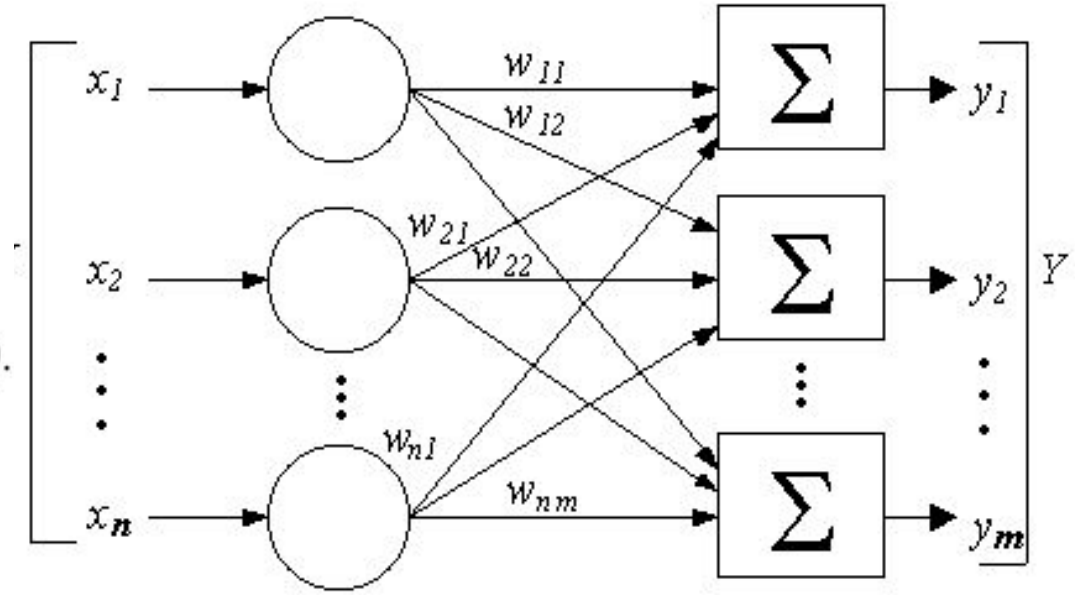
План лекции

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

Повторение*

Линейный слой

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



Два линейных слоя подряд

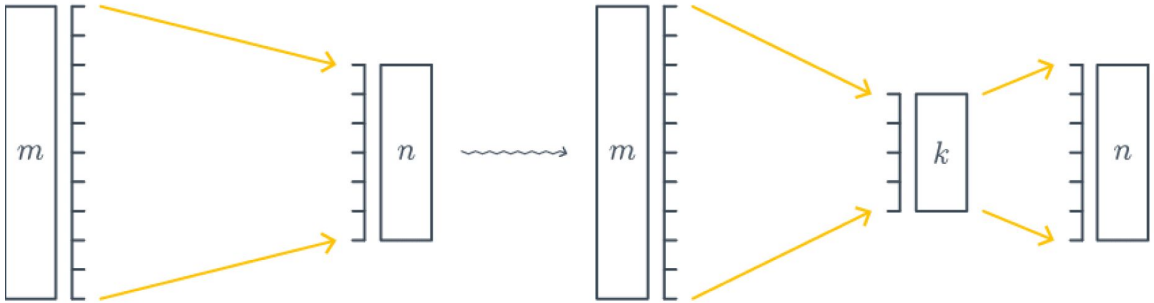
Линейная комбинация линейных отображений есть линейное отображение, то есть два последовательных линейных слоя эквивалентны одному линейному слою.

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

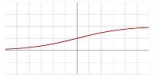
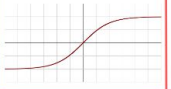
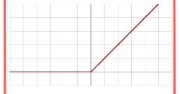
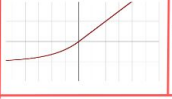
Два линейных слоя подряд

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 $k \ll m,n$:

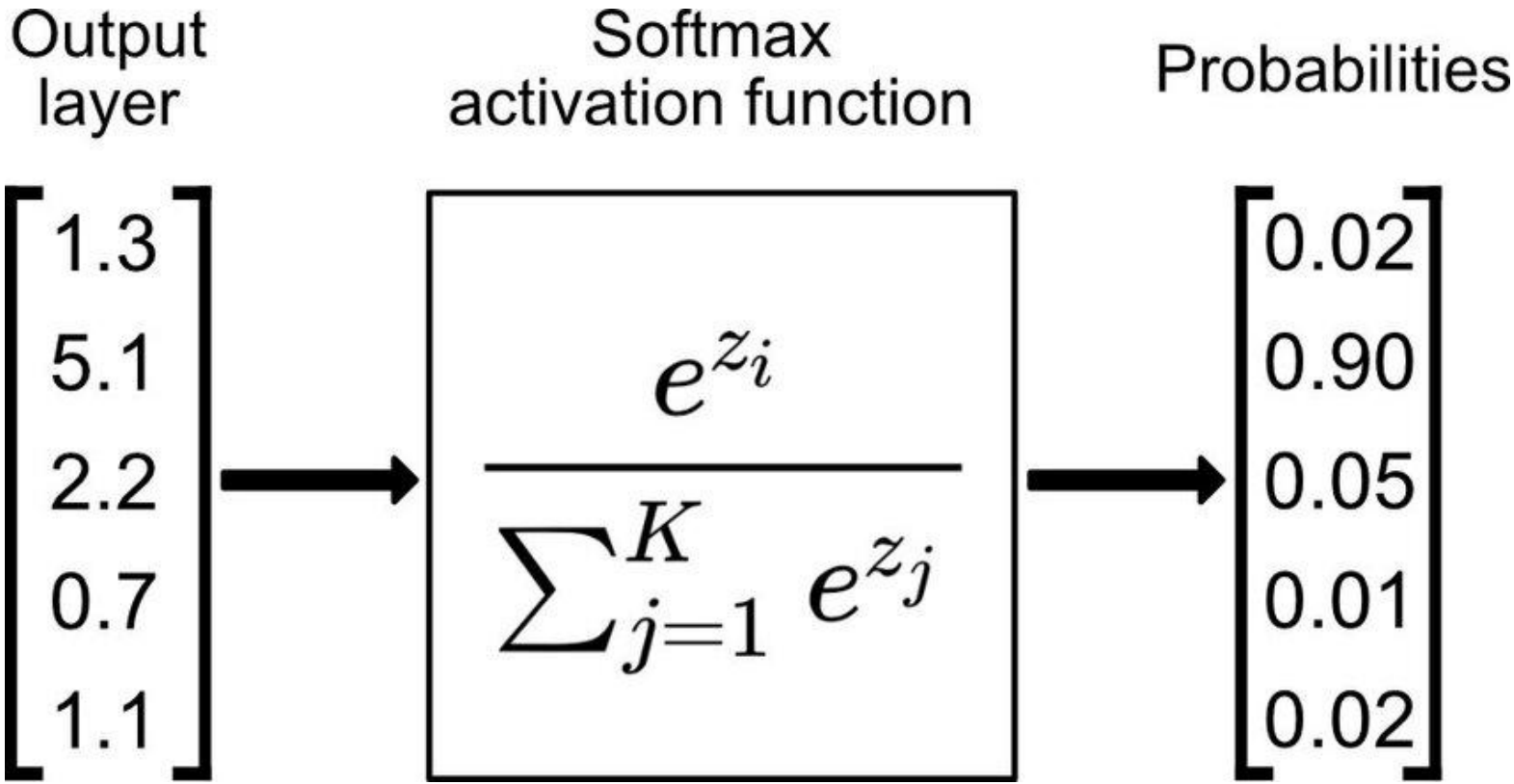
С точки зрения линейной алгебры это примерно то же самое, что потребовать, чтобы матрица исходного линейного слоя имела ранг не выше k. И с точки зрения сужения «информационного канала» это иногда может сработать. Два линейных слоя подряд стоит ставить, только если вы хорошо понимаете, чего хотите добиться.



Нелийности

ACTIVATION FUNCTION	PLOT	EQUATION	DERIVATIVE	RANGE
Linear		f(x) = x	f'(x) = 1	(-∞, ∞)
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, ∞)
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, ∞)

SoftMax


Output layer
Softmax activation function
Probabilities
1.3
5.1
2.2
0.7
1.1
$\frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$
0.02
0.90
0.05
0.01
0.02

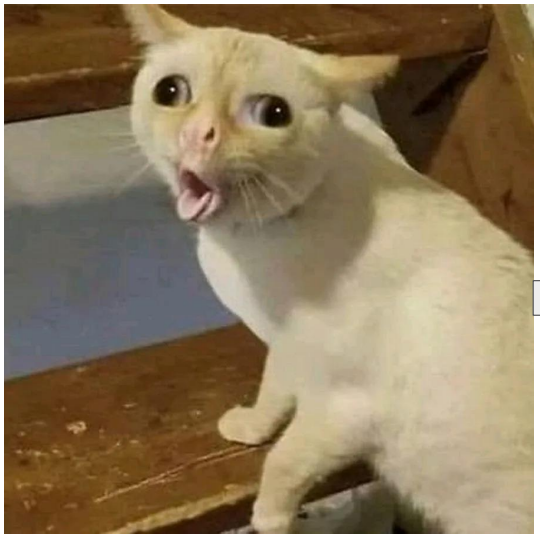
ML common knowledge

Робастность

Робастность — это устойчивость метода или модели к шуму, ошибкам и неожиданным ситуациям

neural net

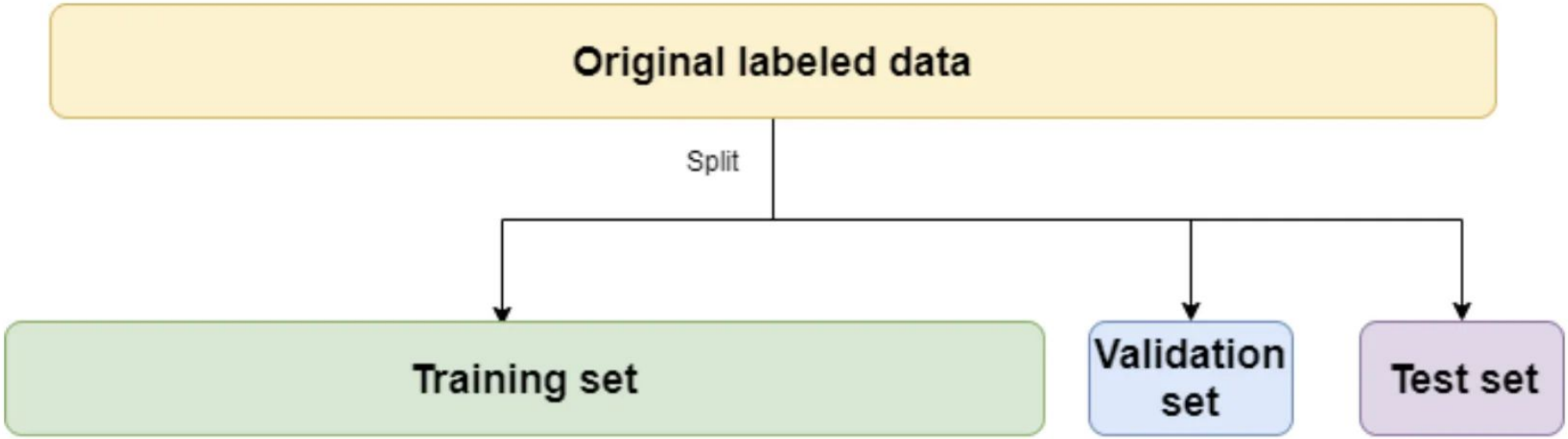
Котик



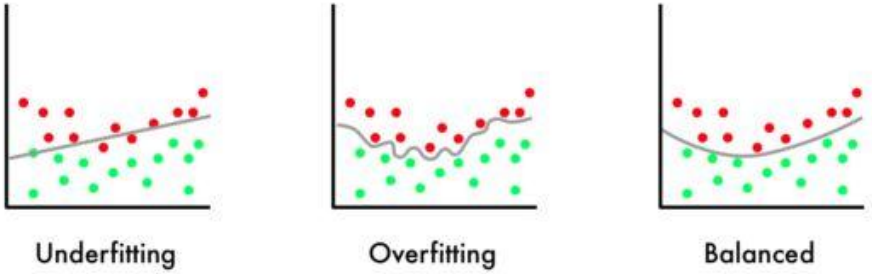
neural net

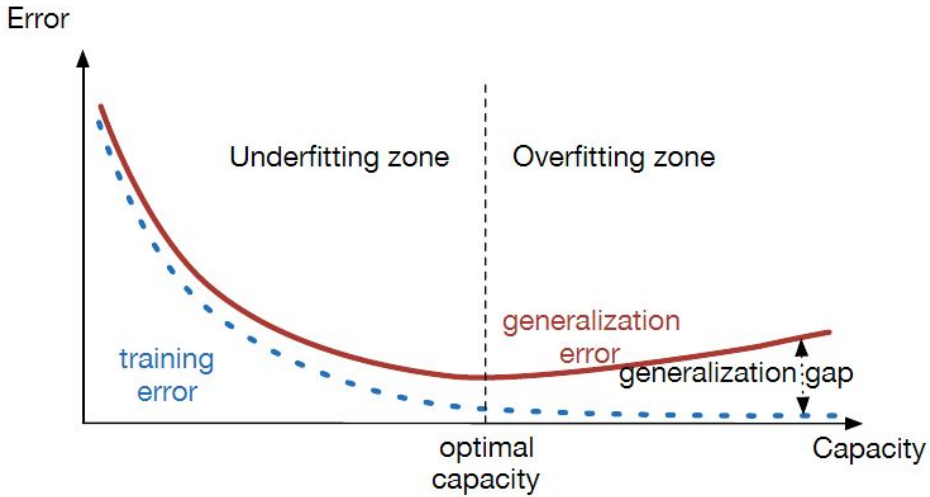
???

Train, Validation, Test



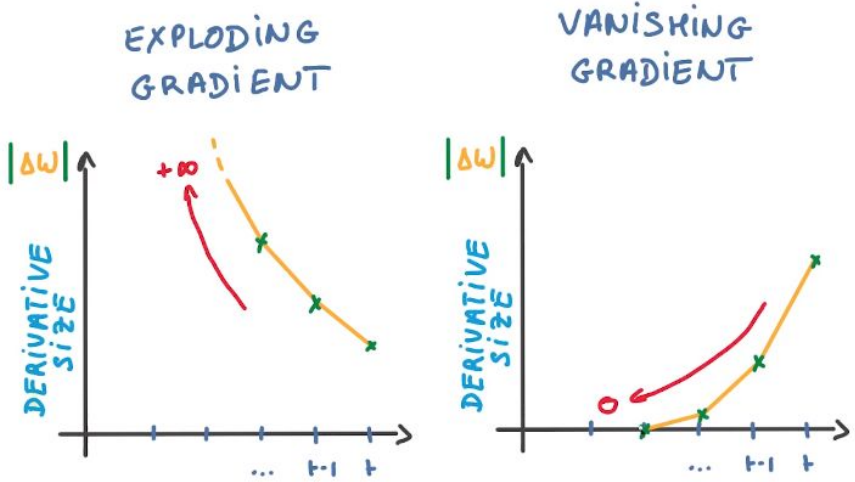
Переобучение





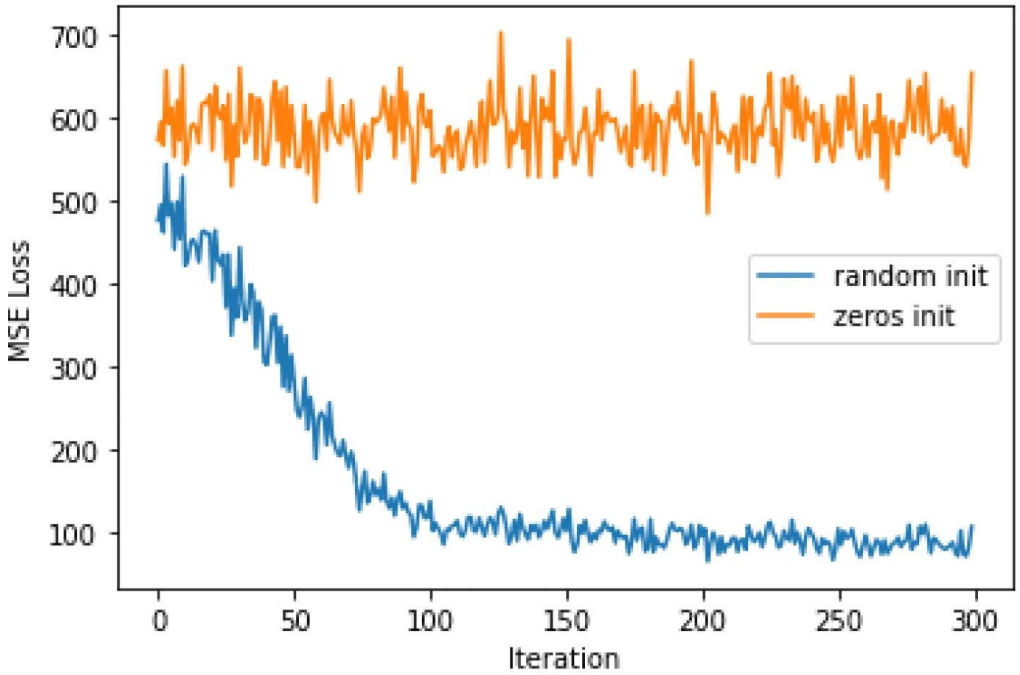
Взрыв и затухание градиента

При обратном распространении ошибки при прохождении через слои нейронной сети в элементах градиента могут накапливаться большие значения, что будет приводить к сильным изменениям весов, что сильно “разболтает” веса – взрыв градиента. Обратная ситуация – затухание.



Инициализация весов

Инициализация константой



Веса будут меняться одинаково внутри слоёв

Xavier

Рассмотрим одну активацию:

y_i

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

Тогда её дисперсия:

$$\operatorname{Var}(y_i) = \operatorname{Var}(w_i x_i) = \mathbb{E}\left[x_i^2 w_i^2\right] - (\mathbb{E}[x_i w_i])^2 = \quad \leftarrow \text{ } x \text{ и } w \text{ независимы}$$

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

Используем симметричную функцию активации (значит мат. ож. $x = 0$) и берём распределение w с мат. ожиданием в 0

Xavier

Рассмотрим одну активацию:

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

y_i

Тогда её дисперсия:

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

x и *w* независимы

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

Используем симметричную функцию активации (значит мат. ож. $x = 0$) и берём распределение *w* с мат. ожиданием в 0

Xavier

Итого получаем: $\mathrm{Var}(y_i) = \mathrm{Var}(w_i)\,\mathrm{Var}(x_i)$

Если предположить, что *x* и *w* сэмплируются независимо из одного распределения (сильное утверждение), то получаем:

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

число нейронов на выходе слоя

Т.е. дисперсия выходов пропорциональна дисперсии входов с коэффициентом $n_{out}Var(w_i)$

Xavier

Ранее часто веса инициализировали из распределения:

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

Из-за чего дисперсия активации падала втрое с каждым слоем, т.е. сигнал “затухал”:

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

Xavier (backward pass)

На обратном проходе похожая ситуация. Если $z^{(l+1)} = f\left(y^{(l)}\right)$, где *l* – номер слоя, а *f* – функция активации, то обратное распространение ошибки выглядит следующим образом:

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

~1 для симметричной *f* с единичной производной в нуле (напр., tanh)

Т.е. дисперсия градиентов пропорциональна дисперсии выходов с коэффициентом $n_{in} Var(w_i)$

Xavier (idea)

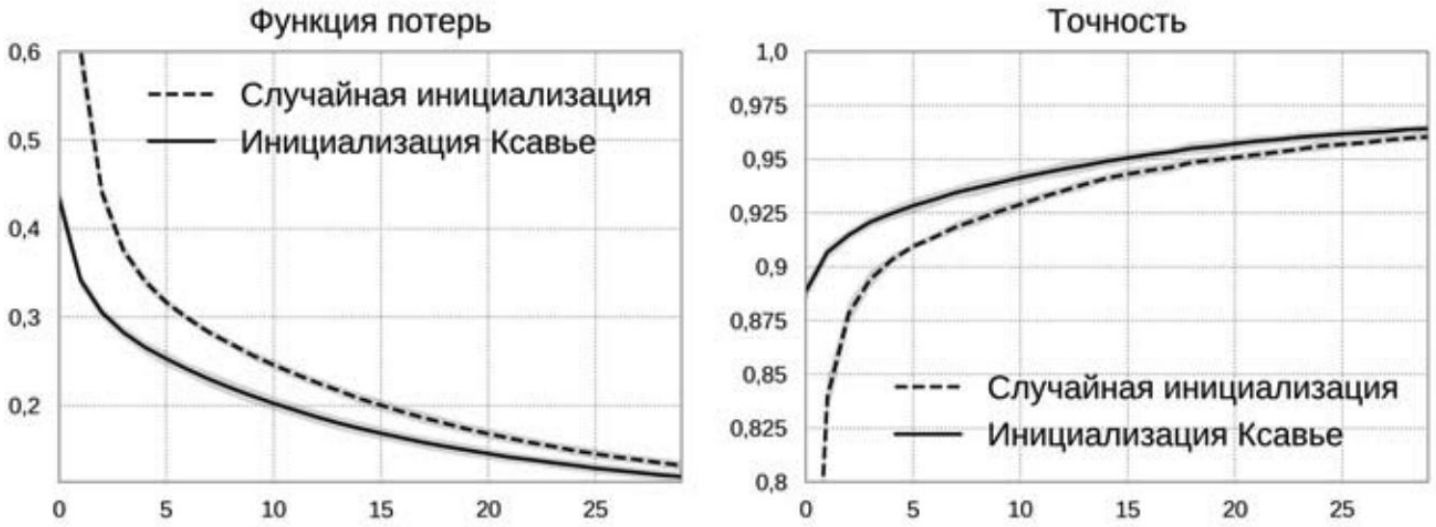
Чтобы не затухал ни сигнал, ни градиент, надо сэмплировать веса из распределения с дисперсией:

$$\mathrm{Var}\left(w_i\right) = \frac{2}{n_{\mathrm{in}} + n_{\mathrm{out}}}$$

Тогда для равномерного распределения в центре с нулём, границы будут такие:

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

Инициализация просто нормальным распределением



Kaiming (for ReLU)

Но функция активации не всегда симметрично относительно нуля. Есть популярный ReLU, для отбросить первое слагаемое не получится:

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

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

Дальше выкладки похожи на Xavier, но явно используется формула ReLU. Для него выводится следующее оптимальное распределение:

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

Регуляризация нейросетей

Регуляризация нейросетей

Техники для борьбы с переобучением, повышения робастности 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 пара наиболее распространённых примеров: L_1, L_2 -регуляризации (a.k.a.weight decay)

Регуляризатор может быть применён как к весам модели, так и к активациям/выходам, в зависимости от желаемых целей

L_2-ругуляризация

- 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

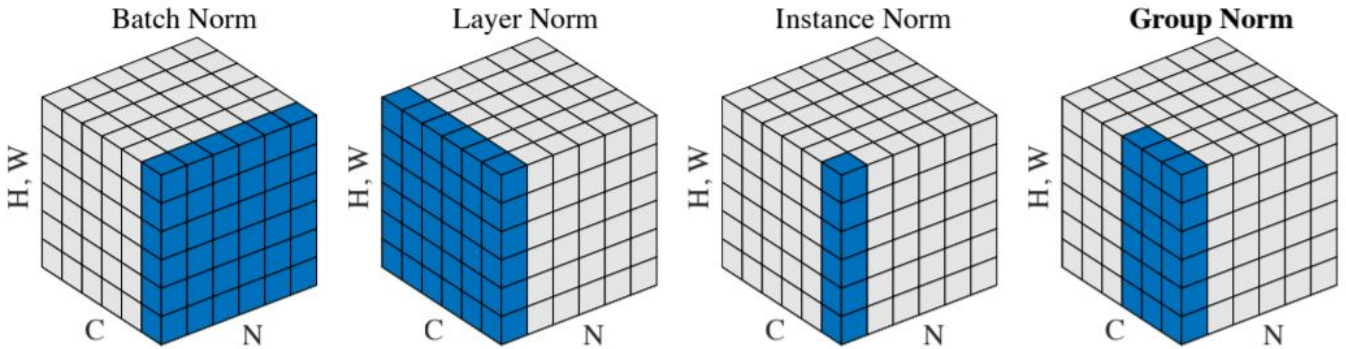
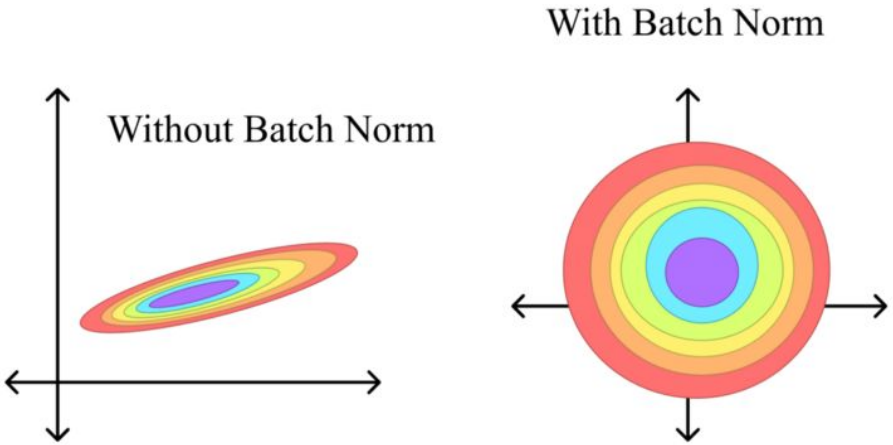


Figure 2. **Normalization methods**. Each subplot shows a feature map tensor, with N as the batch axis, C as the channel axis, and (H, W) as the spatial axes. The pixels in blue are normalized by the same mean and variance, computed by aggregating the values of these pixels.

BatchNorm

Когда веса нижележащих слоёв обновляются, распределение активаций на входе последующих слоёв "дрейфует", из-за чего обучение становится менее стабильным и требует более аккуратного подбора learning rate и инициализации. Это называется Internal Covariate Shift, с ним борется Batch Normalization и его аналоги.



BatchNorm

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

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

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

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

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

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

[Ioffe&Szegedy, 2015]

На тесте работает по другому!

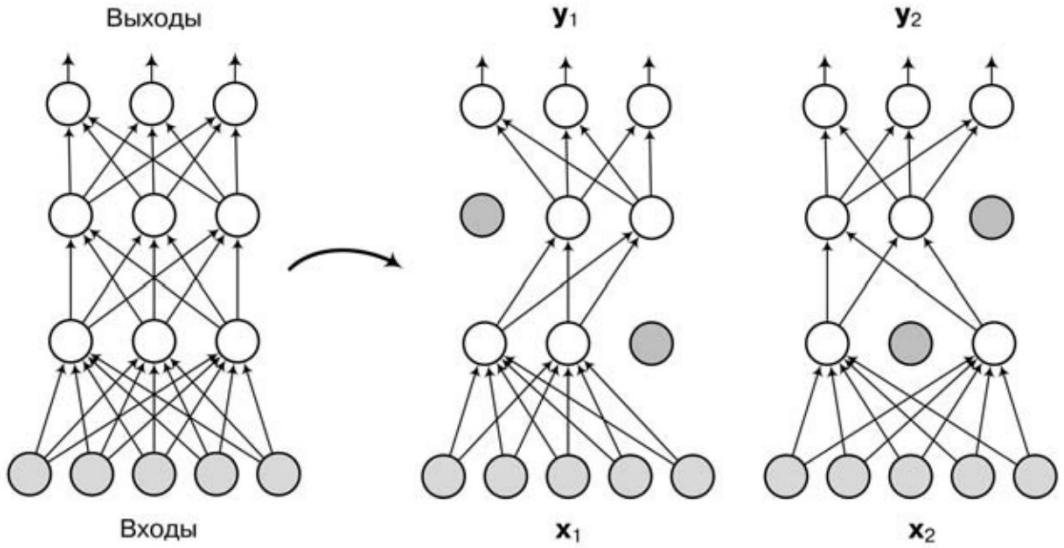
Используются предпосчитанные оценки средних и дисперсий.

Оценки средних и дисперсий вычисляются скользящим средним во время обучения.

u, sigma – статистики, накапливаемые по батчу
gamma, beta – параметры, обучаемые градиентным спуском

DropOut

Оставляем каждую активацию с вероятностью 1 - p, а с вероятностью p приравниваем её 0



На последнем слое DropOut обычно не делают, т.к. нужен вектор определённой размерности

DropOut. А как применять?

Сэмплировать “прореженные” DropOut’ом сети, применять их все, а затем усреднять

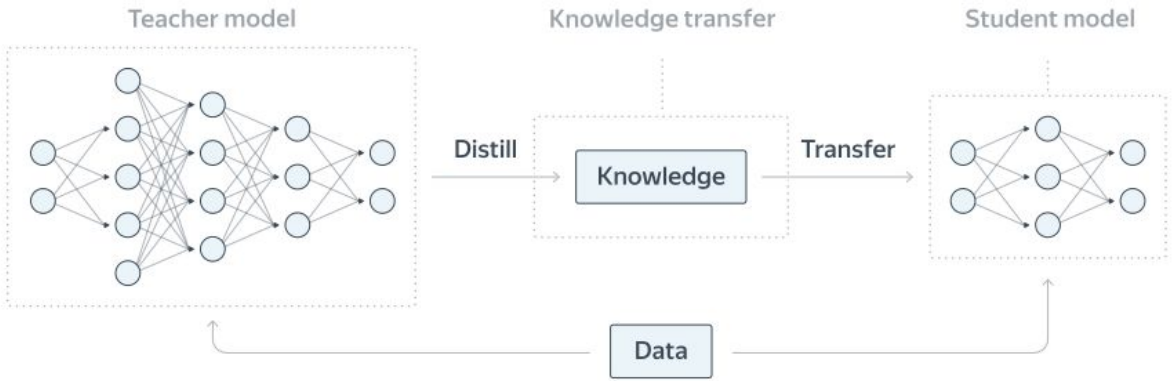
Оказывается это примерно эквивалентно применению сети без “прореживаний”, в которой каждая активация умножается на 1 - p, благодаря чему сохраняем их математическое ожидание

Это объясняет и мотивацию: DropOut позволяет нам в некотором смысле усреднить все возможные под-архитектуры нашей сети, сделать активацию каждого нейрона более информативной.

DropOut. Техническая реализация

	DropOut	Inverted DropOut
Обучение	$h_{\text{train}} = m \odot h, \quad m \sim \text{Bernoulli}(1-p)$	$h_{\text{train}} = \frac{m}{1-p} \odot h, \quad m \sim \text{Bernoulli}(1-p)$
Инференс	$h_{\text{test}} = (1-p) \cdot h$	$h_{\text{test}} = h$

Дистилляция



Учим маленькую модель повторять предсказания большой модели. Её возможности более ограничены, поэтому ей не удаётся переобучаться, как большой модели

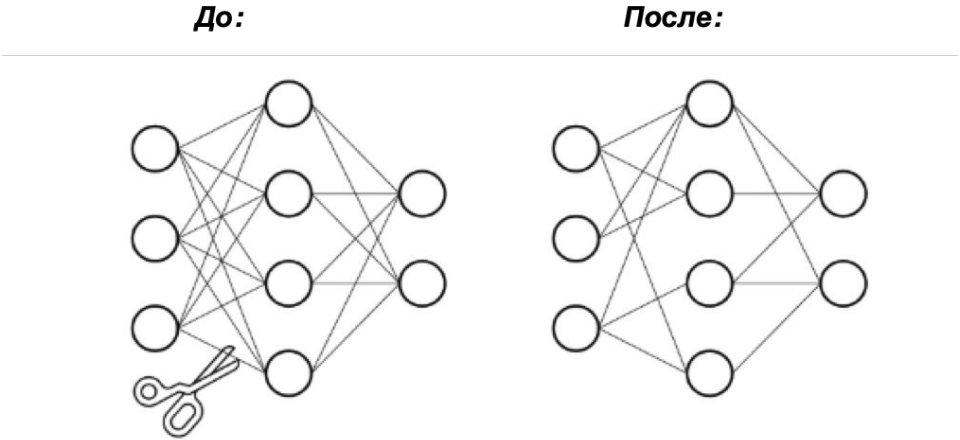
Подробнее об этом методе через одну лекцию 😀

Квантизация

Веса модели переводим с высокой точности (fp32) к более низкой (fp16/fp8/fp4/int8). Модель снова не может так сильно переобучаться под данные и вынуждена описывать закономерности проще

Подробнее об этом методе через одну лекцию 😀

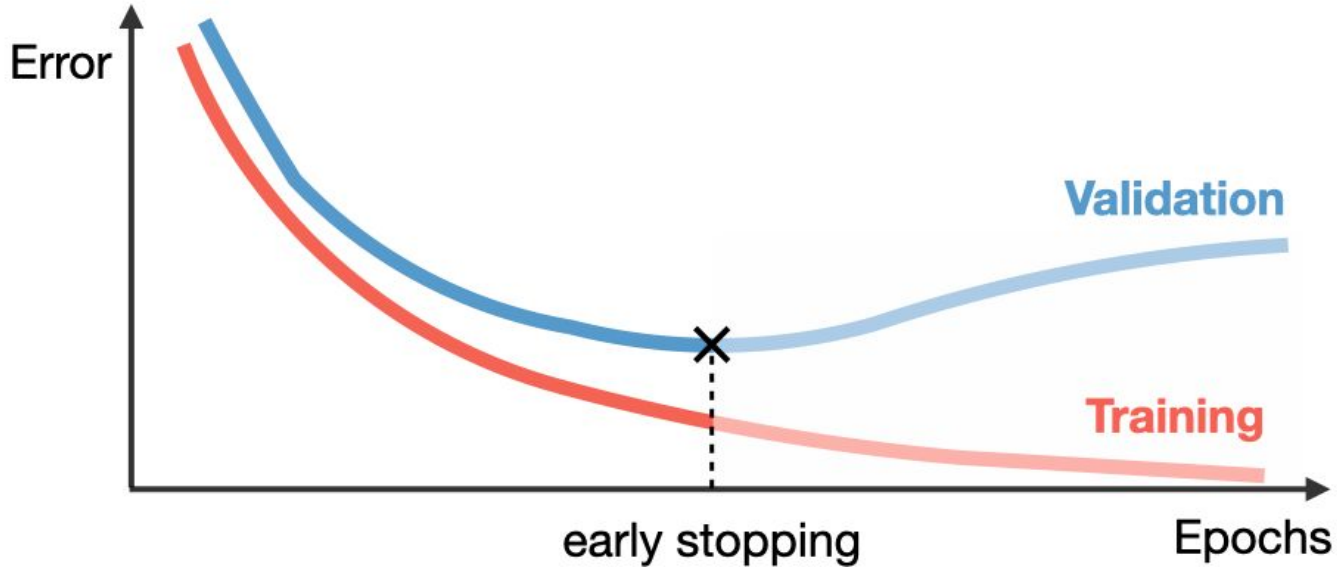
Прунниг



Выбираем критерий важности для весов, наименее важные веса убираем и дообучаем модель. Архитектура модели становится всё более специфичной, модель держится за “важные” признаки и игнорирует “шумные”

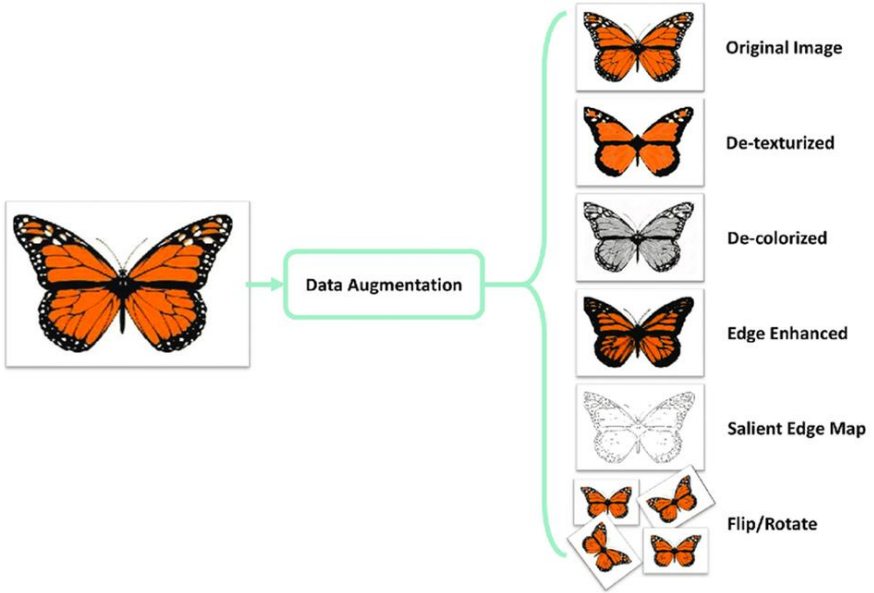
Подробнее об этом методе через одну лекцию 😀

Оптимизация. Early stopping



Изменение данных

- Аугментации – устойчивость к некоторым преобразованиям
  - повороты и отражения картинок
  - замена случайных слов в тексте на синонимы



- Data mixup – создание промежуточных примеров через линейную комбинацию входов и меток
- Label smoothing – заменяем строгие метки классификации 0/1 на “вероятности”, позволяя модели “сомневаться в разметке”

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., Кадурин А.
- Глубокое обучение – Гудфеллоу Я., Бенджио И., Курвилль А.